A Coalgebraic Approach to Reducing Finitary Automata

Keri D'Angelo
Department of Computer Science, Cornell University, Ithaca, NY, USA

Alexandra Silva
Department of Computer Science, Cornell University, Ithaca, NY, USA

Abstract

Compact representations of automata are important for efficiency. In this paper, we study methods to compute *reduced automata*, in which no two states accept the same language. We do this for *finitary automata* (FA), an abstract definition that encompasses probabilistic and weighted automata. Our procedure makes use of Milius' locally finite fixpoint. We present a reduction algorithm that instantiates to probabilistic and $\mathbb{S}$-linear weighted automata (WA) for a large class of semirings. Moreover, we propose a potential connection between properness of a semiring and our provided reduction algorithm for WAs, paving the way for future work in connecting the reduction of automata to the properness of their associated coalgebras.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases coalgebras, finitary automata, state reduction

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Given a language L it is natural to ask whether there exists a *small* automaton with a state that that accepts L. Whereas the notion of smallest is clear for deterministic automata, the same is not the case for other types of automata. A deterministic automaton is state-minimal iff all its states are reachable and no state is redundant; that is, no state accepts the same language as any other state. Hence, given a deterministic automaton we can minimize it by eliminating redundant states (and then take reachability from the initial state). Unfortunately, this property does not hold for more general automata: eliminating redundant states does not necessarily yield a minimal probabilistic or non-deterministic automaton, though it does yield a smaller automaton, which we call *reduced* and which is interesting to study as procedures to compute reduced automata are simpler than minimization procedures (see e.g. [11] for probabilistic automata). In particular, we will look at reduction procedures for *finitary automata*, an abstract definition that encompasses probabilistic automata (PA) and weighted automata (WA).

Let $\mathcal{T} : \mathbf{Set} \to \mathbf{Set}$ be a finitary monad. Finitary automata are automata that have an output function $\mathsf{out} : S \to O$ and a transition function $\delta : S \to \mathcal{T}(S)^A$. If we assume that $O \cong \mathcal{T}(Y)$ for some finite set Y, we can lift the output and transition maps to operate on elements of $\mathcal{T}(S)$—$\mathsf{out}^\sharp : \mathcal{T}(S) \to O$ and $\delta^\sharp : \mathcal{T}(S) \to \mathcal{T}(S)^A$—and this is useful in defining the semantics of the automata, which we will detail later. Typically, automata also include an initial state. Here, we do not include the initial state in the definition and will study the semantics parametric on a state. Therefore, we simply write finitary automata as pairs $\mathcal{A} = (S, \langle \mathsf{out}, \delta \rangle)$ (these are coalgebras for the functor $O \times \mathcal{T}(-)^A$).

Finitary automata are a direct generalization of non-deterministic automata: the difference between them is the type of *next* state. In one case, it is a non-deterministic choice, whereas in the other it is a *combination*. By combination, we mean an element of $\mathcal{T}(X)$. These notions are instances of *side-effects* captured by a monad [15]. PAs and WAs are special cases by taking specific monads to capture these side-effects (see Section 2). Besides PAs and WAs, some other types of automata that also have this abstract view include nominal and quantum automata.

In this paper, we focus on the notion of a *reduced automaton*. Given an automaton, we say it is *reduced* if no state is redundant (what we call reduced is called observable in [3]). In the cases of PAs and WAs, the notion of a state being redundant is more subtle than for deterministic automata: a state is redundant if its language is a combination of the languages of other states; i.e. for PAs, a *convex* combination, and for WAs, a *linear* combination. A deterministic automaton that is reduced will be minimal when eliminating unreachable states. This does not hold for finitary automata: it is possible that an automaton may be reduced, and yet once we eliminate unreachable states from a specific start state, we obtain an automaton that is not state minimal (see Example 10).

We study the problem of reducing finitary automata coalgebraically. As described earlier, finitary automata can be lifted from **Set** to $\mathbf{EM}(\mathcal{T})$, where **Set** is the category of sets and set functions, and $\mathbf{EM}(\mathcal{T})$ is the Eilenberg-Moore category over the monad $\mathcal{T}$. If there exists a notion of a *base* for all free and finitely generated elements of $\mathcal{T}(X)$ and there exists an algorithm to find such a minimal base, then a reduced automaton can always be calculated. We use the word base rather than basis since we do not assume the base is unique nor does every element need to be able to be written as a *unique* combination of the base. It becomes clear why this base condition is necessary in Section 4.2.

This type of work has been done for deterministic automata which have state spaces in **Set**. To reduce deterministic automata, one can take the epi-mono factorization of the map from the state space X into the carrier of the final coalgebra $\mathbf{2}^{A^*}$ in **Set**, where $\mathbf{2}$ represents the two element set. This approach cannot be taken for $\mathbf{EM}(\mathcal{T})$ categories in general, where quotients are not always well-understood. In this paper, we present a generalized algorithm for finitary automata in $\mathbf{EM}(\mathcal{T})$, and explicitly describe the procedure for the special cases of probabilistic automata and $\mathbb{S}$-linear weighted automata.

The coalgebraic perspective shows that reduced automata can be viewed as an image in a canonical object, ϑF, the locally finite fixpoint of a functor F (Definition 11): it can be constructed in an analogous fashion to the determinization of non-deterministic automata followed by a minimization procedure in a category with extra algebraic structure. The locally finite fixpoint is the colimit of the inclusion functor from the category of coalgebras with finitely generated carrier into the full coalgebra category, and under certain assumptions, the locally finite fixpoint is a subcoalgebra of the final coalgebra. We show finitary automata satisfy these assumptions. In the case of probabilistic automata, this extra structure gives rise to procedures in $\mathbf{EM}(\mathcal{D})$, the category of convex algebras and convex maps. In the case of $\mathbb{S}$-linear weighted automata, this extra structure gives rise to procedures in $\mathbf{EM}(\mathcal{S}_{\mathbb{S}})$, the category of semimodules and semimodule homomorphisms (see Example 2 and Example 4).

While both reduced and state-minimal automata offer gains in space efficiency for representing languages, we argue in this paper that though state minimal automata are perhaps the most obvious small representation to study, there are advantages in considering reduced automata as compact acceptors of languages. In particular, it is known for probabilistic automata that minimization is NP-hard [11] whereas the decision problem for reduced PA is in polynomial time (this is a consequence of Theorem 30).

Finally, we begin forming a connection between properness and reducibility. Properness of semirings was first introduced in [8], but has more recently been extended to describe functors [13]. If a functor is proper, the rational fixpoint, which is similar to the locally finite fixpoint used in this paper, is a subcoalgebra of the final coalgebra. This allows for understanding the behavior of automata.

In a nutshell, the main contributions (and structure) of the paper are as follows:

1. In Section 3, we prove a concrete formula for ϑF where F is finitary over $\mathbf{EM}(\mathcal{T})$ that enables its computation in **Set**—this can be seen as one of the main advantages of the coalgebraic outlook on the problem offering guidance on how to generalize reduction to more types of automata;
2. A characterization of a reduced finitary automaton by viewing its image in the locally finite fixpoint ϑF is shown in Section 3;
3. Two special cases of this framework are PAs and $\mathbb{S}$-linear WAs (for a large class of semirings $\mathbb{S}$). In Section 4, we give the complete construction of these procedures. For probabilistic automata, we provide a complete algorithm on how to explicitly compute the reduced automaton as well as a result (Theorem 30) that implies the decision problem for reduced PA is in polynomial time.
4. We discuss directions for future work in Section 5, including a detailed explanation on how our work connects to proper semirings/functors.

2 Preliminaries

We assume basic knowledge of category theory (functors and natural transformations). We first begin with some facts on monads and then coalgebras.

▶ **Definition 1** (Monad). *A monad is a triple (T, η, μ) where $T\colon \mathbf{C} \to \mathbf{C}$ is a functor on a category $\mathbf{C}$, $\eta\colon X \to TX$ and $\mu\colon TTX \to TX$ are natural transformations (called the unit and multiplication, respectively) satisfying $\mu \circ \eta = id = \mu \circ T\eta$ and $\mu \circ \mu = \mu \circ T\mu$.*

(T, η, μ) is a finitary monad if the functor T commutes with filtered colimits.

▶ **Example 2** (Monads). We give examples of three monads on **Set**.

1. The powerset monad is given by $\mathcal{P}(X) = \{U \mid U \subseteq X\}$, $\eta(x) = \{x\}$, and $\mu(\Phi) = \bigcup_{S \in \Phi} S$.
2. The (finitely supported) distribution monad is given by $\mathcal{D}(X) = \{\phi \mid \sum \phi(x) = 1\}$ (from Definition 19), with natural transformations

$$\eta(x) = \lambda y. \begin{cases} 1 & y = x \\ 0 & \text{otherwise} \end{cases} \qquad\qquad \mu(\Phi)(x) = \sum_{\varphi \in \mathbf{supp}(\Phi)} \varphi(x) \times \Phi(\varphi).$$

3. The (finitely supported) free semimodule monad for a semiring $\mathbb{S}$ (with unity) is given by $\mathcal{S}_{\mathbb{S}}(X) = \{\varphi : X \to \mathbb{S} \mid \sum_{i=1}^{n} \varphi(x_i) \cdot x_i\}$ (from Definition 31), with natural transformations

$$\eta(x) = \lambda y. \begin{cases} 1 & y = x \\ 0 & \text{otherwise} \end{cases} \qquad\qquad \mu(\Phi)(x) = \sum_{\varphi \in \mathbf{supp}(\Phi)} \varphi(x) \times \Phi(\varphi).$$

$\mathcal{P}$, $\mathcal{D}$, and $\mathcal{S}_{\mathbb{S}}$ are all examples of finitary monads.

A *coalgebra* is a pair $(X, t\colon X \to \mathcal{F}X)$, where X is the *state space* and t is the transition dynamics which is parametric on a functor $\mathcal{F}\colon \mathbf{C} \to \mathbf{C}$. Coalgebras cover a range of automata types: deterministic automata are coalgebras for the functor $D(X) = 2 \times X^A$ on **Set** (the category of sets and functions), weighted automata (over a field $\mathbb{F}$) are coalgebras for the functor $W(V) = \mathbb{F} \times V^A$ on **Vect** (the category of vector spaces and linear functions), and as we will see probabilistic automata can be seen as coalgebras in **Conv** (the category of convex sets and convex maps). The advantage of studying different automata as coalgebras is that many important notions are fully determined by $\mathcal{F}$, e.g homomorphisms, behavioral equivalence. More recently, there's been a research effort in showing that $\mathcal{F}$ is also rich enough to design abstract algorithms, including minimization algorithms [12, 6, 21, 4]. An important concept in coalgebra is *finality*: a coalgebra $(Z, z\colon Z \to \mathcal{F}Z)$ is *final* if for all other coalgebras $(X, f\colon X \to \mathcal{F}X)$ there exists a unique structure-preserving homomorphism $[\![-]\!]\colon X \to Z$, as depicted on the right.

$$\begin{tikzcd} X \arrow[r, "{[\![-]\!]}"] \arrow[d, "f"'] & Z \arrow[d, "z"] \\ \mathcal{F}X \arrow[r, dashed] & \mathcal{F}Z \end{tikzcd}$$

Final coalgebras are an abstract way of capturing behavior: any state on another coalgebra can be mapped uniquely into it. Hence, $[\![x]\!]$ is a canonical representative of the behavior denoted by x. For concrete functors, this instantiates to familiar things:

▶ **Example 3** (Final coalgebras)**.** We give examples of final coalgebras in **Set** and **Vect**.

