Algebraic Language Theory with Effects

Fabian Lenke
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Stefan Milius
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Henning Urbat
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Thorsten Wißmann
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Regular languages – the languages accepted by deterministic finite automata – are known to be precisely the languages recognized by finite monoids. This characterization is the origin of algebraic language theory. In this paper, we generalize the correspondence between automata and monoids to automata with generic computational effects given by a monad, providing the foundations of an *effectful* algebraic language theory. We show that, under suitable conditions on the monad, a language is computable by an effectful automaton precisely when it is recognizable by (1) an effectful monoid morphism into an effect-free finite monoid, and (2) a monoid morphism into a monad-monoid bialgebra whose carrier is a finitely generated algebra for the monad, the former mode of recognition being conceptually completely new. Our prime application is a novel algebraic approach to languages computed by probabilistic finite automata. Additionally, we derive new algebraic characterizations for nondeterministic probabilistic finite automata and for weighted finite automata over unrestricted semirings, generalizing previous results on weighted algebraic recognition over commutative rings.

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

Keywords and phrases Automaton, Monoid, Monad, Effect, Algebraic language theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.165

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full version with appendix*: `https://arxiv.org/abs/2410.12569` [38]

Funding Fabian Lenke and Henning Urbat are supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 470467389. Stefan Milius is supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 517924115.

1 Introduction

The algebraic approach to finite automata rests on the observation that regular languages (the languages accepted by finite automata) coincide with the languages recognized by *finite monoids*. This provides the basis for investigating complex properties of regular languages with semigroup-theoretic methods. A prime example is the celebrated result by McNaughton, Papert, and Schützenberger [42,56] that a regular language is definable in first-order logic over finite words iff its syntactic monoid (the minimal monoid recognizing the language) is aperiodic. Since the latter property is easy to verify, this implies decidability of first-order definability. Similar characterizations and decidability results are known for numerous subclasses of regular languages [25, 46, 48]. The correspondence between automata and monoids has been generalized beyond regular languages, for example to ω-regular languages [45, 66], languages of words over linear orders [10], data languages [17], tree languages [6, 12, 18, 55], cost functions [23], and weighted languages over commutative rings [52].

In this paper, we show that the equivalence between automata and monoids can be established at the more general level of automata with generic computational effects given by a monad $\mathbb{T}$ [31, 59]. This class of automata forms a common generalization of a wide range of automata models such as probabilistic automata [51], nondeterministic probabilistic automata [64], weighted automata [26], and even pushdown automata and Turing machines [31]. We introduce two modes of algebraic recognition for $\mathbb{T}$-effectful languages, both of which are natural effectful generalizations of the classical recognition by finite monoids: recognition by (1) $\mathbb{T}$-*effectful monoid morphisms* into finite effect-free monoids, and (2) ordinary monoid morphisms into finitely generated $\mathbb{T}$-algebras equipped with an additional monoid structure (plus optional compatibility conditions). We subsequently identify suitable conditions on the monad $\mathbb{T}$ ensuring that (1) and (2) capture precisely the languages computed by finite $\mathbb{T}$-automata; this yields an *effectful automata/monoid correspondence* (Theorems 4.11 and 4.22). Our results fundamentally exploit the double role played by monads in computation, namely as abstractions of both computational effects [43] and algebraic theories [40].

The investigation of algebraic recognition at the present level of generality has several benefits. First and foremost, the abstract perspective provided by generic effects naturally motivates and isolates conceptual ideas that would be easily missed for concrete instantiations of the monad $\mathbb{T}$. Notably, recognition mode (1) above is conceptually completely new, and mode (2) generalizes earlier work on algebraic recognition over commutative varieties [2] to arbitrary (non-commutative) effects. In this way, our theory leads to novel algebraic characterizations of several important automata models.

Our prime application is a characterization of languages computed by probabilistic finite automata (PFAs) [51]. PFAs extend classical deterministic finite automata by probabilistic effects, and thus serve as a natural model of state-based computations that involve uncertainty or randomization. These arise in many application domains [8, 44], as witnessed for instance by the wide range and success of probabilistic model checkers [33, 37]. On the theoretical side, PFAs share some remarkable similarities with finite automata; in particular, machine-independent characterizations of PFA-computable languages in terms of probabilistic regular expressions [19, 53] and probabilistic monadic second-order logic [65] are known. However, the fundamental algebraic perspective on finite automata has thus far withstood a probabilistic generalization. We fill this gap by establishing two different modes of *probabilistic algebraic recognition* (Theorems 3.10 and 3.18) for PFA-computable languages which instantiate the two above-mentioned modes, namely (1) recognition by finite monoids via *probabilistic monoid morphisms*; (2) recognition by *convex monoids* carried by a finitely generated convex set. The first mode emphasizes the effectful nature of probabilistic languages, while the second one relies on traditional universal algebra. These characterizations may serve as a starting point for the algebraic investigation of PFA-computable languages.

Further notable instances of the general theory include algebraic characterizations of nondeterministic probabilistic automata and weighted automata. For the latter, a weighted automata/monoid correspondence was only known for weights from a *commutative ring* [52]; our version applies to general semirings and thus captures additional types of weighted automata such as min-plus and max-plus automata [47, Ch. 5] and transducers [47, Ch. 3].

Related Work. The use of category theory to unify results for different automata models has a long tradition [5, 7, 30]. $\mathbb{T}$-automata were first studied by Goncharov et al. [31] as an instance of effectful *coalgebras* [54, 59]; they also fit into the framework of *functor automata* by Colcombet and Petrişan [24]. On the algebra side, Bojańczyk [14] used monads to unify notions of algebraic recognition. This abstract perspective on algebraic language theory has led to a series of works, including a uniform theory of Eilenberg-type correspondences [63] and an abstract account of logical definability of languages [13]. However, while each of the above works studies either automata-based or algebraic recognition individually, formal connections between both approaches are far less explored at categorical generality. The only results in this direction appear in the work of Adámek et al. [2] on the automata/monoid correspondence in commutative varieties. Our approach takes the step to non-commutative monads and on the way develops the entirely new concept of effectful language recognition.

2 Algebraic Recognition of Regular Languages

To set the scene for our algebraic approach to effectful languages, we recall the classical correspondence between finite automata and finite monoids as recognizers for regular languages [48, 50]. Let us settle some notation used in the sequel:

▶ **Notation 2.1.** Fix a finite alphabet Σ, and let Σ^* denote the set of finite words over Σ, with empty word $\varepsilon \in \Sigma^*$. We put $1 = \{*\}$ and $2 = \{\bot, \top\}$. A map $x\colon 1 \to X$ is identified with the element $x(*) \in X$, which by abuse of notation is also denoted by $x \in X$. Maps $p\colon X \to 2$ (*predicates*) are identified with subsets of X; in particular, languages are presented as predicates $L\colon \Sigma^* \to 2$. We denote the composite of two maps $f\colon X \to Y$, $g\colon Y \to Z$ by $f\,;g\colon X \to Z$ (note the order!), and the identity map on X by $\mathsf{id}_X\colon X \to X$. We use $\mapsto$ to define anonymous functions. Lastly, $X \to Y$ denotes the set of all maps from X to Y.

Finite Automata. A *deterministic finite automaton* (*DFA*) $\mathcal{A} = (Q, i, \delta, o)$ consists of a finite set Q of *states* and maps representing an *initial state*, *transitions*, and *final states*:

$$i\colon 1 \longrightarrow Q, \qquad \delta\colon Q \times \Sigma \longrightarrow Q, \qquad o\colon Q \longrightarrow 2,$$

Let $\bar\delta\colon \Sigma \to (Q \to Q)$, $\bar\delta(a) = \delta(-, a)$, denote the curried form of δ, and define the *iterated transition map* $\bar\delta^*\colon \Sigma^* \to (Q \to Q)$ and the *language* $L\colon \Sigma^* \to 2$ *computed by* $\mathcal{A}$ by

$$\bar\delta^*(\varepsilon) = \mathsf{id}_Q, \quad \bar\delta^*(wa) = \bar\delta^*(w)\,;\bar\delta^*(a) \quad \text{and} \quad L(w) = i\,;\bar\delta^*(w)\,;o \quad \text{for } w \in \Sigma^*. \tag{2.1}$$

Monoids. A *monoid* $(M, \cdot, e)$ is a set M equipped with an associative multiplication $M \times M \xrightarrow{\cdot} M$ and a neutral element $e \in M$; that is, the equations $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ and $e \cdot x = x = x \cdot e$ hold for all $x, y, z \in M$. A map $h\colon N \to M$ between monoids $(N, \cdot, n)$ and $(M, \cdot, e)$ is a *monoid morphism* if $h(n) = e$ and $h(x \cdot y) = h(x) \cdot h(y)$ for all $x, y \in N$.