1. The final coalgebra of $D(X) = 2 \times X^A$ on **Set** is the pair $(2^{A^*}, \langle \varepsilon?, \partial\rangle)$, with the set of languages over A as the carrier and the transition structure given by language derivatives:

$$\varepsilon?(L) = \begin{cases} 1 & \varepsilon \in L \\ 0 & \text{otherwise} \end{cases} \qquad\qquad \partial(L)(a) = \{w \mid aw \in L\}$$

   $[\![-]\!]$ assigns to a state x of a deterministic automaton the regular language accepted by x.
2. The final coalgebra of $W(V) = \mathbb{R} \times V^A$ on **Vect** is the pair $(\mathbb{R}\langle\langle A^*\rangle\rangle, \langle o, t\rangle)$, with the set of weighted languages (formal power series) over A as the carrier and the transition structure given by the linear maps: $o(\sigma) = \sigma(\varepsilon)$ and $t(\sigma)(a)(w) = \sigma(aw)$. $[\![-]\!]$ assigns to a state x the rational power series denoting the weighted language accepted by x.

The two examples of final coalgebras above are actually of similar nature (hence the closeness in their definitions). The second example is in the category of vector spaces and linear maps, which is an instance of a category of algebras for a monad, a concept we recall next (the first example is also an instance albeit for the simple identity monad!).

Given a monad T, an algebra for T (or T-algebra) is a pair $(X, h\colon TX \to X)$ where h, the algebra map, satisfies $h \circ \eta = id$ and $h \circ Th = h \circ \mu$. The category of algebras for a monad T, also called the category of *Eilenberg-Moore algebras for* T, denoted $\mathbf{EM}(T)$, has T-algebras as objects and structure preserving maps morphisms.

▶ **Example 4** (Algebras for a Monad)**.** We instantiate T-algebras for the monads of Example 2.

1. $\mathbf{EM}(\mathcal{P})$ is the category of join-semilattices and join-preserving maps.
2. $\mathbf{EM}(\mathcal{D})$ is the category of convex sets and convex maps.
3. $\mathbf{EM}(\mathcal{S}_\mathbb{S})$ is the category of semimodules over the semiring $\mathbb{S}$ and $\mathbb{S}$-linear maps.

The second example above could also be presented with affine maps, as we have the following result from [19]: For a convex set P and any $Q \subseteq E$ for E a real coordinate space, f preserves convex combinations iff f preserves affine combinations.

We can now show how the two examples in Example 3 are in fact an instance of the same result [10]:

▶ **Theorem 5.** *Let T be a monad (in* **Set***) and O be an algebra for T. The final coalgebra of the functor $M(X) = O \times X^A$, where $M\colon \mathbf{EM}(T) \to \mathbf{EM}(T)$, is the set of O-weighted languages O^{A^*}, is as follows:*

$$\begin{tikzcd} X \arrow[r, dashed, "{[\![-]\!]}"] \arrow[d, "f"'] \arrow[dr, phantom, "\star"] & O^{A^*} \arrow[d, "\cong"', "z"] \\ O \times X^A \arrow[r, dashed] & O \times (O^{A^*})^A \end{tikzcd} \qquad z(\varphi) = \langle \varphi(\varepsilon), \lambda a.\lambda w.\varphi(aw)\rangle$$

3 Characterizing Reduced Finitary Automata

The type of automata we study in this paper are what we call finitary automata. Finitary automata use a finitary monad in their transition structure. For the remainder of this paper, we assume the monad $\mathcal{T}$ to denote a finitary monad on **Set**.

▶ **Definition 6** (Finitary Automata (FA)). *A finitary automaton is a pair $(Q, \langle \delta, \mathsf{out} \rangle)$, consisting of a set of states Q, a transition function $\delta \colon Q \to \mathcal{T}(Q)^A$, and an output function $\mathsf{out} \colon Q \to O$, where $\mathcal{T} : \mathbf{Set} \to \mathbf{Set}$ is a finitary monad, and $O \cong \mathcal{T}(L)$ for some set L.*

▶ Remark 7. Later, we will further assume the monad $\mathcal{T}$ satisfies a *base condition*, which we define to mean that for all free and finitely generated carriers in $\mathbf{EM}(\mathcal{T})$, that is, the carrier is of the form $\mathcal{T}(X)$ for some finite set X, there exists an algorithm to find a *base* (a minimal generating set, not necessarily unique) $\overline{X} \subseteq X$ such that $\mathcal{T}(\overline{X}) \cong \mathcal{T}(X)$. We will see an example in Section 4.2 why this base condition is necessary.

Finitary automata are coalgebras for the **Set** functor $O \times \mathcal{T}(-)^A$. In order to understand how finitary automata work as acceptors of languages we use a coalgebraic generalization of the usual subset construction for non-deterministic automata [10] in the following way:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "{\langle \mathsf{out}, \delta \rangle}"'] & \mathcal{T}(X) \arrow[r, dashed, "{[\![-]\!]}"] \arrow[dl, "{\langle \mathsf{out}^\sharp, \delta^\sharp \rangle}"] \arrow[d, phantom, "\circledast"] & O^{A^*} \arrow[d, "\cong"'] \\
O \times \mathcal{T}(X)^A \arrow[rr, dashed] & & O \times (O^{A^*})^A
\end{tikzcd}$$

Here, η is the monad unit and, because $O \cong T(L)$, the coalgebra $\langle \mathsf{out}, \delta \rangle \colon X \to O \times \mathcal{T}(X)^A$ in **Set** can be lifted to a coalgebra $\langle \mathsf{out}^\sharp, \delta^\sharp \rangle \colon \mathcal{T}(X) \to O \times \mathcal{T}(X)^A$ in $\mathbf{EM}(\mathcal{T})$ (see e.g. [10]).

The commutativity of $\circledast$ comes from Theorem 5 and therefore $[\![-]\!]$ is actually a homomorphism in $\mathbf{EM}(\mathcal{T})$. For intuition, if we replace $\mathcal{T}$ by the powerset monad in the above diagram (with $\eta(x) = \{x\}$) and O by $\mathbf{2}$, the two-element set, the above automata would coincide with non-deterministic automata (NFA) and both the final state and transition functions would correspond to the well-known subset construction: e.g. $\mathsf{out}^\sharp(U) = 1 \iff \exists u \in U.\ \mathsf{out}(u) = 1$. The map $[\![-]\!]$ would assign to state $\{x\}$ the language it accepts (as defined for an NFA).

We will often use L_x to denote the language accepted by a state, which is $[\![\eta(x)]\!]$.

Minimal deterministic automata have a universal property: they are unique (up-to isomorphism) for a given language. In the cases of non-deterministic and probabilistic automata, there are non-isomorphic state minimal automata accepting the same (probabilistic) language. For this reason, algorithms to find minimal probabilistic automata are hard(er) to design. Our idea for developing an approach to reduction of automata stemmed from wanting a more efficient way to reduce the size of the state space. We will take a coalgebraic perspective which allows us to generalize reduction beyond just these two types of automata, namely, to finitary automata. The goal of this section is to show an alternative to minimal automata, which we will call *reduced automata*, and show reduced automata have a universal property similar to minimal deterministic automata.

▶ **Definition 8** (Redundant State). *Let X be the set of states. A state $x \in X$ is considered redundant if there exists some $y \in \mathcal{T}(X \backslash \{x\})$ such that x and y accept the same language; that is, $[\![\eta(x)]\!] = [\![y]\!]$.*

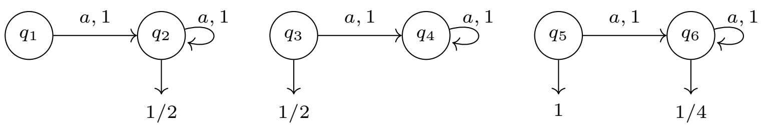
▶ **Definition 9** (Reduced automata). *An* automaton $\mathcal{A} = (Q, \langle \mathsf{out}, \delta \rangle)$ *is* reduced *if there are no redundant states.*

For example, a *probabilistic automaton* $\mathcal{A} = (Q, \langle \delta, \mathsf{out} \rangle)$ is *reduced* if there is no $q \in Q$ such that, for all $w \in A^*$

$$L_q(w) = \sum_{s \in Q \setminus \{q\}} r_s \times L_s(w) \qquad \text{and} \sum_{s \in Q \setminus \{q\}} r_s = 1, \quad r_s \in [0,1]$$

That is, L_q is not a convex combination of languages of other states in Q.

▶ **Example 10** (Minimal vs reduced). Clearly, minimal automata are reduced but not the other way around. We give an example of a PA over the alphabet $A = \{a\}$. We depict the transitions and output functions as a state diagram—missing outputs are 0.



This automaton is not reduced. The language of state q_6 can be written using the languages of q_2 and q_4 (or q_1 and q_3): $L_{q_6} = 0.5 L_{q_2} + 0.5 L_{q_4}$. Eliminating state q_6 would result in a 5 state automaton that is reduced—note how the choice of writing L_{q_6} as a convex combination results in two different transition structures; this illustrates that reduced automata are not unique. The reduced automaton is not minimal: for example if we fix q_5 as the initial state, we can see that there is an automaton with 2 states accepting L_{q_5}.

In the next sub-section we will provide a characterization of the reduced automaton as the image of a unique map in an Eilenberg-Moore category, and this will show a way to develop algorithms to compute this reduced automaton.

In Section 3.2 and Section 3.3 we show the following: first, we compute the image of a finitary coalgebra $(\mathcal{T}(X), \beta)$ in the final coalgebra. This utilizes a certain colimit, called the locally finite fixpoint (Definition 11). We then show if there exists an algorithm to compute a minimal base for $\mathcal{T}(X)$, we can reduce the automata. This algorithm will come into play in the choice of the maps in the colimit we compute (Equation (3)). This will relate the state space of the original automaton $\mathcal{T}(X)$ and the reduced automaton $\mathcal{T}(\overline{X})$ in the final coalgebra. Due to some of the nice properties we have that are discussed below, we are able to compute the colimit in **Set** and then revert back to $\mathbf{EM}(\mathcal{T})$.

3.1 Reduced automata as subcoalgebras of the locally finite fixpoint

Our goal here is to "factor" the morphism

$$[\![-]\!] = \mathcal{T}(X) \overset{e}{\twoheadrightarrow} I \overset{i}{\rightarrowtail} O^{A^*} \tag{1}$$

where this factorization extends to the coalgebra structure so I can be given an automaton structure. Because i is a mono, we can see that I does not contain two states that can be mapped to the same language.

We first solve the following questions: When and how can we effectively compute I and its transition structure? Can such an I be represented as an automaton in **Set** rather than in $\mathbf{EM}(\mathcal{T})$: in other words, when does there exist a set Y such that $I \cong \mathcal{T}(Y)$? This representation in **Set** will be exactly the reduced automaton as the language of a state in Y will not be a combination of other states, since i is monic.

Computing I: Epi-mono factorization in $\mathbf{EM}(\mathcal{T})$.

Deterministic automata have state spaces in **Set**. Thus, when looking at language equivalence of two states, one can easily compute the epi-mono factorization of the final map $[\![-]\!] \colon Q \to \mathbf{2}^{A^*}$. This is because quotients are well-understood in **Set**. When viewing $\mathbf{EM}(\mathcal{T})$ in general, this same notion of quotient does not translate since $\mathbf{EM}(\mathcal{T})$ may be neither abelian nor additive. For example, for $\mathcal{D}$ the finitely supported distribution monad (Definition 19), $\mathbf{EM}(\mathcal{D})$ is neither abelian nor additive. Thus actually computing the epi-mono factorization in Equation (1) is no longer as straightforward as taking a quotient, since the notion of a quotient is not yet fully understood in $\mathbf{EM}(\mathcal{T})$ (see e.g [9]). This lack of clarity on how to obtain I from computing e in Equation (1) led us to take a different perspective, made possible by the coalgebraic outlook. We will compute I using the fact that I must be a subcoalgebra of the final coalgebra. Because all $\mathcal{T}(X)$ are finitely generated state spaces, we can look at its image in the *locally-finite fixpoint* (Definition 11) of a finitary functor F over $\mathbf{EM}(\mathcal{T})$. This gets us a kind of epi-mono factorization into the final coalgebra. We will study this fixpoint below and we will show how the explicit computation of the fixpoint can be used to eliminate the redundant states.

3.2 Locally finite fixpoint

We briefly recall some facts on the the locally finite fixpoint of a functor. Below, it is assumed $\mathcal{A}$ is a locally finitely presentable (lfp) category. The goal is to calculate and use this fixpoint in determining reduction of finitary automata.

▶ **Definition 11** (Locally finite FP [13]). *The* locally finite fixpoint *of a functor* $F \colon \mathcal{A} \to \mathcal{A}$ *is given as the colimit* $(\vartheta F, \ell) = \varinjlim(\mathbf{Coalg}_{\mathsf{fg}}(F) \overset{i}{\hookrightarrow} \mathbf{Coalg}(F))$*, where* i *is the inclusion functor and* $\mathbf{Coalg}_{\mathsf{fg}}(F)$ *consists of all* F*-coalgebras with finitely generated carriers.*

▶ **Theorem 12** ([13]). *Suppose that the finitary functor* $F : \mathcal{A} \to \mathcal{A}$ *preserves monos. Then* $(\vartheta F, \ell)$ *is a fixpoint for* F*, and it is a subcoalgebra of* vF *(*vF *denotes the final* F*-coalgebra).*

To use the preceding theorem, we first show the coalgebra we are working with is a finitary functor.

▶ **Theorem 13** (Lifting to $\mathbf{EM}(\mathcal{T})$). $F : \mathbf{EM}(\mathcal{T}) \to \mathbf{EM}(\mathcal{T})$ *defined by* $F(Q) = O \times Q^A$ *is a lifting of a functor* $F_0 : \mathbf{Set} \to \mathbf{Set}$ *defined by* $F_0(X) = O \times X^A$.

Proof. By lifting of a functor, it is meant that $F_0 \circ U = U \circ F$, where $U : \mathbf{EM}(\mathcal{T}) \to \mathbf{Set}$ is the forgetful functor. $F_0 \circ U(Q, \alpha) = F_0(Q) = O \times Q^A$. In the other direction, $U \circ F(Q, \alpha) = O \times Q^A$ in **Set** since we apply F to the algebra (Q, α) and then the forgetful functor removes the algebra structure. ◀

▶ **Theorem 14.** F_0 *is a finitary functor.*

Proof. The functor F_0 can be rewritten as $O \times (A, -)$. We want to show it preserves filtered colimits. A is a finite alphabet and therefore finitely presented in **Set** so $(A, -)$ commutes with filtered colimits. Moreover, filtered colimits commute with limits in **Set**, so we have F_0 preserves filtered colimits. ◀

Therefore, F is a finitary functor [13, Remark 3.2].

Now it can be seen that the category of F-coalgebras, $\mathbf{Coalg}(F)$, we are considering in this paper satisfy all necessary assumptions. All categories considered here are of the form $\mathbf{EM}(\mathcal{T})$

for a finitary monad $\mathcal{T} : \mathbf{Set} \to \mathbf{Set}$. Eilenberg-Moore categories over a finitary monad on **Set** are algebraic categories, and thus lfp. The assumption of F preserving monos is not necessary since we showed in Theorem 13 that F is a lifting of a set functor [13]. Thus, ϑF for the finitary functor $F(Q) = O \times Q^A$ over $\mathbf{EM}(\mathcal{T})$ is a subcoalgebra of the final coalgebra. To use this, we first calculate the colimit from Definition 11: $(\vartheta F, \ell) = \varinjlim(\mathbf{Coalg}_{\mathsf{fg}}(F) \overset{i}{\hookrightarrow} \mathbf{Coalg}(F))$. We use the fact that the forgetful functor $U\colon \mathbf{Coalg}(F) \to \mathbf{EM}(\mathcal{T})$ creates colimits (this holds for any forgetful functor $U\colon \mathbf{Coalg}(F) \to \mathcal{A}$ [14]). This gives the following composition of morphisms

$$\mathbf{Coalg}_{\mathsf{fg}}(F) \overset{i}{\hookrightarrow} \mathbf{Coalg}(F) \xrightarrow{U} \mathbf{EM}(\mathcal{T})$$

allowing the colimit to be computed in $\mathbf{EM}(\mathcal{T})$. We want to apply a similar result for the forgetful functor $\bar{U} : \mathbf{EM}(\mathcal{T}) \to \mathbf{Set}$ to be able take the colimit in **Set**, like as done for DFAs, which would lead to the following chain of morphisms:

$$\mathbf{Coalg}_{\mathsf{fg}}(F) \overset{i}{\hookrightarrow} \mathbf{Coalg}(F) \xrightarrow{U} \mathbf{EM}(\mathcal{T}) \xrightarrow{\overline{U}} \mathbf{Set}$$

The forgetful functor $\bar{U} : \mathbf{EM}(\mathcal{T}) \to \mathbf{Set}$ creates colimits that are preserved by $\mathcal{T}$, and $\mathcal{T}$ preserves filtered colimits since it is finitary. Using the following result from [1] allows us to compute ϑF in **Set**.

▶ **Theorem 15** ([1]). *ϑF is a filtered colimit.*

In this paper, we offer an explicit formula for ϑF for finitary coalgebras.

$$\vartheta F = \varinjlim(\overline{U} \circ U \circ i) = \left(\bigsqcup_{((C,\alpha),\beta) \in \mathbf{Coalg}_{\mathsf{fg}}(F)} \overline{U} \circ U \circ i((C,\alpha),\beta) \right) \Big/ \sim \tag{2}$$

and hence

$$\vartheta F = \left(\bigsqcup_{(C,\alpha) \in \mathbf{EM}(\mathcal{T})_{\mathsf{fg}}} C \right) \Big/ \sim \tag{3}$$

where $\mathbf{EM}(\mathcal{T})_{\mathsf{fg}}$ denotes the subcategory of finitely generated elements in $\mathbf{EM}(\mathcal{T})$ and $\sim$ is the smallest equivalence relation on the coproduct in **Set**; i.e. the disjoint union, such that $x \sim \overline{U} \circ U \circ i(f(x))$ for all $f : (C,\alpha) \to (D,\beta)$ in $\mathbf{Coalg}_{\mathsf{fg}}(F)$ and all $x \in C$. Moreover, since $\mathbf{Coalg}_{\mathsf{fg}}(F)$ is filtered, we have that the equivalence relation is precisely

$$(x \in C) \sim (y \in D) \iff \exists E, (f : C \to E), (g : D \to E) \text{ such that } f(x) = g(y). \tag{4}$$

3.3 Fixpoints for Reduction

We now use the explicit description of ϑF (Equation (3)) and focus on the image of $\mathcal{T}(X)$ inside ϑF:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dd, "\langle \mathsf{out}, \delta \rangle"'] & \mathcal{T}(X) \arrow[rr, dashed, "\llbracket - \rrbracket"] \arrow[ddl, "\langle \mathsf{out}^\sharp, \delta^\sharp \rangle"] \arrow[dr, "q \circ i_{\mathcal{T}(X)}"'] & & O^{A^*} \arrow[dd, "\cong"] \\
& & (\bigsqcup C)\big/\sim \arrow[ur, tail] \arrow[d] & \\
O \times \mathcal{T}(X)^A \arrow[rr] & & O \times ((\bigsqcup C)\big/\sim)^A \arrow[r, tail] & O \times (O^{A^*})^A
\end{tikzcd} \tag{5}$$

Here, $i_{\mathcal{T}(X)}$ represents the inclusion of $\mathcal{T}(X)$ into the coproduct and q is the projection onto the quotient. As $\mathcal{T}(X)$ generates the state space of an automaton with set of states X, we need a notion of quotient to be able to identify redundant states, which is where we make use of the locally finite fixpoint.

We first take the image of $\mathcal{T}(X)$ into the coproduct $\coprod C$ (in **Set**). C represents the underlying set of a finitely generated element in $\mathbf{EM}(\mathcal{T})$. We now use Equation (3) to see that the image of $(\mathcal{T}(X), \langle \mathsf{out}^\sharp, \delta^\sharp\rangle)$ under $[\![-]\!]$, which we denote by $\mathsf{im}(\mathcal{T}(X))$, lies in $\coprod C/\sim$. The goal now is to reduce the set of states X by using the definition of $\sim$.