▶ **Example 2.2.** **(1)** For every set X, the set $X \to X$ of endomaps forms a monoid with multiplication given by composition ; and neutral element $\mathsf{id}_X\colon X \to X$.

(2) The set Σ^* of words, with concatenation as multiplication and neutral element ε, is the *free monoid* on Σ: for every monoid $(M, \cdot, e)$ and every map $h_0\colon \Sigma \to M$, there exists a unique monoid morphism $h\colon \Sigma^* \to M$ such that $h_0(a) = h(a)$ for all $a \in \Sigma$. The morphism h, the *free extension* of h_0, is given by $h(a_1 \cdots a_n) = h_0(a_1) \cdot \cdots \cdot h_0(a_n)$ for $a_1, \ldots, a_n \in \Sigma$. For instance, the map $\bar\delta^*\colon \Sigma^* \to (Q \to Q)$ defined in (2.1) is the free extension of $\bar\delta\colon \Sigma \to (Q \to Q)$.

Monoids enable an algebraic notion of language recognition. A monoid M *recognizes* the language $L\colon \Sigma^* \to 2$ if there exists a monoid morphism $h\colon \Sigma^* \to M$ and a predicate $p\colon M \to 2$ such that $L = h\,;p$. Monoid recognition captures precisely the regular languages:

▶ **Theorem 2.3** [50, Thm. 1]**.** *For every language $L\colon \Sigma^* \to 2$, there exists a DFA computing L iff there exists a finite monoid recognizing L.*

Proof sketch. If a DFA $\mathcal{A} = (Q, i, \delta, o)$ computes L, the monoid $Q \to Q$ recognizes L via the morphism $\bar{\delta}^* \colon \Sigma^* \to (Q \to Q)$ and predicate $p \colon (Q \to Q) \to 2$ defined by $p(f) = i\,;f\,;o \in 2$.

Conversely, given a finite monoid $(M, \cdot, e)$ that recognizes L via a monoid morphism $h \colon \Sigma^* \to M$ and a predicate $p \colon M \to 2$, we can turn M into a DFA $\mathcal{A} = (M, e, \delta, p)$ computing L with transitions defined by $\delta(m, a) = m \cdot h(a)$. ◀

3 Algebraic Recognition of Probabilistic Languages

Before we present the correspondence of automata and monoids on the categorical level of general effectful automata in Section 4, we whet the reader's appetite by illustrating the important special case of probabilistic languages. These are languages computed by *probabilistic finite automata* [51], whose computational effects are finite probability distributions. All results in this section are instances of those in Section 4; however, for the convenience of the reader we provide sketches of the concrete arguments our general proofs instantiate to.

3.1 Probability Distributions and Probabilistic Channels

A *finite probability distribution* on a set X is a map $d \colon X \to [0,1]$ whose *support* $\operatorname{supp}(d) := \{x \in X \mid d(x) \neq 0\}$ is finite and which satisfies $\sum_{x \in X} d(x) = 1$. A distribution can be represented as a finite formal sum $\sum_{i \in I} r_i x_i$ where $x_i \in X$, $r_i \in [0,1]$, $\sum_{i \in I} r_i = 1$ and $d(x) = \sum_{i \in I \colon x_i = x} r_i$ for $x \in X$. The set of all distributions on X is denoted by $\mathcal{D}X$.

A map of the form $f \colon X \to \mathcal{D}Y$, denoted by $f \colon X \multimap\!\!\to Y$, is a *probabilistic channel* or *Markov kernel* from X to Y. Intuitively, f is a map from X to Y that assigns to a given input x the output y with probability $f(x)(y)$. We write $X \multimap\!\!\to Y$ for the set of all probabilistic channels from X to Y. Two probabilistic channels $f \colon X \multimap\!\!\to Y$ and $g \colon Y \multimap\!\!\to Z$ can be composed to yield a probabilistic channel $f \fatsemi g \colon X \multimap\!\!\to Z$ given by $(f \fatsemi g)(x) = \big(z \mapsto \sum_{y \in Y} f(x)(y) \cdot g(y)(z)\big)$. The *unit* at X is the probabilistic channel $\eta_X \colon X \multimap\!\!\to X$ sending $x \in X$ to the *Dirac distribution* $\delta_x \in \mathcal{D}X$ defined by $\delta_x = (y \mapsto 1 \text{ if } x = y \text{ else } 0)$. Using sum notation, composition of probabilistic channels is given by $(f \fatsemi g)(x) = \sum_{i \in I} \sum_{j \in J_i} r_i r_{ij} z_{ij}$, where $f(x) = \sum_{i \in I} r_i y_i$ and $g(y_i) = \sum_{j \in J_i} r_{ij} z_{ij}$, and the unit is $\eta_X(x) = 1x$. Composition $\fatsemi$ of probabilistic channels is associative and has η as an identity: $(f \fatsemi g) \fatsemi h = f \fatsemi (g \fatsemi h)$ and $\eta_X \fatsemi f = f \fatsemi \eta_Y$ for $f \colon X \multimap\!\!\to Y, g \colon Y \multimap\!\!\to Z$ and $h \colon Z \multimap\!\!\to W$. A channel $f \colon X \multimap\!\!\to 2$ is a *probabilistic predicate* $f \colon X \to [0,1]$ on X, where we identify $\mathcal{D}2$ with the unit interval. A map $f \colon X \to Y$ induces the *pure* channel $f\,; \eta_Y \colon X \multimap\!\!\to Y$, which we also denote by $f \colon X \to Y$ by abuse of notation.

3.2 Probabilistic Automata

Probabilistic automata, due to Rabin [51], generalize deterministic automata. Transitions are no longer given by a unique successor state $\delta(q, a)$ for every state q and input a, but a probability distribution over possible successor states. Accordingly, probabilistic automata compute *probabilistic languages*, which are simply probabilistic predicates $\Sigma^* \multimap\!\!\to 2$. Formally:

▶ **Definition 3.1.** A *probabilistic finite automaton (PFA)* $\mathcal{A} = (Q, i, \delta, o)$ consists of a finite set Q of *states* and the following channels for an *initial distribution*, the *transition distributions*, and an *acceptance predicate*, respectively:

$$i \colon 1 \multimap\!\!\longrightarrow Q, \qquad \delta \colon Q \times \Sigma \multimap\!\!\longrightarrow Q, \qquad o \colon Q \multimap\!\!\longrightarrow 2.$$

We denote by $\bar{\delta} \colon \Sigma \to (Q \multimap\!\!\to Q)$ the curried form of δ and define the *iterated transition map* $\bar{\delta}^* \colon \Sigma^* \to (Q \multimap\!\!\to Q)$ and the *language* $L \colon \Sigma^* \multimap\!\!\to 2$ *computed by* $\mathcal{A}$, respectively, by

$$\bar{\delta}^*(\varepsilon) = \eta_Q, \quad \bar{\delta}^*(wa) = \bar{\delta}^*(w) \fatsemi \bar{\delta}^*(a) \quad \text{and} \quad L(w) = i \fatsemi \bar{\delta}^*(w) \fatsemi o \quad \text{for } w \in \Sigma^*.$$

Comparing with the definition of DFAs in Section 2, we see that DFAs are precisely PFAs where i, δ, o are pure maps.

▶ **Remark 3.2.** We think of $i(q)$ as the probability that $\mathcal{A}$ starts in state q, of $\delta(q,a)(q')$ as the probability that $\mathcal{A}$ transitions from q to q' on input a, and of $o(q) \in \mathcal{D}2 \cong [0,1]$ as the probability that q is accepting. Unravelling the above definition, the language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} 2$ computed by $\mathcal{A}$ is given by the explicit formula

$$L(w) = \textstyle\sum_{\vec{q}\in Q^{n+1}} i(q_0) \cdot \big(\prod_{k=1}^{n} \delta(q_{k-1},a_k)(q_k)\big) \cdot o(q_n) \quad \text{for } w = a_1 \cdots a_n. \tag{3.1}$$

The summand for $\vec{q} \in Q^{n+1}$ is the probability that, on input w, the automaton takes the path $\vec{q}$ and accepts w. Hence, $L(w)$ is the total acceptance probability.