First note that any morphism in $\mathbf{EM}(\mathcal{T})$ with domain of the form $\mathcal{T}(X)$ is completely determined by where it sends X. We use the definition of $\sim$ to be consistent with that of filtered colimits since ϑF is filtered. Since we are viewing $\mathsf{im}(\mathcal{T}(X))$ inside $\vartheta(F)$, f and g in the definition of the colimit (Equation (4)) are restricted to having (co)domain $\mathcal{T}(X)$.

It is at this point where our reason for needing a base becomes necessary. In order to proceed with determining what f and g will be in Equation (4), we must calculate a base of $\mathcal{T}(X)$ in $\mathbf{EM}(\mathcal{T})$. We assume an existing algorithm for calculating $\overline{X}$, the *base* of $\mathcal{T}(X)$ (Remark 7). In other words, $\mathcal{T}(X)$ generates the same language as $\mathcal{T}(\overline{X})$. Choose the map $f : \mathcal{T}(X) \to \mathcal{T}(X)$ to be defined as $f(x) = x$ for all $x \in \overline{X}$. For $x \in X\backslash\overline{X}$, choose $y \in \mathcal{T}(X\backslash\{x\})$ such that y accepts the same language as x and define $f(x) = y$. Note that the definition of base assumes a choice may need to be made here, that is, there may be more than one $y \in \mathcal{T}(X\backslash\{x\})$ that accepts the same language as x, resulting in a reduced automaton that may not be unique.

We note that the image of f is contained within $\mathcal{T}(\overline{X})$. For the function g, we choose g to be the identity map on $\mathcal{T}(X)$. This will reduce the set X by keeping all states $x \in \overline{X}$ as f relates them to themselves, and relating redundant states in X to a combination of states in $\overline{X}$. Both f and g were designed so they can be lifted to the category $\mathbf{EM}(\mathcal{T})$.

Correctness

We show two things here. The first is that $\mathcal{T}(X) \sim \mathcal{T}(\overline{X})$ for a base $\overline{X}$, and the second is no smaller $Y \subsetneq \overline{X}$ satisfies $\mathcal{T}(X) \sim \mathcal{T}(Y)$; that is, $\overline{X}$ is a smallest such set that generates the same language.

Consider the following diagram.

$$\begin{tikzcd}
\mathcal{T}(X) \arrow[r, "f"] \arrow[d, "c"'] & \mathcal{T}(\overline{X}) \arrow[d, "\overline{c}"'] & \mathcal{T}(\overline{X}) \arrow[l, "\mathbf{1}"'] \arrow[d, "\overline{c}"] \\
F(\mathcal{T}(X)) \arrow[r, "F(f)"'] & F(\mathcal{T}(\overline{X})) & F(\mathcal{T}(\overline{X})) \arrow[l, "F(\mathbf{1})"]
\end{tikzcd} \tag{6}$$

where $\mathbf{1}$ represents the identity morphism and f is the same f as defined at the end of the previous subsection. Since $\mathcal{T}(\overline{X}) \subseteq \mathcal{T}(X)$, $\overline{c}$ is simply a restriction of c. Using that g here is the identity map, we now show the diagram is commutative.

▶ **Theorem 16.** *The diagram in Equation* (6) *is commutative.*

Proof. For the left square,

$$F(f) \circ c(x) = F(f)\langle \mathsf{out}(x), \delta_x\rangle = \langle \mathsf{out}(f(x)), \delta_{f(x)}\rangle = \overline{c}(f(x)) \tag{7}$$

For the right square,

$$F(\mathbf{1}) \circ \overline{c}(\overline{x}) = F(\mathbf{1})\langle \mathsf{out}(\overline{x}), \delta_{\overline{x}}\rangle = \langle \mathsf{out}(\overline{x}), \delta_{\overline{x}}\rangle = \overline{c} \circ \mathbf{1}(\overline{x}) \tag{8}$$

◀

We next show $\mathcal{T}(\overline{X})$ corresponds to the reduced state space; that is, there are no smaller state spaces that will generate the same language.

▶ **Theorem 17.** *If $Y \subsetneq X$, then $\mathcal{T}(Y) \not\sim \mathcal{T}(X)$.*

Proof. To show this is the *most* reduced state space, assume we have some $Y \subsetneq \overline{X}$. We will show $\mathcal{T}(Y) \not\sim \mathcal{T}(X)$.

Assume that there does exist some $Y \subsetneq \overline{X}$ such that $\mathcal{T}(Y) \sim \mathcal{T}(X)$. This would mean we have the following commutative diagram.

$$\begin{tikzcd}
\mathcal{T}(X) \arrow[r, "f"] \arrow[d, "c"'] & C \arrow[d, "e"'] & \mathcal{T}(Y) \arrow[l, "g"] \arrow[d, "d"] \\
F(\mathcal{T}(X)) \arrow[r, "F(f)"'] & F(C) & F(\mathcal{T}(\overline{X})) \arrow[l, "F(g)"]
\end{tikzcd}$$

where C is a finitely generated object in $\mathbf{EM}(\mathcal{T})$ and f and g are maps in $\mathbf{EM}(\mathcal{T})$. Since $Y \subsetneq \overline{X}$ and $\overline{X}$ is a base for the state space, this would imply there exists some $x \in \mathcal{T}(X)$ and $y \in \mathcal{T}(Y)$ such that $f(x) = g(y)$ but x and y do not generate the same language. Moreover, since Theorem 12 says ϑF is a subcoalgebra of the final coalgebra, $f(x) = g(y)$ implies $x \sim y$ and thus $[\![x]\!] = [\![y]\!]$, a contradiction. Therefore, there exists no $Y \subsetneq \overline{X}$ such that $\mathcal{T}(X) \sim \mathcal{T}(Y)$. ◀

4 Examples

In this section, we give two examples of finitary automata. We give explicit constructions of the locally finite fixpoint for each example and utilize algorithms to define the map f in each example's respective colimit.

4.1 Probabilistic Automata

Our first example of finitary automata is probabilistic automata.

▶ Remark 18. In Section 5, we will make a connection with the work done in [17]. We note that in this section, although we work over the category $\mathbf{EM}(\mathcal{D})$, the category of convex sets, everything in this section can easily be translated to $\mathbf{EM}(\mathcal{D}_s)$, the category of positive convex algebras (**PCA**). ($\mathcal{D}_s$ denotes the finitely supported sub-distribution monad).

▶ **Definition 19** (Distributions)**.** *Let $\mathcal{D} : \mathbf{Set} \to \mathbf{Set}$ denote finitely supported distributions on a set X, defined as $\mathcal{D}(X) = \{\phi : X \to [0,1] \mid \sum \phi(x) = 1\}$ where the support of each ϕ is finite. The support of a distribution is the set $\mathsf{supp}(\phi) = \{x \mid \phi(x) \neq 0\}$.*

▶ **Definition 20** (Probabilistic automaton (PA))**.** *A* probabilistic automaton *is a tuple $(Q, \langle \delta, \mathsf{out}\rangle)$, consisting of a set of states Q, a transition function $\delta\colon Q \to \mathcal{D}(Q)^A$, and an output function $\mathsf{out}\colon Q \to [0,1]$.*

The output function can be seen as a row vector (labelled by Q) with entries in $[0,1]$. δ can be equivalently represented as an A-indexed family of functions $\delta_a\colon Q \to \mathcal{D}(Q)$, each of which can be seen as square matrices (labelled by Q). Note that $\mathcal{D}(\mathbf{2}) = [0,1]$. Thus, a probabilistic automaton can be viewed as the coalgebra $F(X) = [0,1] \times X^A$ on $\mathbf{EM}(\mathcal{D})$, where $\mathcal{D}$ is clearly a finitary monad. Since $\mathbf{EM}(\mathcal{D}) \cong \mathbf{Conv}$, we use the two interchangeably for the remainder of this paper.

Reverse determinization: from Conv to Set.

If we begin with a convex set C, we want to determine when it can be written in the form $\mathcal{D}(Y)$ where Y is a finite set. We characterize this in the following theorem; the proof is a consequence of [5, Theorem 1], stating that every finitely generated convex set has a unique base, and that base is precisely the extreme points of the convex set.

▶ **Theorem 21.** *A convex set C is isomorphic to $\mathcal{D}(Y)$ as convex sets for a finite set Y if and only if C is a simplex.*

Proof. Consider a simplex C with the set of vertices Y. By definition of simplex, the Y are affinely independent, and thus, every point in C is a unique convex combination of vertices. Define a map $f : C \to \mathcal{D}(Y)$ where $y \mapsto y$ for all $y \in Y$, and such that for every $c \in C$ that is not a vertex, rewrite c as its unique convex combination of vertices. Then c is of the form $r_1 \cdot y_1 + \cdots + r_n \cdot y_n$. Define the map $r_1 \cdot y_1 + \cdots + r_n \cdot y_n \mapsto \varphi$ where $\varphi(y_i) = r_i$ for $i = 1 \cdots n$. This map is clearly convex since $r_1 \cdot f(y_1) + \cdots + r_n \cdot f(y_n) = r_1 \cdot \varphi_{y_1} + \cdots r_n \cdot \varphi_{y_n}$ where $\varphi_{y_i}(y_i) = 1$ is equal to φ defined by $\varphi(y_i) = r_i$. ◀

Therefore, given a coalgebra $C \to [0,1] \times C^A$, one can find a finite set Y and coalgebra $Y \to [0,1] \times \mathcal{D}(Y)^A$ in **Set** such that $\mathcal{D}(Y) \cong C$ (in **Conv**) if and only if C is a simplex. The set Y will be precisely the vertices, or extreme points, of the simplex C.

We revisit the diagram in Equation (5) and obtain the following diagram in **Conv**.