▶ **Remark 3.3.**

(1) Rabin's original notion of PFA [51] features an initial state and a set of final states, which amounts to restricting $i\colon 1 \mathrel{-\!\!\circ\!\!\!\to} Q$ and $o\colon Q \mathrel{-\!\!\circ\!\!\!\to} 2$ in Definition 3.1 to pure channels. Except for the behaviour on the empty word, the two versions are expressively equivalent [21, Lemmas 3.1.1 and 3.1.2].

(2) The PFA model used here is often called *reactive* in the literature, as opposed to *generative* PFAs, whose transition map is of type $Q \mathrel{-\!\!\circ\!\!\!\to} 1 + Q \times \Sigma$, computing *stochastic languages*, which are (not necessarily finite!) distributions over Σ^*. Generative PFA are not instances of $\mathbb{T}$-automata (Definition 4.3) and therefore not considered in this paper.

Our aim is to understand PFA-computable probabilistic languages in terms of recognition by algebraic structures, in the same way that finite monoids recognize regular languages. To this end, we introduce two modes of probabilistic algebraic recognition and prove that they capture precisely the PFA-computable languages.

3.3 Recognizing Probabilistic Languages by Finite Monoids

For our first mode of probabilistic algebraic recognition, we stick with finite monoids as recognizing structures, and only add probabilistic effects to the recognizing monoid morphisms. First we need some auxiliary machinery:

▶ **Definition 3.4.** *For all finite sets X and all sets Y, Z we define the maps*

$$\xi_{X,Y}\colon \mathcal{D}(X \to Y) \to (X \to \mathcal{D}Y), \qquad \xi_{X,Y}(d)(x) = \big(y \mapsto \textstyle\sum_{f\colon X\to Y, f(x)=y} d(f)\big), \tag{3.2}$$

$$\lambda_{X,Y}\colon (X \to \mathcal{D}Y) \to \mathcal{D}(X \to Y), \qquad \lambda_{X,Y}(g) = \big(f \mapsto \textstyle\prod_{x\in X} g(x)(f(x))\big), \tag{3.3}$$

$$\pi_{Y,Z}\colon \mathcal{D}Y \times \mathcal{D}Z \to \mathcal{D}(Y \times Z), \qquad \pi_{Y,Z}(d,e) = \big((y,z) \mapsto d(y)\cdot e(z)\big). \tag{3.4}$$

Intuitively, $\xi_{X,Y}(d)(x)(y)$ is the probability of picking some f according to d satisfying $f(x) = y$. For a channel $g\colon X \mathrel{-\!\!\circ\!\!\!\to} Y$, the probability of $f\colon X \to Y$ in the distribution $\lambda_{X,Y}(g)$ provides a measure of how compatible f is to g. The map $\pi_{Y,Z}$ sends two distributions on Y and Z to their *product distribution* on $Y \times Z$. Well-definedness of ξ and λ and the next lemma can be shown by calculation; conceptually, they follow from $\mathcal{D}$ being an *affine monad*.

▶ **Lemma 3.5.** *We have $\lambda_{X,Y} \mathbin{;} \xi_{X,Y} = \mathsf{id}$. In particular, the map $\xi_{X,Y}$ is surjective.*

▶ **Notation 3.6.** Given channels $f_1\colon X_1 \mathrel{-\!\!\circ\!\!\!\to} Y_1$ and $f_2\colon X_2 \mathrel{-\!\!\circ\!\!\!\to} Y_2$, we define the channel $f_1 \bar{\times} f_2\colon X_1 \times X_2 \mathrel{-\!\!\circ\!\!\!\to} Y_1 \times Y_2$ as the composite in diagram in Figure 1.

▶ **Definition 3.7.** A *probabilistic monoid morphism* from a monoid $(N, \cdot, n)$ to a monoid $(M, \cdot, e)$ is a channel $h\colon N \mathrel{-\!\!\circ\!\!\!\to} M$ making the diagram in Figure 2 commute, where the maps $\cdot$, n, e are regarded as pure channels.

$$\begin{tikzcd}
X_1 \times X_2 \arrow[r, "f_1 \times f_2"] \arrow[dr, "\circ" marking, "f_1 \bar{\times} f_2"'] & \mathcal{D}Y_1 \times \mathcal{D}Y_2 \arrow[d, "\circ" marking, "\pi_{Y_1,Y_2}"] \\
& Y_1 \times Y_2
\end{tikzcd}$$

Figure 1 Definition of $\bar{\times}$.

$$\begin{tikzcd}
N \times N \arrow[r, "\cdot"] \arrow[d, "\circ" marking, "h \bar{\times} h"'] & N \arrow[d, "\circ" marking, "h"'] & 1 \arrow[l, "n"'] \arrow[dl, "e"] \\
M \times M \arrow[r, "\cdot"] & M &
\end{tikzcd}$$

Figure 2 Probabilistic monoid morphism.

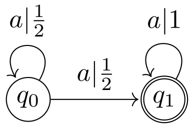

Figure 3 A PFA.

▶ **Remark 3.8.** The universal property of Σ^* extends to the probabilistic case: For every monoid $(M, \cdot, e)$ and every channel $h_0\colon \Sigma \mathrel{-\!\!\circ\!\!\!\to} M$, there exists a unique probabilistic monoid morphism $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} M$ with $h(a) = h_0(a)$ for all $a \in \Sigma$. It is given by

$$h(w) = \big(m \mapsto \textstyle\sum_{m=m_1\cdot\dots\cdot m_n} \prod_{i=1}^{n} h_0(a_i)(m_i)\big) \qquad \text{for } w = a_1 \cdots a_n \in \Sigma^*.$$

▶ **Definition 3.9.** A monoid M *probabilistically recognizes* a probabilistic language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} 2$ if there exists a probabilistic monoid morphism $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} M$ and a probabilistic predicate $p\colon M \mathrel{-\!\!\circ\!\!\!\to} 2$ such that $L = h \fatsemi p$.

We stress that, in Definition 3.9, probabilistic effects only appear in the channels h and p, while the monoid M itself is pure. At first sight, it may seem more natural to use "probabilistic monoids", with proper channels as neutral element $1 \mathrel{-\!\!\circ\!\!\!\to} M$ and multiplication $M \times M \mathrel{-\!\!\circ\!\!\!\to} M$, as recognizers. Remarkably, we need not require this additional generality:

▶ **Theorem 3.10.** *For every probabilistic language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} 2$, there exists a PFA computing L iff there exists a finite monoid probabilistically recognizing L.*

Proof sketch. Given a PFA $\mathcal{A} = (Q, \delta, i, o)$ computing L, the finite monoid $Q \to Q$ probabilistically recognizes L via the probabilistic monoid morphism $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} (Q \to Q)$ that freely extends the probabilistic channel $\bar{\delta} \fatsemi \lambda_{Q,Q}\colon \Sigma \mathrel{-\!\!\circ\!\!\!\to} (Q \to Q)$, and the probabilistic predicate $p\colon (Q \to Q) \mathrel{-\!\!\circ\!\!\!\to} 2$ defined by $p(f) = i \fatsemi f \fatsemi o \in \mathcal{D}2$. Explicitly, the maps h and p are given by

$$h(w) = \big(f \mapsto \textstyle\sum_{f=f_1;\cdots;f_n} \prod_{i=1}^{n} \prod_{q\in Q} \delta(q, a_i)(f_i(q))\big) \quad \text{and} \quad p(f) = \textstyle\sum_{q\in Q} i(q) \cdot o(f(q)),$$

for $w = a_1 \cdots a_n$. A lengthy calculation using Lemma 3.5 shows that $L = h \fatsemi p$.

Conversely, if a finite monoid $(M, \cdot, e)$ probabilistically recognizes L via $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} M$ and $p\colon M \mathrel{-\!\!\circ\!\!\!\to} 2$, then the PFA $\mathcal{A} = (M, \delta, e, p)$ computes L, with $\delta\colon M \times \Sigma \mathrel{-\!\!\circ\!\!\!\to} M$ defined by

$$\delta(m, a) = \big(n \mapsto \textstyle\sum_{m' \,:\, m\cdot m' = n} h(a)(m')\big). \qquad ◀$$

▶ **Example 3.11.** The probabilistic monoid morphism induced by the channel $h_0\colon \{a\} \mathrel{-\!\!\circ\!\!\!\to} (\{0,1\}, \vee, 0)$ sending $a \mapsto \frac{1}{2}0 + \frac{1}{2}1$, together with the pure predicate $\mathsf{id}\colon \{0,1\} \to \{0,1\}$ recognizes the language $L(a^n) = 1 - \frac{1}{2^n}$. The corresponding PFA due to Theorem 3.10 is shown in Figure 3.

3.4 Recognizing Probabilistic Languages by Convex Monoids

The presence of probabilistic effects in the recognizing morphisms places the above mode of probabilistic recognition outside standard universal algebra. In this section we develop an equivalent, purely algebraic approach based on the theory of convex sets.

A *convex set* [61] is a set X equipped with a family of binary operations $+_r\colon X \times X \to X$ ($r \in [0,1]$) subject to following equations, where $s' = r + s - rs \neq 0$ and $r' = \frac{r}{s'}$:

$$x +_r x = x, \quad x +_0 y = y, \quad x +_r y = y +_{1-r} x, \quad x +_r (y +_s z) = (x +_{r'} y) +_{s'} z$$

A map $f\colon X \to Y$ between convex sets is *affine* if $f(x +_r x') = f(x) +_r f(x')$ for $x, x' \in X$ and $r \in [0,1]$.

▶ **Example 3.12.**

(1) The prototypical convex sets are convex subsets $X \subseteq \mathbb{R}^\kappa$ (for a cardinal κ) with the operations $\vec{x} +_r \vec{y} := r \cdot \vec{x} + (1-r) \cdot \vec{y}$. Up to affine isomorphism, these are precisely the *cancellative* convex sets [61], which are those satisfying

$$x +_r y = x +_r z \implies y = z \qquad \text{for all } x, y, z \in X \text{ and } r \in (0,1).$$

(2) The set $\mathcal{D}X$ of distributions on a set X is a convex set with structure given by $d +_r e = \big(x \mapsto r \cdot d(x) + (1-r) \cdot e(x)\big)$ for $d, e \in \mathcal{D}X$. This is the *free convex set on* X: every map $h\colon X \to Y$ to a convex set Y extends uniquely to an affine map $h^{\#}\colon \mathcal{D}X \to Y$ such that $h = \eta_X \mathbin{;} h^{\#}$. Concretely, $h^{\#}$ can be defined by $h^{\#}(\sum_{i=1}^n r_i x_i) = h(x_1) +_{r_1} h^{\#}(\sum_{i=2}^n \frac{r_i}{1-r_1} x_i)$.

(3) For all sets X and Y, the set $X \mathrel{-\!\!\circ\!\!\!\to} Y$ forms a convex set with the operations $f +_r g = \big(x \mapsto f(x) +_r g(x)\big)$ for $f, g\colon X \mathrel{-\!\!\circ\!\!\!\to} Y$, $x \in X$ and $r \in [0,1]$.

Reutenauer [52] showed that finite-dimensional $\mathbb{R}$-*algebras* – real vector spaces equipped with a compatible monoid structure – precisely recognize rational power series $\Sigma^* \to \mathbb{R}$. For algebraic recognition of probabilistic languages, we generalize $\mathbb{R}$-algebras to convex monoids, which are monoids with an additional convex structure that is respected by the multiplication:

▶ **Definition 3.13.** A *convex monoid* is a convex set M equipped with a monoid structure $(M, \bullet, e)$ whose multiplication $\bullet\colon M \times M \to M$ satisfies

$$(m +_r m') \bullet n = m \bullet n +_r m' \bullet n \quad \text{and} \quad m \bullet (n +_r n') = m \bullet n +_r m \bullet n'. \tag{3.5}$$

▶ **Example 3.14.**

(1) The convex set $\mathcal{D}\Sigma^*$ with multiplication $(\sum_{i \in I} r_i v_i) \bullet (\sum_{j \in J} s_j w_j) = \sum_{i \in I,\, j \in J} r_i s_j v_i w_j$ and neutral element $\eta_{\Sigma^*}(\varepsilon) = 1\varepsilon$ is the *free convex monoid* on Σ: every map $h_0\colon \Sigma \to M$ to a convex monoid M extends to a unique affine monoid morphism $h\colon \mathcal{D}\Sigma^* \to M$ such that $h(1a) = h_0(a)$.

(2) For every set X, the convex set $X \mathrel{-\!\!\circ\!\!\!\to} X$ from Example 3.123 forms a convex monoid with channel composition $\fatsemi$ as multiplication and unit η_X as neutral element.

▶ **Definition 3.15.** A convex monoid M *recognizes* a language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} 2$ if there exists a monoid morphism $h\colon \Sigma^* \to M$ and an affine map $p\colon M \to \mathcal{D}2$ such that $L = h \mathbin{;} p$.

For a correspondence between PFA-computable languages and convex monoids, we need to impose a suitable finiteness restriction on the latter. Finite convex monoids are not sufficient; instead, we shall work with a more permissive notion of finiteness:

▶ **Definition 3.16.** A convex set X is *finitely generated* if there exists an affine surjection $s\colon \mathcal{D}G \twoheadrightarrow X$ for some finite set G. A convex monoid is *fg-carried* if its underlying convex set is finitely generated.

Intuitively, this definition says that X is the convex hull of a finite subset $s[G] \subseteq X$: every element in X is a convex combination of the elements $s(g), g \in G$.

▶ **Example 3.17.**

(1) A convex subset of $\mathbb{R}^n$ is finitely generated iff it is a *bounded convex polytope* [32], that is, it is compact and has finitely many extremal points.

(2) For all finite sets X, the convex monoid $X \mathrel{-\!\!\circ\!\!\!\to} X$ from Example 3.123 is fg-carried. This is witnessed by the map $\xi_{X,X}\colon \mathcal{D}(X \to X) \twoheadrightarrow (X \to \mathcal{D}X)$ of (3.2), which is surjective by Lemma 3.5, and affine, since it is the free extension of the map $f \mapsto f \mathbin{;} \eta_X$.

Fg-carried convex monoids give rise to our second algebraic characterization of PFAs:

▶ **Theorem 3.18.** *For every probabilistic language $L\colon \Sigma^* \mathrel{-\!\circ\!\!\to} 2$, there exists a PFA computing L iff there exists an fg-carried convex monoid recognizing L.*

Proof sketch. Given a PFA $\mathcal{A} = (Q, \delta, i, f)$ computing L, the fg-carried convex monoid $Q \mathrel{-\!\circ\!\!\to} Q$ from Example 3.172 recognizes L via the morphism $\bar{\delta}^*\colon \Sigma^* \to (Q \mathrel{-\!\circ\!\!\to} Q)$ and the affine map $p\colon (Q \mathrel{-\!\circ\!\!\to} Q) \to \mathcal{D}2$ given by $p(f) = i \fatsemi f \fatsemi o \in \mathcal{D}2$.

Conversely, suppose that $(M, \cdot, e)$ is an fg-carried convex monoid (witnessed by an affine surjection $s\colon \mathcal{D}Q \twoheadrightarrow M$) recognizing L via $h\colon \Sigma^* \to M$ and $p\colon M \to \mathcal{D}2$. Define the PFA $\mathcal{A}_Q = (Q, \delta, i, o)$ where $o = \eta_Q \mathbin{;} s \mathbin{;} p$, the transition distribution δ is chosen such that $s(\delta(g, a)) = s(g) \cdot h(a)$ for all $g \in Q$ and $a \in \Sigma$, and the initial distribution i is chosen such that $s(i) = e$; such choices exist by surjectivity of s. Then $\mathcal{A}_Q$ computes L. ◀

A remarkable property of fg-carried convex monoids is that they always admit a finite presentation, despite not necessarily being finite themselves. To see this, let us recall some terminology from universal algebra. Given an equational class $\mathcal{V}$ of algebras over a finitary signature Λ, a *finite presentation* of an algebra $A \in \mathcal{V}$ is given by (1) a finite set G of *generators*, (2) a finite set R of *relations* $s_i = t_i$ $(i = 1, \ldots, n)$ where $s_i, t_i \in T_\Lambda G$ are Λ-terms in variables from G, and (3) a surjective Λ-algebra morphism $q\colon T_\Lambda G \twoheadrightarrow A$ satisfying $q(s_i) = q(t_i)$ for all i, subject to the universal property that every morphism $h\colon T_\Lambda G \to B$, where $B \in \mathcal{V}$ and $h(s_i) = h(t_i)$ for all i, factorizes through q. In particular, we have the notions of *finitely presentable convex set* and *finitely presentable convex monoid.*

By a non-trivial result due to Sokolova and Woracek [60], finitely generated convex sets are finitely presentable (the converse holds trivially). This is the key to the following theorem; a categorical version of it is later proved in Theorem 4.17.