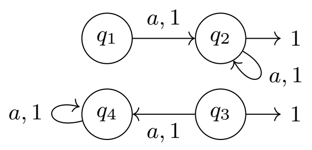
$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dd, "\langle \mathsf{out}, \delta \rangle"'] & \mathcal{D}(X) \arrow[r, dashed, two heads, "e"] \arrow[rrr, dashed, bend left, "[\![-]\!]"] \arrow[ddl, "\langle \mathsf{out}^\sharp, \delta^\sharp \rangle"] & I \cong \mathcal{D}(Y) \arrow[rr, tail, "i"] \arrow[dr, tail] \arrow[dd] & & {[0,1]^{A^*}} \arrow[dd, "\cong"'] \\
& & & \vartheta F \arrow[ur, tail] \arrow[d] & \\
{[0,1] \times \mathcal{D}(X)^A} \arrow[rr] & & {[0,1] \times I^A} \arrow[r, tail] & {[0,1] \times (\vartheta F)^A} \arrow[r, tail] & {[0,1] \times \left([0,1]^{A^*}\right)^A}
\end{tikzcd} \tag{9}$$

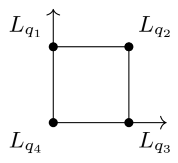
Next, we apply an algorithm developed in [20] which finds the frame of the conical hull. The full algorithm can be found in Appendix A. We also include a complete running example to show how the algorithm works in Appendix A.1. The algorithm remains the same for finding extreme points as for finding the conical hull, except a row of 1's is appended to the bottom of the matrix the algorithm utilizes. Therefore, this algorithm can also be used in the next subsection on WAs for the semiring $\mathbb{S} = \mathbb{R}^+$ (by not appending the row of 1's on the bottom of the matrix). Although we could use a span argument here to find the extreme points of a convex polytope, we use this algorithm for two reasons. The first reason is for efficiency, while the second reason is that we want to demonstrate any algorithm that exists that finds a base can be inserted into our approach.

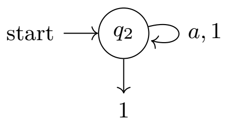
The algorithm will find the base of $\mathsf{conv}(X)$, the convex hull of the language acceptance vectors for $x \in X$. By Section 3, the image of $\mathcal{D}(X)$ in ϑF in Equation (9) will be equivalent to $\mathcal{D}(\overline{X})$, where $\overline{X}$ are the set of extreme points of $\mathsf{conv}(X)$.

▶ **Example 22** (Reduced is not unique nor minimal)**.** It should be emphasized that the reduced automaton, as described above, is not unique unless the vertices of $\mathsf{conv}(X)$ are affinely independent. If they are not affinely independent, when we remove a state $x \in X \backslash \overline{X}$, we write it as one of the (possibly many) convex combination of states in $\overline{X}$. This choice influences how we define the function $f : \mathcal{D}(X) \to \mathcal{D}(\overline{X})$ above and therefore the automaton we obtain.

Let us illustrate this through an example (see also Example 10).







The automaton on the left has state q_1 whose language L_{q_1} can be represented by the vector $(0, 1)$ in $\mathbb{R}^2$ (depicted in the middle), similarly we have q_2 as (1,1), q_3 as (1,0), and q_4 as (0,0). The convex hull of these points gives a square in $\mathbb{R}^2$, and all states are extreme points (and hence the automaton is reduced). It is immediately clear these points are not affinely independent, since $\mathbb{R}^n$ can have at most $n+1$ affinely independent points. Take for instance the center of the square $(\frac{1}{2}, \frac{1}{2})$. This can be written as both $\frac{1}{2}q_1 + \frac{1}{2}q_3$ and also as $\frac{1}{2}q_2 + \frac{1}{2}q_4$. Once a point can be written as two different convex combinations, it can be written as infinitely many different convex combinations. Thus, the choice f makes will determine what convex combination of states are used when eliminating a specific point.

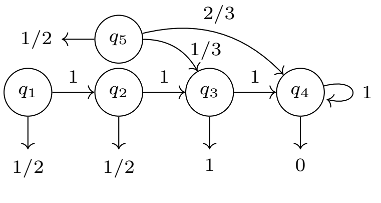
This example also illustrates the distinction between reduction and minimization. If we choose initial state q_2, a minimal automaton recognizing the language L_{q_2} is depicted on the right above. The key difference between the reduced and the minimal automaton is that in the latter *reachability* has been taken into account, whereas the reduced automaton contains all states necessary for *any* choice of initial distribution.

4.1.1 Reducing using probabilistic observation tables

We now present an algorithm for reducing probabilistic automata, providing a data structure that can be used to distinguish the different languages accepted by the states of the original automaton. We fix a finite probabilistic automaton $(Q, \langle \mathsf{out}, \delta \rangle)$. The data structure we use to reduce this automaton stems from Angluin's learning algorithm for DFAs [2]. Although our rows are labeled by states rather than words, we reuse the name *observation table* here. For readability we will use obs to denote the final coalgebra map $[\![-]\!] \colon \mathcal{D}(X) \to [0,1]^{A^*}$. For $x \in X$ we abuse notation and write $\mathsf{obs}(x)$ instead of $\mathsf{obs}(\eta(q))$.

▶ **Definition 23** (Observation table). *An* observation table *is defined by a set of words $E \subseteq A^*$. It is the function $\mathsf{row}_E \colon Q \to [0,1]^E$ defined by $\mathsf{row}_E(q)(e) = \mathsf{obs}(q)(e)$. We often identify the table by the set E.*

▶ **Example 24.** Consider the automaton on the left below. On the right is an observation table for $E = \{\varepsilon, a, aa\}$.



	ε	a	aa
q_1	1/2	1/2	1
q_2	1/2	1	0
q_3	1	0	0
q_4	0	0	0
q_5	1/2	1/3	0

The image of the row_{A^*} function clearly generates the image of obs, and from this finite set of generators we want to extract the base which will be used as states of the reduced automaton. However, this table has an infinite number of columns. It is often useful to consider convex combinations of the rows of an observation table. As such, we introduce the following definition.

▶ **Definition 25** (Probabilistic extension of row). *For each $E \subseteq A^*$, we define the* probabilistic extension of row*, written $\overline{\mathsf{row}}_E \colon \mathcal{D}(Q) \to [0,1]^E$, to be the unique homomorphism extending row_E. For $v_1, v_2 \in \mathcal{D}(Q)$, we write $v_1 \equiv_E v_2$ when $\overline{\mathsf{row}}_E(v_1)(e) = \overline{\mathsf{row}}_E(v_2)(e)$ for all $e \in E$.*

In particular, we have $\overline{\mathsf{row}}_{A^*} = \mathsf{obs}$. Observe that for each $E \subseteq A^*$, the image of $\overline{\mathsf{row}}_E$ is a convex polytope $P_E \subseteq [0,1]^E$. When $E \subseteq F \subseteq A^*$, the restriction map from $[0,1]^F$ to $[0,1]^E$ yields a surjective homomorphism from P_F to P_E. To determine which states in Q correspond to extreme points of P_{A^*}, we wish to find a finite set E such that the restriction map from P_{A^*} to P_E is an isomorphism.

▶ **Definition 26.** *Let $E \subseteq A^*$. We inductively define, for each $k \in \mathbb{N}$*

$$A^0E \overset{\text{def}}{=} E \quad \text{and} \quad A^{k+1}E \overset{\text{def}}{=} \{aw \mid a \in A, w \in A^kE\}.$$

We then define $A^*E \overset{\text{def}}{=} \bigcup_{k\in\mathbb{N}} A^kE = \{we \mid w \in A^*, e \in E\}$.

▶ **Definition 27** (Consistency)**.** *Let $E \subseteq A^*$. We say that the observation table corresponding to E is* consistent *if for all $v_1, v_2 \in \mathcal{D}(Q)$, we have $v_1 \equiv_E v_2$ implies $v_1 \equiv_{AE} v_2$.*

It can be shown that when E is consistent, the equivalence relations $\equiv_E$ and $\equiv_{A^*E}$ coincide. We focus primarily on the case where $\varepsilon \in E$.

▶ **Theorem 28.** *If $E \subseteq A^*$ is consistent and $\varepsilon \in E$, then for all $v_1, v_2 \in \mathcal{D}(Q)$ we have $v_1 \equiv_E v_2$ implies $\mathsf{obs}(v_1) = \mathsf{obs}(v_2)$.*

Proof. Suppose $v_1 \equiv_E v_2$ and $w \in A^*$ has length $k \in \mathbb{N}$. Then $w = w\varepsilon \in A^kE$. Since $v_1 \equiv_{A^kE} v_2$ (Lemma 37), we conclude $\mathsf{obs}(v_1)(w) = \overline{\mathsf{row}}_{A^kE}(v_1)(w) = \overline{\mathsf{row}}_{A^kE}(v_1)(w) = \mathsf{obs}(v_2)(w)$. ◀

▶ **Corollary 29.** *If $E \subseteq A^*$ is consistent and $\varepsilon \in E$, then the restriction map from the image of obs to the image of $\overline{\mathsf{row}}_E$ is an isomorphism.*

Using this data structure, we now shift our attention to determining whether or not an observation table with a finite number of columns is consistent. The main result is summarized in the following theorem.

▶ **Theorem 30.** *There is a polynomial-time algorithm which determines whether or not the observation table $E = \{e_1, e_2, \ldots, e_m\}$ with $m \in \mathbb{N}$ is consistent. When E is inconsistent, the algorithm produces a pair (a, i) with $a \in A$ and $1 \leq i \leq m$ such that $\mathsf{obs}(v_1)(ae_i) \neq \mathsf{obs}(v_2)(ae_i)$ for some $v_1, v_2 \in \mathcal{D}(Q)$ satisfying $v_1 \equiv_E v_2$.*

In Appendix B.1, we include further details on how to build a consistent table and a reduced automaton from it. Although we currently only have the above result for probabilistic automata, in the future, it would be interesting to study how far it generalizes for other functors and monads, as well.

4.2 $\mathbb{S}$-linear Weighted Automata

Our second example of finitary automata is $\mathbb{S}$-linear weighted automata (WA).

▶ **Definition 31** (Free Semimodule Monad)**.** *Let $\mathcal{S}_{\mathbb{S}} : \mathbf{Set} \to \mathbf{Set}$ denote finitely supported $\mathbb{S}$-combinations on a set X, defined as $\mathcal{S}_{\mathbb{S}}(X) = \{\phi : X \to \mathbb{S} \mid \sum \phi(x) \in \mathbb{S}\}$ where the support of each ϕ is finite.*

▶ **Definition 32** ($\mathbb{S}$-linear weighted automaton (WA))**.** *A* $\mathbb{S}$-linear weighted automaton *over a semiring $\mathbb{S}$ is a tuple $(Q, \langle\delta, \mathsf{out}\rangle)$, consisting of a set of states Q, a transition function $\delta\colon Q \to \mathcal{S}_{\mathbb{S}}(Q)^A$, and an output function $\mathsf{out}\colon Q \to \mathbb{S}$.*

An $\mathbb{S}$-linear weighted automaton can be viewed as the coalgebra $F(X) = \mathbb{S} \times \mathcal{S}_{\mathbb{S}}(X)^A$ on $\mathbf{EM}(\mathcal{S}_{\mathbb{S}})$, the category of $\mathbb{S}$-semimodules, where $\mathcal{S}_{\mathbb{S}}(\mathbf{1}) = \mathbb{S}$. Note $\mathcal{S}_{\mathbb{S}}$ is a finitary monad. As we alluded to earlier, our algorithm does not currently work for all semirings $\mathbb{S}$, since finding a base of certain semirings is not always feasible. We explain this in further detail below. The goal is to reduce the number of states by finding a minimal subset $\overline{X}$ such that $span(X) = span(\overline{X})$ in $\mathbf{SMod}(\mathbb{S})$, the span (over $\mathbb{S}$) of the language vectors for all $x \in X$.