▶ **Theorem 3.19.** *Every fg-carried convex monoid is finitely presentable.*

Proof sketch. Let M be an fg-carried convex monoid. Choose a finite presentation (G, R, q) of M as a convex set. Since the multiplication of M is fully determined by its action on $q[G]$ by (3.5), this extends to a finite presentation of the convex monoid M by adding a relation $g \cdot g' = t_{g,g'}$ for each $g, g' \in G$, where $t_{g,g'}$ is any term in the signature of convex sets such that $q(t_{g,g'}) = q(g) \cdot q(g')$. ◀

We will see in Example 3.24 that the converse of Theorem 3.19 does not hold.

3.5 Syntactic Convex Monoids

Compared to ordinary monoids, convex monoids are fairly complex structures. The benefit of the additional complexity is the existence of *canonical* recognizers for probabilistic languages. Recall that every regular language $L\colon \Sigma^* \to 2$ has a canonical recognizer, the *syntactic monoid* $\mathrm{Syn}(L)$. It is given by the quotient $\Sigma^*/{\approx_L}$ of the free monoid Σ^* modulo the *syntactic congruence* $\approx_L \subseteq \Sigma^* \times \Sigma^*$, where $v \approx_L w$ iff $L(xvy) = L(xwy)$ for all $x, y \in \Sigma^*$. The projection $h_L\colon \Sigma^* \twoheadrightarrow \mathrm{Syn}(L)$, sending $w \in \Sigma^*$ to its congruence class $[w]$, recognizes L via $p\colon \mathrm{Syn}(L) \to 2$ with $p([w]) = 1$ iff $w \in L$. Moreover, h_L factorizes through every surjective morphism $h\colon \Sigma^* \twoheadrightarrow M$ recognizing L, thus h_L is the "smallest" surjective morphism recognizing L. For probabilistic languages, this notion generalizes as follows:

▶ **Definition 3.20.** Given a probabilistic language $L\colon \Sigma^* \mathrel{-\!\circ\!\!\to} 2$, a *syntactic convex monoid* of L is a convex monoid $\mathrm{Syn}(L)$ with a surjective affine monoid morphism $h_L\colon \mathcal{D}\Sigma^* \twoheadrightarrow \mathrm{Syn}(L)$ such that $\eta_{\Sigma^*} \mathbin{;} h_L$ recognizes L and, moreover, h_L factorizes through every surjective affine monoid morphism h such that $\eta_{\Sigma^*} \mathbin{;} h$ recognizes L:

$$\begin{tikzcd}
\Sigma^* \arrow[r, "\eta_{\Sigma^*}"] \arrow[rrr, bend left, "L"] & \mathcal{D}\Sigma^* \arrow[r, two heads, "\forall h"] \arrow[dr, two heads, "h_L"'] & M \arrow[r, "p"] \arrow[d, dashed, "\exists"'] & {[0,1]} \\
& & \mathrm{Syn}(L) \arrow[ur, "p_L"'] &
\end{tikzcd}$$

Syntactic structures for formal languages are well-studied from a categorical perspective [2, 14, 63]. The following result is an instance of [2, Thm. 3.14]:

▶ **Theorem 3.21.** *Every probabilistic language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\to} 2$ has a syntactic convex monoid, unique up to isomorphism. It is presented by generators Σ and relations given by the* syntactic congruence $\approx_L \subseteq \mathcal{D}\Sigma^* \times \mathcal{D}\Sigma^*$ *defined in Equation* (3.6)*. The maps $h_L\colon \mathcal{D}\Sigma^* \to \mathrm{Syn}(L)$ and $p_L\colon \mathrm{Syn}(L) \to [0,1]$ are given by $h_L(\sum_i r_i v_i) = [\sum_i r_i v_i]$ and $p_L([\sum_i r_i v_i]) = \sum_i r_i L(v_i)$.*

$$\textstyle\sum_i r_i v_i \approx_L \sum_j s_j w_j \quad \textit{iff} \quad \forall x, y \in \Sigma^*\colon\ \sum_i r_i L(x v_i y) = \sum_j s_j L(x w_j y) \tag{3.6}$$

Alternatively, one can construct the syntactic convex monoid as the *transition monoid* of the *minimal $\mathcal{D}$-automaton* (see the full version [38]), entailing a restriction on the convex structure of syntactic convex monoids:

▶ **Theorem 3.22.** *For every language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\to} 2$, the convex set* $\mathrm{Syn}(L)$ *is cancellative.*

Syntactic convex monoids are useful as a descriptional tool for characterizing (and potentially deciding) properties of languages in algebraic terms. Here is a simple illustration:

▶ **Example 3.23.** A probabilistic language $L\colon \Sigma^* \mathrel{-\!\!\circ\!\!\to} 2$ is *commutative* if $L(a_1 \cdots a_n) = L(a_{\pi(1)} \cdots a_{\pi(n)})$ for all $a_1, \ldots, a_n \in \Sigma$ and all permutations π of $\{1, \ldots, n\}$. One easily verifies that L is commutative iff $\mathrm{Syn}(L)$ is a commutative convex monoid.

Let us note that while every PFA-computable language is recognized by some fg-carried convex monoid (Theorem 3.18), its *syntactic* convex monoid is generally not fg-carried:

▶ **Example 3.24.** Consider the PFA from Example 3.11 computing the probabilistic language $L(a^n) = 1 - 2^{-n}$. Its syntactic convex monoid $\mathrm{Syn}(L)$ is isomorphic to the half-open interval $(0, 1] \subseteq \mathbb{R}$ with the usual convex structure and multiplication of reals; indeed, the map $i\colon \mathcal{D}\Sigma^*/{\approx_L} \to (0, 1]$ given by $[\sum_k r_k a^{n_k}] \mapsto \sum_k r_k \cdot 2^{-n_k}$ is easily seen to be an isomorphism. Since the finitely generated convex subsets of $\mathbb{R}$ are closed intervals, $\mathrm{Syn}(L)$ is not fg-carried. However, despite Theorem 3.19 not applying here, the convex monoid $(0, 1]$ can be shown to be finitely presentable, with the finite presentation given by a single generator a and a single relation $e +_{\frac{1}{3}} a \bullet a = a$. The proof is somewhat intricate; see the full version [38] for details.

▶ **Open Problem.** *Is* $\mathrm{Syn}(L)$ *finitely presentable for every PFA-computable language L?*

4 Algebraic Recognition of Effectful Languages

We now turn to the main results of our paper: two novel modes of algebraic recognition for languages computed by effectful automata. All results are parametric in the computational effect, which is modelled by a monad satisfying a suitable condition. Our results instatiate to the characterizations of PFA-computable probabilistic languages from Section 3. Other instances of our results yield new algebraic characterizations for languages recognized by weighted automata and automata that combine nondeterministic and probabilistic branching.

4.1 Monads

In the following, familiarity with basic category theory is assumed; see Mac Lane [39] for a gentle introduction. We recall some concepts from the theory of monads [40] to fix our terminology and notation. We write Set for the category of sets and functions. A *monad* $\mathbb{T} = (T, \eta, \mu)$ on Set is a triple consisting of an endofunctor $T\colon \mathsf{Set} \to \mathsf{Set}$ and two natural transformations $\eta\colon \mathsf{Id} \to T$ (the *unit*) and $\mu\colon TT \to T$ (the *multiplication*), satisfying the laws $T\mu \mathbin{;} \mu = \mu T \mathbin{;} \mu$ and $T\eta \mathbin{;} \mu = \mathsf{Id}_T = \eta T \mathbin{;} \mu$.

The *Kleisli category* $\mathcal{K}\ell(\mathbb{T})$ has sets as objects, and a morphism from X to Y, denoted $f\colon X \mathrel{-\!\!\circ\!\!\!\rightarrow} Y$, is a map $f\colon X \to TY$. The composite of $f\colon X \mathrel{-\!\!\circ\!\!\!\rightarrow} Y$ and $g\colon Y \mathrel{-\!\!\circ\!\!\!\rightarrow} Z$ is denoted by $f \fatsemi g\colon X \mathrel{-\!\!\circ\!\!\!\rightarrow} Z$ and defined by $f \fatsemi g = f \mathbin{;} Tg \mathbin{;} \mu_Z$. The identity morphism on X is the component $\eta_X\colon X \mathrel{-\!\!\circ\!\!\!\rightarrow} X$ of the unit. Intuitively, a Kleisli morphism is a function with computational effects given by the monad $\mathbb{T}$ [43]. A map $f\colon X \to Y$ is identified with the Kleisli morphism $f \mathbin{;} \eta_Y\colon X \mathrel{-\!\!\circ\!\!\!\rightarrow} Y$; such Kleisli morphisms are said to be *pure*, since they correspond to effect-free computations. Note that there is no need for operator precedence between ; and $\fatsemi$ since $(f \mathbin{;} g) \fatsemi h = f \mathbin{;} (g \fatsemi h)$ and $(g \fatsemi h) \mathbin{;} k = g \fatsemi (h \mathbin{;} k)$ for all Kleisli morphisms $g\colon X \to TY, h\colon Y \to TZ$ and pure maps $f\colon W \to X, k\colon TZ \to U$.