WAs differ from PAs, where the convex hull has a unique base (its extreme points). Finitely generated $\mathbb{S}$-semimodules do not have this property. First, they do not necessarily have a unique base, thus even if we can find a reduced automaton, it may not be unique in its set of states; i.e. there may exist two different reduced automata with different state spaces. This part is not a problem assuming all bases have the same cardinality, because if this is the case, different reduced automata would be isomorphic. The problem we can encounter is that there may exist different bases of an $\mathbb{S}$-semimodule of different cardinality. Therefore, there are some semirings where it may not currently be feasible to find a reduced automaton (with any known algorithms), even though we may still be able to find an automaton with a smaller state space.

Reverse determinization: from SMod to Set.

▶ **Theorem 33.** *An $\mathbb{S}$-semimodule M is isomorphic to $\mathcal{S}_{\mathbb{S}}(Y)$ as semimodules for a finite set Y iff $M \cong \mathbb{S}^n$ for some $n \in \mathbb{N}$.*

Proof. $\mathcal{S}_{\mathbb{S}}(Y)$ is a free and finitely generated $\mathbb{S}$-semimodule, and all free and finitely generated semimodules are of the form $\mathbb{S}^n$. ◀

Therefore, given a coalgebra $S \to \mathbb{S} \times S^A$ for an $\mathbb{S}$-semimodule S, one can find a finite set X and coalgebra $X \to \mathbb{S} \times \mathcal{S}_s(X)^A$ in **Set** such that $\mathcal{S}_{\mathbb{S}}(X) \cong S$ (in **SMod**) if and only if $S \cong \mathbb{S}^n$ for some $n \in \mathbb{N}$.

▶ Remark 34. We must assume certain conditions on the semiring $\mathbb{S}$ in order to reduce. The first is we assume $\mathbb{S}$ is a commutative semiring. The second assumption is $k(\mathbb{S}) = 1$, where $k(\mathbb{S}) = max\{t \in \mathbb{N} \mid \text{the } \mathbb{S}\text{-semimodule } \mathbb{S} \text{ has a basis with t elements}\}$ ([18]). Examples of semirings satisfying these conditions include vector spaces, Noetherian local semirings, R_+ for a ring $R \subset \mathbb{R}$ and skew fields.

We revisit the diagram in Equation (5) and obtain the following diagram in **SMod**.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dd, "\langle \mathsf{out}, \delta \rangle"'] & \mathcal{S}_{\mathbb{S}}(X) \arrow[r, dashed, two heads, "e"] \arrow[ddl, "\langle \mathsf{out}^\sharp, \delta^\sharp \rangle"] \arrow[rrr, dashed, bend left, "\llbracket - \rrbracket"] & I \cong \mathcal{S}_{\mathbb{S}}(Y) \arrow[rr, tail, "i"] \arrow[dr, tail] \arrow[dd] & & \mathbb{S}^{A^*} \arrow[dd, "\cong"'] \\
& & & \vartheta F \arrow[ur, tail] \arrow[d] & \\
\mathbb{S} \times \mathcal{S}_{\mathbb{S}}(X)^A \arrow[rr] & & \mathbb{S} \times I^A \arrow[r, tail] & \mathbb{S} \times (\vartheta F)^A \arrow[r, tail] & \mathbb{S} \times (\mathbb{S}^{A^*})^A
\end{tikzcd} \tag{10}$$

To relate $\mathcal{S}_{\mathbb{S}}(X)$ with its reduced state space, we need to find a base of $\mathcal{S}_{\mathbb{S}}(X)$. We utilize a span approach as follows.

Choose any $x \in X$ and check if $x \in \mathbf{span}(X \backslash \{x\})$. If x is in the span, remove x from X. If x is not in the span, keep x in X. Continue in this way until we do this for all $x \in X$. Since we assume $\mathbb{S}$ is commutative and $k(\mathbb{S}) = 1$, all bases have the same cardinality and thus the order in which we check each $x \in X$ does not matter. This algorithm will find a base of $\mathbf{span}(X)$, the span of the language acceptance vectors for $x \in X$. By Section 3, the image of $\mathcal{D}(X)$ in ϑF in Equation (9) will be equivalent to $\mathcal{S}_{\mathbb{S}}(\overline{X})$, where $\overline{X}$ is the base.

▶ Remark 35. If a semiring $\mathbb{S}$ does not satisfy commutativity and $k(\mathbb{S}) \neq 1$, there is no guarantee we will obtain a reduced automata using this approach. We may still be able to obtain a smaller automata, but it may not be a reduced one. Recall from the last section that we did not utilize a span approach to find a base for probabilistic automata, so there may exist other algorithms for a semiring $\mathbb{S}$ where $k(\mathbb{S}) \geq 1$ that can find a minimal size base. Inserting such an algorithm would allow for reduction of the WA.

5 Discussion and Future Work

In this paper, we studied *reduced finitary automata*, which are finitary automata with no redundant states. Two well-known examples of finitary automata are probabilistic automata and $\mathbb{S}$-linear weighted automata for a large class of semirings $\mathbb{S}$. Reduced automata are not necessarily minimal (see e.g. Examples 10 and 22) as the reduction process does not take into account a particular initial state. Because of this, we end up with an automaton that is not minimal for a particular initial state but instead can then be used to look at different initial states – this of course comes at the cost of having extra states, yet with no redundancy.

Reduced finitary automata fit naturally in the coalgebraic framework, where it is common to ignore the initial state. The coalgebraic outlook gave us valuable guidance on the algebraic structures involved in the reduction process, justifying when we could go back to a set-based automaton. We strongly believe this approach will guide us in other generalizations, e.g to work with coalgebras on $\mathbf{EM}(\mathcal{M})$, where $\mathcal{M}$ is not over $\mathbf{Set}$.

Our work also opens a door to a potential relationship between reduction and properness. Recall that a weighted automaton for a semiring $\mathbb{S}$ is reducible using the span approach discussed in Section 4.2 iff $k(\mathbb{S}) = 1$. We conjecture that if weighted automata over the semiring $\mathbb{S}$ are reducible (using the span approach), then $\mathbb{S}$ is a proper semiring. From [17], to prove properness of some cubic functors, the authors create a zig-zag with the middle node $Z = \langle (c_{1_w}(x), c_{2_w}(y)) \rangle_{w \in A^*}$, a subsemimodule of $\mathbb{S}^{n_1} \times \mathbb{S}^{n_2}$, where $x \in \mathbb{S}^{n_1}$ and $y \in \mathbb{S}^{n_2}$ are assumed to be trace equivalent. If Z is always finitely generated for any x and y trace equivalent, it then follows that the semiring $\mathbb{S}$ is proper. We would like to show for reducible semirings $\mathbb{S}$, Z is finitely generated.

We envisage that the above problem can be tackled as follows. The accepted language of a WA over $\mathbb{S}$ can be viewed as a subset of $\mathbb{S}\langle\langle A^* \rangle\rangle$, the set of formal power series. In [7], there is a connection between residual languages and finitely generated subsemimodules of $\mathbb{S}\langle\langle A^* \rangle\rangle$, although only the semirings $\mathbb{R}, \mathbb{Q}, \mathbb{R}_+$, and $\mathbb{Q}_+$ are considered; however, we note these four semirings are known to be proper. If the subsemimodule of $\mathbb{S}\langle\langle A^* \rangle\rangle$ representing the accepted language of a WA is finitely generated, then only a finite amount of words $\{w_i\}_i \subsetneq A^*$ need to be checked to understand how the automata behaves on all of A^*. If this holds, this would then show for $\mathbb{S}$-linear weighted automata $\mathcal{A}$ and $\mathcal{B}$ where states $x \in \mathcal{A}$ and $y \in \mathcal{B}$ are trace equivalent, Z is generated by $m_1 \cdot m_2$ elements, where m_1 is the number of words that need to be checked for $\mathcal{A}$ and m_2 is the number of words that need to be checked for $\mathcal{B}$. Using results from [7, 16] we want to try to show for any $\mathbb{S}$-linear weighted automata with $k(\mathbb{S}) = 1$, the subsemimodule of $\mathbb{S}\langle\langle A^* \rangle\rangle$ representing the accepted language of the automaton is finitely generated. If this can be done, this would show that if weighted automata over $\mathbb{S}$ are reducible, then $\mathbb{S}$ is proper.

Furthermore, the idea of proper semirings was extended to proper functors in [13]. In [17], it was proven that the functor $F(X) = [0,1] \times X^A$ is a proper functor on $\mathbf{PCA}$. This leads us to one of our foremost goals: to determine whether the functor $F(X) = [0,1] \times X^A$ on $\mathbf{Conv}$ is proper. We have evidence to support this is true, but we leave this as future work.

References

1 Jiří Adámek, Stefan Milius, and Henning Urbat. On algebras with effectful iteration. In Corina Cîrstea, editor, *Proc. Coalgebraic Methods in Computer Science (CMCS'18)*, Lecture Notes Comput. Sci., 2018. To appear.

2 Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987. doi:10.1016/0890-5401(87)90052-6.

3 Michael A. Arbib and Ernest G. Manes. Adjoint machines, state-behavior machines, and duality. *Journal of Pure and Applied Algebra*, 6(3):313–344, 1975. URL: https://www.sciencedirect.com/science/article/pii/0022404975900286, doi:https://doi.org/10.1016/0022-4049(75)90028-6.

4 Fabian Birkmann, Hans-Peter Deifel, and Stefan Milius. Distributed coalgebraic partition refinement. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part II*, volume 13244 of *Lecture Notes in Computer Science*, pages 159–177. Springer, 2022. doi:10.1007/978-3-030-99527-0_9.

5 Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. Presenting convex sets of probability distributions by convex semilattices and unique bases ((co)algebraic pearls). In Fabio Gadducci and Alexandra Silva, editors, *9th Conference on Algebra and Coalgebra in Computer Science, CALCO 2021, August 31 to September 3, 2021, Salzburg, Austria*, volume 211 of *LIPIcs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CALCO.2021.11.

6 Hans-Peter Deifel, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Generic partition refinement and weighted tree automata. In Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira, editors, *Formal Methods - The Next 30 Years - Third World Congress, FM 2019, Porto, Portugal, October 7-11, 2019, Proceedings*, volume 11800 of *Lecture Notes in Computer Science*, pages 280–297. Springer, 2019. doi:10.1007/978-3-030-30942-8_18.

7 François Denis and Yann Esposito. On rational stochastic languages. *Fundam. Inf.*, 86(1,2):41–77, apr 2008.

8 Zoltan Esik and Andreas Maletti. Simulation vs. equivalence. 04 2010.

9 Joseph Gubeladze. Affine-compact functors. *Advances in Geometry*, 19:487 – 504, 2019.

10 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *J. Comput. Syst. Sci.*, 81(5):859–879, 2015. doi:10.1016/j.jcss.2014.12.005.

11 Stefan Kiefer and Björn Wachter. Stability and complexity of minimising probabilistic automata. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 2014. doi:10.1007/978-3-662-43951-7_23.

12 Barbara König and Sebastian Küpper. Generic partition refinement algorithms for coalgebras and an instantiation to weighted automata. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2014. doi:10.1007/978-3-662-44602-7_24.

13 Stefan Milius. Proper functors and their rational fixed point. *CoRR*, abs/1705.09198, 2017. URL: `http://arxiv.org/abs/1705.09198`, `arXiv:1705.09198`.

14 Stefan Milius, Dirk Pattinson, and Thorsten Wißmann. A new foundation for finitary corecursion. *CoRR*, abs/1601.01532, 2016. URL: `http://arxiv.org/abs/1601.01532`, `arXiv:1601.01532`.

15 Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991. `doi:10.1016/0890-5401(91)90052-4`.

16 Azaria Paz. Introduction to probabilistic automata. 1971.

17 Ana Sokolova and Harald Woracek. Proper semirings and proper convex functors. In *Foundations of Software Science and Computation Structure*, 2018.

18 Yi-Jia Tan. Bases in semimodules over commutative semirings. *Linear Algebra and its Applications*, 443:139–152, 2014. URL: `https://www.sciencedirect.com/science/article/pii/S0024379513007234`, `doi:https://doi.org/10.1016/j.laa.2013.11.024`.

19 Lawrence Vincent Valby. A category of polytopes, 2006.

20 Roger J. B. Wets and Christoph Witzgall. Algorithms for frames and lineality spaces of cones. *Journal of research of the national bureau of standards*, 71B, 1967.

21 Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. Efficient and modular coalgebraic partition refinement. *Log. Methods Comput. Sci.*, 16(1), 2020. `doi:10.23638/LMCS-16(1:8)2020`.

A Algorithm to compute extreme points

The following algorithm from [20] begins with a matrix A with columns as points in $\mathbb{R}^m$. It determines the frame of the conical hull of these points. The matrix A is inputted into this algorithm, and the algorithm will produce a matrix that will contain columns which are a subset of the columns of A, which give the frame of the conical hull.

Algorithm.

i (Canonical form) Use Jordan elimination for bringing A into canonical form. Every column is labeled undecided.

ii (Constant column) Select a nonbasic undecided column A_c as "constant column." If there are no such columns, go to (viii); else proceed to (iii).

iii (Pilot row) If $A_c \geq 0$, delete A_c ad return to (ii). Else select ${}_pA$ such that $a_{pc} < 0$; call it the "pilot row."

iv (Pivot column) If a_{pc} is the only negative entry in the pilot row, change the label of A_c from "undecided" to "necessary" and go to (ii). Else select a "pivot column" A_l such that $a_{pl} < 0$ and $l \neq c$.

v (Pivot row) Select a "pivot row" ${}_rA$ such that

$$0 \leq \frac{a_{rc}}{a_{rl}} \leq \min\left\{\frac{a_{ic}}{a_{il}} | a_{ic} \geq 0, a_{il} > 0\right\}$$

vi (Pivoting) Execute a simplex step (=Jordan transformation) with a_{rl} as pivot. In the absence of degeneracies, this will increase the old entry a_{pc} while keeping nonnegative entries of A_c nonnegative.

vii (Return) If the new entry a_{pc} is still negative, keep the p^{th} row as pilot row and go to (iv). Else go to (iii).

viii (Termination) If all basic columns are decided, terminate the procedure. Else select an undecided basic column A_c.

ix (Clear basis) Suppose $a_{rc} = 1$. If this is the only positive entry in ${}_rA$, then change the label of the column A_c from "undecided" to "necessary" and go to (viii). Else pivot so as to remove the undecided column A_c from the basis. Go to (iii) with A_c as constant column. (End of the algorithm).

A.1 Example Application

Take the points $\hat{A}_1 = (2, 2), \hat{A}_2 = (4, 2), \hat{A}_3 = (2, 4), \hat{A}_4 = (4, 4), \hat{A}_5 = (3, 3), \hat{A}_6 = (4, 3)$.

We perform the algorithm above on the matrix

$$A = \begin{bmatrix} 2 & 4 & 2 & 4 & 3 & 4 \\ 2 & 2 & 4 & 4 & 3 & 3 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(i) Jordan elimination brings A into canonical form.

$$A = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1/2 \\ 0 & 1 & 0 & 1 & 1/2 & 1 \\ 0 & 0 & 1 & 1 & 1/2 & 1/2 \end{bmatrix}$$

(ii) Choose A_4 non-basic undecided column. Call A_4 the "constant column".

(iii) A_4 is not non-negative, thus select row ${}_1A$ since $a_{14} < 0$ and call ${}_1A$ the "pivot row".

(iv) a_{14} is not the only negative entry in ${}_1A$ since $a_{16} < 0$. Thus, A_6 is a "pivot column".

(v) $a_{24}, a_{34} \geq 0$ and $a_{26}, a_{36} > 0$. The minimum of the set is 1, so select ${}_2A$ to be the "pivot row".

(vi) a_{26} is the pivot point. We execute a Jordan elimination step with a_{26} as the pivot. This gives

$$A = \begin{bmatrix} 1 & 1/2 & 0 & -1/2 & 1/4 & 0 \\ 0 & 1 & 0 & 1 & 1/2 & 1 \\ 0 & 1/2 & 1 & 1/2 & 1/4 & 0 \end{bmatrix}$$

(vii) a_{14} is still negative so keep the first row ${}_1A$ as the pilot row and go to (iv).

(iv) a_{14} is the only negative entry in the pilot row so change A_4 from undecided to necessary. Thus, $(4, 4)$ is a vertex. Go to (ii).

(ii) Select A_5 as "constant column". $A_5 \geq 0$ so delete A_5.

$$A = \begin{bmatrix} 1 & 0 & 0 & -1 & -1/2 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1/2 \end{bmatrix}$$

Go to (ii).

(ii). Select A_6.

(iii) A_6 is not non-negative, so select ${}_1A$ since $a_{16} < 0$ and ${}_1A$ become the "pilot row". a_{16} is not the only non-negative entry in ${}_1A$ since $a_{14} < 0$. Select "pivot column" A_4.

(v) The minimum of the set, where $l = 4, p = 1$, and $c = 6$, is $1/2$. Thus, choose "pivot row" ${}_3A$.

(vi) Executing a simplex step with a_{34} as pivot gives

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1/2 \\ 0 & 0 & 1 & 1 & 1/2 \end{bmatrix}$$

(vii) a_{16} is no longer negative. Go to (iii).
(iii) $A_6 \geq 0$ so delete A_6.

$$A = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Go to (ii).
(ii) No more undecided nonbasis columns. Go to (viii).
(viii) Select A_1.
(ix) $a_{11} = 1$ and is the only positive entry in row 1. Thus, A_1 is necessary and $(2,2)$ is a vertex.
(viii) Select A_2.
(ix) a_{22} is not the only positive entry in row 2. Pivot as to remove the undecided column A_2 from the basis.

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

where we moved A_2 to the last column to put the basis elements together.
(ii) A_2 is a "constant column".
(iii) A_2 is not non-negative so select ${}_2A$ since $a_{24} < 0$ as the "pilot row".
(iv) a_{24} is the only negative entry in row 2 so A_2 becomes necessary, and $(4,2)$ is a vertex.
(viii) Choose A_3.
(ix) $a_{33} = 1$, but is not the only positive entry in row 3 because $a_{34} > 0$. Pivot as to remove the undecided column A_3 from the basis.

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Go to (iii)
(iii) A_3 is not non-negative so select ${}_2A$ since $a_{24} < 0$ as the "pilot row".
(iv) a_{24} is the only negative entry in row 2, thus, A_3 becomes necessary, and $(2,4)$ is a vertex.

B Extra Section 4.1.1

▶ **Lemma 36.** *If $E \subseteq A^*$ is consistent, then AE is consistent.*

Proof. Let $\overline{\delta}\colon \mathcal{D}(Q) \to \mathcal{D}(Q)^A$ be the unique homomorphism extending δ. Suppose $v_1, v_2 \in \mathcal{D}(Q)$ with $v_1 \equiv_{AE} v_2$. For each $a \in A$ and $e \in E$, we have

$$\overline{\mathsf{row}}_E(\overline{\delta}(v_1)(a))(e) = \overline{\mathsf{row}}_{AE}(v_1)(ae) = \overline{\mathsf{row}}_{AE}(v_2)(ae) = \overline{\mathsf{row}}_E(\overline{\delta}(v_2)(a))(e),$$

hence $\overline{\delta}(v_1)(a) \equiv_E \overline{\delta}(v_2)(a)$. Then consistency of E yields $\overline{\delta}(v_1)(a) \equiv_{AE} \overline{\delta}(v_2)(a)$ for each $a \in A$. In particular, for $w \in AE$, we have

$$\overline{\mathsf{row}}_{A^2E}(v_1)(aw) = \overline{\mathsf{row}}_{AE}(\overline{\delta}(v_1)(a))(w) = \overline{\mathsf{row}}_{AE}(\overline{\delta}(v_2)(a))(w) = \overline{\mathsf{row}}_{A^2E}(v_2)(aw),$$

so $v_1 \equiv_{A^2E} v_2$ as needed. ◀

▶ **Lemma 37.** *If $E \subseteq A^*$ is consistent and $v_1, v_2 \in \mathcal{D}(X)$ satisfy $v_1 \equiv_E v_2$, then for each $k \in \mathbb{N}$ we have $v_1 \equiv_{A^k E} v_2$.*

Proof. Straightforward induction on k using the previous lemma. ◀

B.1 Reduction algorithm for probabilistic automata

It is convenient for computational purposes to represent observation tables and transition functions as real-valued matrices.