Further, a $\mathbb{T}$-*algebra* (A, a) consists of set A (the *carrier*) and a map $a\colon TA \to A$ (the *structure*) satisfying the *associative law* $Ta \mathbin{;} a = \mu_A \mathbin{;} a$ and the *unit law* $\eta_A \mathbin{;} a = \mathsf{id}_A$. A *morphism* from (A, a) to a $\mathbb{T}$-algebra (B, b) (a $\mathbb{T}$-*morphism* for short) is a map $h\colon A \to B$ such that $a \mathbin{;} h = Th \mathbin{;} b$. We write $\mathsf{Alg}(\mathbb{T})$ for the category of $\mathbb{T}$-algebras and $\mathbb{T}$-morphisms. Products in $\mathsf{Alg}(\mathbb{T})$ are formed in Set: the product algebra of (A_i, a_i), $i \in I$, has the structure $T(\prod_i A_i) \xrightarrow{\langle Tp_i\rangle_i} \prod_i TA_i \xrightarrow{\prod_i a_i} \prod_i A_i$, where $p_i\colon \prod_i A_i \to A_i$ is the ith product projection.

The forgetful functor from $\mathsf{Alg}(\mathbb{T})$ to Set has a left adjoint sending a set X to the *free* $\mathbb{T}$-algebra $\mathbb{T}X = (TX, \mu_X)$ over X. For every $\mathbb{T}$-algebra (A, a) and every map $h\colon X \to A$, there exists a unique $\mathbb{T}$-morphism $h^{\#}\colon \mathbb{T}X \to (A, a)$ such that $\eta_X \mathbin{;} h^{\#} = h$; that $\mathbb{T}$-morphism $h^{\#}$ is the *free extension* of h. The Kleisli category $\mathcal{K}\ell(\mathbb{T})$ is equivalent to a full subcategory of $\mathsf{Alg}(\mathbb{T})$ via the embedding $X \mapsto \mathbb{T}X$ and $h \mapsto h^{\#}$.

Monads provide a categorical view of universal algebra. Every finitary algebraic theory (Λ, E), given by a signature Λ of finitary operation symbols and a set E of equations between Λ-terms, induces a monad $\mathbb{T}$ on Set, where TX is the carrier of the free (Λ, E)-algebra (viz. the set of all Λ-terms over X modulo the equations in E), and $\eta_X\colon X \to TX$ and $\mu_X\colon TTX \to TX$ are given by inclusion of variables and flattening of terms over terms. Then $\mathbb{T}$-algebras bijectively correspond to Λ-algebras satisfying all equations in E, that is, algebras of the *variety* specified by (Λ, E). Monads $\mathbb{T}$ induced by finitary algebraic theories are precisely the *finitary* monads, that is, those preserving directed colimits [3].

Every monad $\mathbb{T}$ has a canonical *left strength*, the natural transformation

$$\mathsf{ls}_{X,Y}\colon X \times TY \to T(X \times Y) \qquad \text{defined by} \qquad \mathsf{ls}_{X,Y}(x, t) = T(y \mapsto (x, y))(t).$$

Its *right strength* $\mathsf{rs}_{X,Y}\colon TX \times Y \to T(X \times Y)$ is defined analogously. The monad $\mathbb{T}$ is *commutative* if $\mathsf{rs}_{X,TY} \fatsemi \mathsf{ls}_{X,Y} = \mathsf{ls}_{TX,Y} \fatsemi \mathsf{rs}_{X,Y}$ in $\mathcal{K}\ell(\mathbb{T})$. For every monad (be it commutative or not), we denote the left-hand composite of this equation – a *double strength* – by

$$\pi_{X,Y} := \bigl(TX \times TY \xrightarrow{\mathsf{rs}_{X,TY}} T(X \times TY) \xrightarrow{T\mathsf{ls}_{X,Y}} TT(X \times Y) \xrightarrow{\mu_{X\times Y}} T(X \times Y)\bigr). \tag{4.1}$$

Commutative finitary monads are precisely those induced by a *commutative* algebraic theory (Λ, E). This means that all operations commute with each other; for example, for all binary operations $\alpha, \beta \in \Lambda$, we have $\alpha(\beta(x_{1,2}, x_{1,2}), \beta(x_{2,1}, x_{2,2})) = \beta(\alpha(x_{1,1}, x_{2,1}), \alpha(x_{1,2}, x_{2,2}))$, and similarly for every pair of operations of other (not necessarily equal) arities.

▶ **Example 4.1.** In our applications we shall encounter the following monads:

(1) The *distribution monad* $\mathcal{D}$ sends a set X to the set $\mathcal{D}X$ of all finite probability distributions on X (Section 3.1), and a map $f\colon X \to Y$ to the map $\mathcal{D}f\colon \mathcal{D}X \to \mathcal{D}Y$ defined by $\mathcal{D}f(d) = \big(y \mapsto \sum_{x\in X,\, f(x)=y} d(x)\big)$; in sum notation, $\mathcal{D}f(\sum_i r_i x_i) = \sum_i r_i f(x_i)$. Its unit $\eta_X\colon X \to \mathcal{D}X$ is given by $\eta_X(x) = \delta_x$, and the multiplication $\mu_X\colon \mathcal{D}\mathcal{D}X \to \mathcal{D}X$ is defined by $\mu_X(e) = \big(x \mapsto \sum_{d\in\mathcal{D}X} e(d)\cdot d(x)\big)$. In sum notation, $\eta_X(x) = 1x$ and $\mu_X(\sum_{i\in I} r_i(\sum_{j\in J_i} r_{ij}x_{ij})) = \sum_{i\in I}\sum_{j\in J_i} r_i r_{ij} x_{ij}$. The Kleisli category of $\mathcal{D}$ is the category of sets and probabilistic channels, and algebras for $\mathcal{D}$ are precisely the convex sets [62]. The monad $\mathcal{D}$ is commutative; the natural transformation (4.1) is concretely given by (3.4).

(2) The *convex power set of distributions monad* $\mathcal{C}$ sends a set X to the set $\mathcal{C}X$ of non-empty, finitely generated convex subsets of $\mathcal{D}X$, and a map $f\colon X \to Y$ to $\mathcal{C}f\colon \mathcal{C}X \to \mathcal{C}Y$ defined by $\mathcal{C}f(S) = \{\mathcal{D}f(d) \mid d \in S\}$. Its unit $\eta_X\colon X \to \mathcal{C}X$ is $\eta_X(x) = \{\delta_x\}$, and the multiplication $\mu_X\colon \mathcal{C}\mathcal{C}X \to \mathcal{C}X$ is defined by

$$\mu_X(S) = \textstyle\bigcup_{\Phi\in S}\{\sum_{U\in\mathrm{supp}(\Phi)} \Phi(U)\cdot d_U \mid \forall U \in \mathrm{supp}(\Phi)\colon d_U \in U\} \qquad \text{for every } S \in \mathcal{C}\mathcal{C}X.$$

Algebras for $\mathcal{C}$ are precisely *convex semilattices* [20], which are convex sets A carrying an additional *semilattice* (i.e. a commutative idempotent semigroup) structure $(A, +)$ satisfying $(x + y) +_r z = (x +_r z) + (y +_r z)$ for $x, y, z \in A, r \in [0, 1]$. The monad $\mathcal{C}$ is not commutative.