▶ **Definition 38.** *When $Q = \{q_1, q_2, \ldots, q_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$ with $n, m \in \mathbb{N}$, we may view the observation table corresponding to E as the real-valued n-by-m matrix $M_E \in [0,1]^{n\times m}$ given by $(M_E)_{i,j} \overset{\text{def}}{=} \mathsf{row}_E(q_i)(e_j)$ for $1 \le i \le n$ and $1 \le j \le m$. Similarly, we may view δ as the family of matrices $\{D_a\}_{a\in A}$ such that each $D_a \in [0,1]^{n\times n}$ is given by $(D_a)_{i,j} \overset{\text{def}}{=} \delta(q_i)(a)(q_j)$ for $1 \le i, j \le n$.*

▶ **Definition 39.** *For each $n \in \mathbb{N}$, we say that a vector $t \in [0,1]^n$ is* stochastic *when $\sum_{i=1}^n t_i = 1$. When $|Q| = n$, the set of stochastic vectors in $[0,1]^n$ is isomorphic to $\mathcal{D}(Q)$.*

Assume $|Q| = n$ and $|E| = m$ with $n, m \in \mathbb{N}$. Observe that the image of $\overline{\mathsf{row}}_E$ coincides with the set of row vectors $\{t^T M_E \mid t \in [0,1]^n \text{ is stochastic}\}$. Furthermore, observe that for each $a \in A$, the matrix product $D_a M_E$ is the observation table for the set of words $\{ae \mid e \in E\}$. It follows that E is inconsistent precisely when there exists $a \in A$ and stochastic vectors $t, s \in [0,1]^n$ such that $t^T M_E = s^T M_E$ but $t^T D_a M_E \neq s^T D_a M_E$. This motivates the following algorithm, which takes as input the following arguments

- An observation table $M \in \mathbb{R}^{n\times m}$
- A transition matrix $D \in \mathbb{R}^{n\times n}$

and decides whether or not the pair (M, D) is consistent; that is, whether or not for all stochastic vectors $t, s \in [0,1]^n$, we have

$$t^T M = s^T M \Rightarrow t^T DM = s^T DM. \tag{11}$$

▶ Remark 40 (Null space). If t and s were not required to be stochastic, consistency would merely amount to $\text{null}(M^T) \subseteq \text{null}(M^T D^T)$. This can be decided in polynomial time by using Gaussian elimination to find a basis for $\text{null}(M^T)$.

Consistency check for (M, D). To solve Equation (11) we use linear programming, and determine whether or not there exist $t, s \in \mathbb{R}^n$ satisfying the following linear constraints:

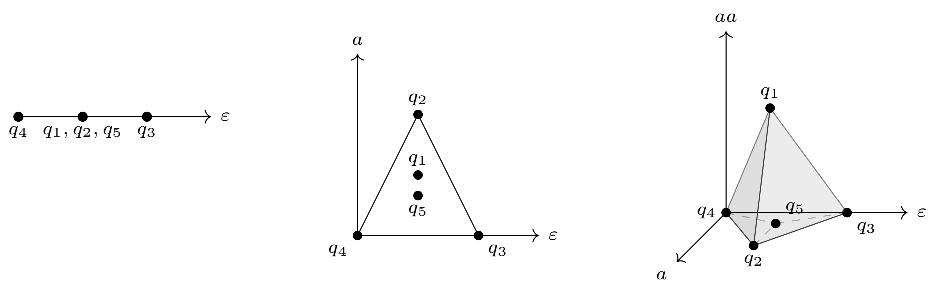
$$t_i, s_i \ge 0 \text{ for } 1 \le i \le n \qquad \sum_{i=1}^n t_i = 1 = \sum_{i=1}^n s_i \qquad \sum_{i=1}^n (t_i - s_i) M_{i,j} = 0 \text{ for } 1 \le j \le m$$

and such that $t^T DM \neq s^T DM$. The first two sets of constraints state that t and s are stochastic. The third constraint states that $(t-s)^T M = 0$. Note that $t^T DM \neq s^T DM$ precisely when, for some $1 \le j \le m$, the j'th component of $(t-s)^T DM$ is nonzero. Since t and s are interchangeable, we may assume this component is positive. For fixed j, this is captured by the linear expression

$$\sum_{i=1}^n (t_i - s_i)(DM)_{i,j} > 0.$$

For each j, we can maximize the objective function $\sum_{i=1}^n (t_i - s_i)(DM)_{i,j}$ subject to the above linear constraints by solving the corresponding linear program. By construction, (M, D) is inconsistent iff at least one of the m objective functions attains a positive value. It is clear that each program can be constructed in polynomial time. Since linear programs can be solved in polynomial time (depending on the algorithm chosen this will be in $O(n^4L)$), the consistency of (M, D) can be decided in polynomial time.

Building a consistent table. We now can build the reduced automaton by incrementally constructing E and then checking if the corresponding observation table is consistent. This leads to iteratively discovering the (state space) polytope dimension. If we consider the automaton from Example 24 we end up with 3 tables for $E = \{\varepsilon\}$, $E = \{\varepsilon, a\}$ and $E = \{\varepsilon, a, aa\}$ (see Example 24 for the last table, which is *consistent*, the other two can be read as subtables thereof by restricting the columns), which can be depicted as follows:



(12)

Building the reduced automaton from a consistent table. Given the extreme points $r_1, r_2, \ldots, r_n \in [0, 1]^E$ of the convex combinations of the rows of a consistent table, which we can calculate using the algorithm of Appendix A, we know these extreme points are included in the rows of the table, since the convex algebra is generated from those rows.

Thus, there are $q_1, q_2, \ldots q_n \in Q$ such that $\mathsf{row}(q_i) = r_i$ for $1 \leq i \leq n$. The reduced automaton can now be built as follows: Let $Q' = \{q_1, q_2, \ldots, q_n\}$ be its state space. For any state $q \in Q$ we can determine a convex combination of states in $\mathcal{D}(Q')$ accepting the same language as q, by looking at the rows corresponding to q and the states in Q'. This allows us to restrict the transition functions from Q to Q'. The output function can also be restricted to Q'.

For example, we can see that the table corresponding to the tetrahedron in Equation (12) (see also Example 24) will be consistent and from that we can obtain the reduced automaton with states $\{q_1, q_2, q_3, q_4\}$ (by computing the extreme points using the algorithm from Appendix A). The transition structure will be obtained by removing q_5 and it would replace its incoming transitions with the appropriate convex combination (the row for q_5 is a convex combination of the rows for q_2, q_3, and q_4 all weighted by $\frac{1}{3}$). Since q_5 does not have incoming transitions the resulting automaton is:

$$\begin{tikzcd}
q_1 \arrow[r, "1"] \arrow[d] & q_2 \arrow[r, "1"] \arrow[d] & q_3 \arrow[r, "1"] \arrow[d] & q_4 \arrow[loop right, "1"] \arrow[d] \\
1/2 & 1/2 & 1 & 0
\end{tikzcd}$$

Note on termination and correctness- The above iterative process will terminate because at each step we are increasing the size of the potential basis: in the worst case scenario, the automaton we started with is already reduced and we will discover that the extreme points include all the states. The guarantee of progression towards the correct observation table is captured by the following theorem:

▶ **Theorem 41** (Quotient Progress Property). *Assume there exists a surjective convex map between convex polytopes $f : C_1 \twoheadrightarrow C_2$ that is not an isomorphism. Then C_2 has strictly lower dimension than C_1.*

Proof of Theorem 41. Let $\{c_1, \dots, c_n\}$ be the unique base of C_1 ordered such that $\{c_1, \dots, c_k\}$ is an affinely-independent subset of maximal size $k \leq n$. In particular, $\{c_1, \dots, c_k\}$ is an affine basis of $\mathsf{aff}(C_1)$. Recall that f extends to a unique surjective affine map $\overline{f} : \mathsf{aff}(C_1) \to \mathsf{aff}(C_2)$. It suffices to show that $\{\overline{f}(c_1), \dots, \overline{f}(c_k)\}$ is affinely-dependent. By assumption, there are distinct points $v_1, v_2 \in C_1$ with $f(v_1) = f(v_2)$. Since $C_1 \subseteq \mathsf{aff}(C_1)$, there are distinct, uniquely-determined vectors $\alpha, \beta \in \mathbb{R}^k$ such that $\sum_{i=1}^k \alpha_i = 1 = \sum_{i=1}^k \beta_i$, $v_1 = \sum_{i=1}^k \alpha_i c_i$, and $v_2 = \sum_{i=1}^k \beta_i c_i$. Then we have

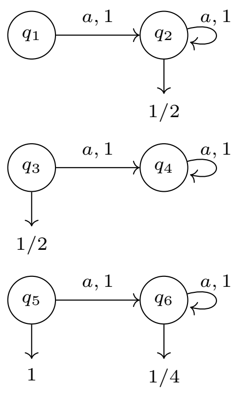
$$\sum_{i=1}^{k} \alpha_i \overline{f}(c_i) = \overline{f}\left(\sum_{i=1}^{k} \alpha_i c_i\right) = \overline{f}(v_1) = \overline{f}(v_2) = \overline{f}\left(\sum_{i=1}^{k} \beta_i c_i\right) = \sum_{i=1}^{k} \beta_i \overline{f}(c_i).$$

Thus, $\{\overline{f}(c_1), \dots, \overline{f}(c_k)\}$ is affinely-dependent as desired.

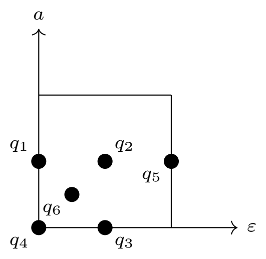
◀

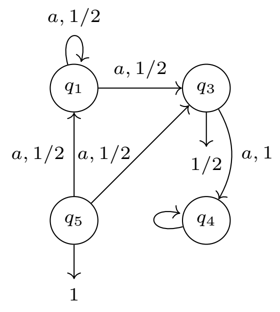
When a column is added that fixes a consistency defect, the convex algebra generated by the old table is a quotient of the convex algebra generated by the new table, where the two are not isomorphic. This means Theorem 41 applies and guarantees the new table generates a space with strictly higher dimension. Correctness of the algorithm follows trivially from termination and the characterization of the images of $\mathcal{D}(X)$ and $D(\bar{X})$ under $[\![-]\!]$ in Section 3.2.

▶ **Example 42** (Reduced automaton Example)**.** We show how the automaton from Example 10 (which we recall on the left below) can be reduced to an automaton with 4 states. In the middle is the consistent table corresponding to the automaton and how the rows in the table can be depicted to recover the convex combinations needed to obtain the reduced automaton on the right.



	ε	a
q_1	0	1/2
q_2	1/2	1/2
q_3	1/2	0
q_4	0	0
q_5	1	1/2
q_6	1/4	1/4


a
ε
q_1
q_2
q_5
q_6
q_4
q_3


$a, 1/2$
q_1
$a, 1/2$
q_3
$a, 1/2$
$a, 1/2$
$1/2$
$a, 1$
q_5
q_4
1