(3) A *semiring* is a set S equipped with both the structure of a monoid $(S, \cdot, 1)$ and of a commutative monoid $(S, +, 0)$ such that multiplication distributes over addition. Every semiring S induces a monad $\mathcal{S}$ sending a set X to $\mathcal{S}X = \{f\colon X \to S \mid f(x) \neq 0 \text{ for finitely many } x \in X\}$, and a map $f\colon X \to Y$ to the map $\mathcal{S}f\colon \mathcal{S}X \to \mathcal{S}Y$ defined by $\mathcal{S}f(g) = \big(y \mapsto \sum_{f(x)=y} g(x)\big)$. The unit $\eta_X\colon X \to \mathcal{S}X$ is given by $\eta_X(x) = (y \mapsto 1 \text{ if } x = y \text{ else } 0)$, and the multiplication $\mu_X\colon \mathcal{S}\mathcal{S}X \to \mathcal{S}X$ is defined by $\mu_X(e) = \big(x \mapsto \sum_{d\in\mathcal{S}X} e(d)\cdot d(x)\big)$. Algebras for $\mathcal{S}$ correspond precisely to *S-semimodules*, that is, commutative monoids $(M, +, 0)$ with an associative scalar multiplication $S \times M \to M$ that distributes over the additive structures of S and M. The monad $\mathcal{S}$ is commutative iff the multiplication of S is commutative. The distribution monad $\mathcal{D}$ is a submonad of $\mathcal{S}$ for the semiring $S = \mathbb{R}$ of reals with the usual operations.

(4) The monad $\mathcal{M}$ sends a set X to $\mathcal{M}X = X^*$ (finite words over X), and a map $f\colon X \to Y$ to $\mathcal{M}f = f^*\colon X^* \to Y^*$ defined by $f^*(x_1 \cdots x_n) = f(x_1)\cdots f(x_n)$. The unit $\eta_X\colon X \to X^*$ is given by $\eta_X(x) = x$ and the multiplication $\mu_X\colon (X^*)^* \to X^*$ by flattening (concatenation) of words. Algebras for $\mathcal{M}$ correspond precisely to monoids.

4.2 Automata with Effects

Automata with computational effects in a monad were introduced in previous work [31, 59]. PFAs as in Definition 3.1 are the instance where the monad is $\mathcal{D}$ and the output algebra is the free $\mathcal{D}$-algebra $\mathcal{D}2$.

▶ **Assumption 4.2.** We fix a monad $\mathbb{T} = (T, \eta, \mu)$ on Set and a $\mathbb{T}$-algebra O.

▶ **Definition 4.3.** A *finite $\mathbb{T}$-automaton* ($\mathbb{T}$-*FA*) $\mathcal{A} = (Q, \delta, i, f)$ consists of a finite set Q of *states* together with two $\mathcal{K}\ell(\mathbb{T})$-morphisms and a map as shown below:

$$i\colon 1 \multimap Q, \qquad \delta\colon Q \times \Sigma \multimap Q, \qquad o\colon Q \longrightarrow O.$$

They represent an *initial state*, *transitions*, and *outputs*, respectively. We define the curried version $\bar{\delta} \colon \Sigma \to (Q \mathrel{-\!\!\circ\!\!\to} Q)$ of δ and the extended transition map $\bar{\delta}^* \colon \Sigma^* \to (Q \mathrel{-\!\!\circ\!\!\to} Q)$ just like in Definition 3.1. The language $L \colon \Sigma^* \to O$ *computed* by $\mathcal{A}$ is given by $L(w) = i \fatsemi \bar{\delta}^*(w) \,;\, o^{\#}$.

▶ **Remark 4.4.**

(1) Earlier works [31, 59] model $\mathbb{T}$-automata as coalgebras $Q \to O \times (TQ)^{\Sigma}$, and their semantics is defined via the final coalgebra for the functor $FX = O \times X^{\Sigma}$, carried by the set O^{Σ^*} of languages [54]. Definition 4.3 is equivalent to the coalgebraic one – modulo initial state and currying of transitions – yet better suited for our algebraic constructions.

(2) $\mathbb{T}$-automata and their language semantics are also instances of the framework of functor automata by Colcombet and Petrişan [24] interpreted either in the Kleisli category for $\mathbb{T}$ for free output algebras $O = \mathbb{T}O_0$, or in the Eilenberg-Moore category for $\mathbb{T}$ for general output algebras and the state object being the free algebra $\mathbb{T}Q$.

(3) Every $\mathbb{T}$-FA is equivalent to a $\mathbb{T}$-FA with a pure initial state: one may simply add a new pure initial state simulating the behaviour of a non-pure one [64, Rem. 1]. Hence, the essence of the effectful nature of $\mathbb{T}$-FAs lies in the transitions.

▶ **Example 4.5.**

(1) $\mathcal{D}$-FAs with the convex output set $O = \mathcal{D}2$ are precisely PFAs.

(2) $\mathcal{C}$-FAs are *nondeterministic probabilistic finite automata* (*NPFAs*) [64]. They combine nondeterministic and probabilistic branching and as such are closely related to Segala systems [58] and Markov decision processes [47, Ch. 36]. We take the output convex semilattice $O = [0,1]_{\max}$ given by the interval $[0,1] \subseteq \mathbb{R}$ with its usual convex structure and taking maxima as the semilattice operation. An NPFA $\mathcal{A} = (Q, \delta, i, o)$ consists of a finite state set Q and maps $i \colon 1 \to \mathcal{C}Q$, $\delta \colon Q \times \Sigma \to \mathcal{C}Q$, and $o \colon Q \to [0,1]$. If $\mathcal{A}$ is in state q and receives the input letter a, then it chooses a distribution $d \in \delta(q,a)$ and transitions to the state q' with the probability $d(q')$. The choices are made with the goal of maximizing the acceptance probability of the input. Formally, $\mathcal{A}$ computes the probabilistic language $L_{\max} \colon \Sigma^* \to [0,1]$ defined for $w = a_1 \cdots a_n$ by

$$w \;\mapsto\; \max\{\textstyle\sum_{\vec{q} \in Q^{n+1}} d_0(q_0) \cdot (\prod_{k=1}^{n} d_{q_{k-1},k}(q_k)) \cdot o(q_n) \mid d_0 \in i, \forall q. \forall k. d_{q,k} \in \delta(q, a_k)\}.$$

Under this semantics, NPFAs are more expressive than PFAs [64]. Two alternative semantics emerge by modifying the output convex semilattice: for $O = [0,1]_{\mathsf{min}}$, the interval $[0,1]$ with the minimum operation, an NPFA computes the language $L_{\min} \colon \Sigma^* \to [0,1]$ of minimal acceptance probabilities. For $O = \mathcal{C}2$, the convex semilattice of closed subintervals of $[0,1]$, it computes the language $L_{\mathrm{int}} \colon \Sigma^* \to \mathcal{C}2$ sending $w \in \Sigma^*$ to the interval $[L_{\min}(w), L_{\max}(w)]$. See van Heerdt et al. [64] for a detailed coalgebraic account of the different semantics.

(3) $\mathcal{S}$-FAs with the output semimodule $O = \mathcal{S}1 \cong S$, are precisely *weighted finite automata* (*WFAs*) [26] over the semiring S. A WFA computes a *weighted language* (or *formal power series*) $L \colon \Sigma^* \to S$, which is given by the formula (3.1), with sums and products formed in S. Interesting choices for the semiring S include:

   (1) the semiring $\mathbb{R}$ of reals – note that PFAs are a special case of WFAs over $\mathbb{R}$;

   (2) the *min-plus* or *max-plus semiring* of natural numbers with the operation min (resp. max) as addition and the operation + as multiplication; WFAs then correspond to *min-plus* and *max-plus automata* computing shortest (resp. longest) paths [47, Ch. 5].

   (3) the semiring of regular languages over an alphabet Γ w.r.t. union and concatenation; WFAs are equivalent to *transducers* computing relations $R \subseteq \Sigma^* \times \Gamma^*$ [47, Ch. 3].

4.3 Recognizing Effectful Languages by Bialgebras

We now introduce two modes of algebraic recognition for languages computed by $\mathbb{T}$-automata. In contrast to the exposition in Section 3 for the instance $\mathbb{T} = \mathcal{D}$, we swap the order of presentation and first consider the recognition by algebraic structures and subsequently the recognition by effectful homomorphisms in Section 4.4, since from the categorical viewpoint the latter is best understood in the context of the former.

▶ **Definition 4.6.** A $(\mathbb{T}, \mathcal{M})$-*bialgebra* is a set M equipped with both a $\mathbb{T}$-algebra structure $a\colon TM \to M$ and the structure of a monoid $(M, \bullet, e)$. It is a $\mathbb{T}$-*monoid* if the monoid multiplication $M \times M \overset{\cdot}{\to} M$ is a $\mathbb{T}$-*bimorphism*: for every $m \in M$ the maps $m \bullet (-), (-) \bullet m\colon M \to M$ are $\mathbb{T}$-endomorphisms on (M, a).

▶ **Remark 4.7.** Both $(\mathbb{T}, \mathcal{M})$-bialgebras and $\mathbb{T}$-monoids admit a categorical view:

(1) $(\mathbb{T}, \mathcal{M})$-bialgebras correspond to algebras for the coproduct [1] of the monads $\mathbb{T}$ and $\mathcal{M}$.

(2) If $\mathbb{T}$ is commutative, then $\mathbb{T}$-monoids correspond to algebras for the composite monad $\mathbb{T}\mathcal{M}$ induced by the canonical distributive law of $\mathcal{M}$ over $\mathbb{T}$ [41, Thm 4.3.4]. They also correspond to monoid objects in the closed monoidal category $(\mathsf{Alg}(\mathbb{T}), \otimes, \mathbb{T}1)$ whose tensor product represents $\mathbb{T}$-bimorphisms (i.e. $\mathbb{T}$-morphisms $A \otimes B \to C$ correspond to $\mathbb{T}$-bimorphisms $A \times B \to C$) [9,36,57]. The internal hom of $A, B \in \mathsf{Alg}(\mathbb{T})$ is the algebra $[A, B]$ of all $\mathbb{T}$-morphisms from A to B, viewed as a subalgebra of the product $B^{|A|}$. In particular, composition $[A, B] \times [B, C] \overset{\cdot}{\to} [A, C]$ is a $\mathbb{T}$-bimorphism, so $([A, A], ;, \mathsf{id}_A)$ is a $\mathbb{T}$-monoid.

▶ **Example 4.8.**

(1) A $(\mathcal{D}, \mathcal{M})$-bialgebra is a convex set carrying an additional monoid structure (without any interaction between the two structures). A $\mathcal{D}$-monoid is a convex monoid, that is, a $(\mathcal{D}, \mathcal{M})$-bialgebra whose monoid multiplication satisfies (3.5).

(2) For every set X, the set $X \mathrel{-\!\!\circ\!\!\!\rightarrow} X$ is a $(\mathbb{T}, \mathcal{M})$-bialgebra with monoid structure given by Kleisli composition and $\mathbb{T}$-algebra structure given by the product $(\mathbb{T}X)^X$ of the free $\mathbb{T}$-algebra $\mathbb{T}X$. If $\mathbb{T}$ is commutative, then $X \mathrel{-\!\!\circ\!\!\!\rightarrow} X$ is a $\mathbb{T}$-monoid; in fact, it is isomorphic to the $\mathbb{T}$-monoid $[\mathbb{T}X, \mathbb{T}X]$ (Remark 4.72).

Both the notion of recognition of probabilistic languages by convex monoids and the respective finiteness condition (Definitions 3.15 and 3.16) are instances of the following:

▶ **Definition 4.9.** A $(\mathbb{T}, \mathcal{M})$-bialgebra M *recognizes* the language $L\colon \Sigma^* \to O$ if $L = h\,;p$ for some monoid morphism $h\colon \Sigma^* \to M$ and some $\mathbb{T}$-morphism $p\colon M \to O$.

▶ **Definition 4.10.** A $\mathbb{T}$-algebra (A, a) is *finitely generated* if there exists a surjective $\mathbb{T}$-morphism $\mathbb{T}G \twoheadrightarrow (A, a)$ for some finite set G. A $(\mathbb{T}, \mathcal{M})$-bialgebra is *fg-carried* if its underlying $\mathbb{T}$-algebra is finitely generated.

▶ **Theorem 4.11.** *Suppose that $X \mathrel{-\!\!\circ\!\!\!\rightarrow} X$ is a finitely generated $\mathbb{T}$-algebra for every finite set X. Then for every language $L\colon \Sigma^* \to O$, the following are equivalent:*

(1) *There exists a $\mathbb{T}$-FA computing L.*

(2) *There exists an fg-carried $(\mathbb{T}, \mathcal{M})$-bialgebra recognizing L.*

If the monad $\mathbb{T}$ is commutative, then these statements are equivalent to:

(3) *There exists an fg-carried $\mathbb{T}$-monoid recognizing L.*

▶ **Remark 4.12.** The condition of the theorem is equivalent to finitely generated $\mathbb{T}$-algebras being closed under finite products.

$\mathbb{T}$	$\mathbb{T}$-FA	M	$\xi_0\colon M \to (X \mathrel{-\!\circ\!\!\!\rightarrow} X)$
$\mathcal{D}$	PFA	$X \to X$	$\xi_0(f) = \big(x \mapsto \eta_X(f(x))\big)$
$\mathcal{C}$	NPFA	$X \to X$	$\xi_0(f) = \big(x \mapsto \eta_X(f(x))\big)$
$\mathcal{S}$	WFA	$X \rightharpoonup X$	$\xi_0(f) = \big(x \mapsto \eta_X(f(x))$ if $f(x)$ is defined, else $0 \in \mathcal{S}X\ \big)$

Figure 4 Witnesses for $X \mathrel{-\!\circ\!\!\!\rightarrow} X$ being (monoidally) finitely generated.

Proof sketch. (1)⇒(2) Let $\mathcal{A} = (Q, \delta, i, o)$ be a $\mathbb{T}$-FA computing L. Then L is recognized by the fg-carried $(\mathbb{T}, \mathcal{M})$-bialgebra $Q \mathrel{-\!\circ\!\!\!\rightarrow} Q$ via the monoid morphism $\bar{\delta}^*\colon \Sigma^* \to (Q \mathrel{-\!\circ\!\!\!\rightarrow} Q)$ and the $\mathbb{T}$-morphism $p\colon (Q \mathrel{-\!\circ\!\!\!\rightarrow} Q) \to O$ which is given by $p(f) = i \mathbin{⨟} f \mathbin{;} o^{\#}$. The proof that p is a $\mathbb{T}$-morphism requires the initial state i to be pure, which we may assume due to Remark 4.4.

(2)⇒(1) Let M be an fg-carried $(\mathbb{T}, \mathcal{M})$-bialgebra recognizing L via the monoid morphism $h\colon \Sigma^* \to M$ and the $\mathbb{T}$-morphism $p\colon M \to O$. Since M is finitely generated as a $\mathbb{T}$-algebra, there exists a surjective $\mathbb{T}$-morphism $s\colon \mathbb{T}Q \twoheadrightarrow M$ for some finite set Q. Let $(M, \bullet, e)$ denote the monoid structure, and let $s_0\colon Q \to M$ and $h_0\colon \Sigma \to M$ be the domain restrictions of s and h. We construct a $\mathbb{T}$-FA $\mathcal{A} = (Q, \delta, i, o)$ computing L as follows: we choose $i\colon 1 \mathrel{-\!\circ\!\!\!\rightarrow} Q$ and $\delta\colon Q \times \Sigma \mathrel{-\!\circ\!\!\!\rightarrow} Q$ such that the first two diagrams below commute – these choices exist because s is surjective. Moreover, we define $o\colon Q \to O$ by $o := s_0 \mathbin{;} p$ as in the third diagram.

$$\begin{tikzcd}
1 \arrow[r, "i"] \arrow[dr, "e"'] & TQ \arrow[d, two heads, "s"] \\
& M
\end{tikzcd}
\qquad
\begin{tikzcd}
Q \times \Sigma \arrow[rr, "\delta"] \arrow[d, "s_0 \times \mathsf{id}"'] & & TQ \arrow[d, two heads, "s"] \\
M \times \Sigma \arrow[r, "\mathsf{id} \times h_0"] & M \times M \arrow[r, "\bullet"] & M
\end{tikzcd}
\qquad
\begin{tikzcd}
Q \arrow[r, "o"] \arrow[d, "s_0"'] & O \\
M \arrow[ur, "p"'] &
\end{tikzcd}$$

One can prove that the $\mathbb{T}$-FA $\mathcal{A}$ computes the language L.

Now suppose that the monad $\mathbb{T}$ is commutative. Then (1)⇒(3) is shown like (1)⇒(2), adding the observation that the recognizing $(\mathbb{T}, \mathcal{M})$-bialgebra $Q \mathrel{-\!\circ\!\!\!\rightarrow} Q$ is a $\mathbb{T}$-monoid as in Example 4.82. The implication (3)⇒(2) is trivial. ◀

▶ **Example 4.13.** The condition of Theorem 4.11 holds for $\mathbb{T} \in \{\mathcal{D}, \mathcal{C}, \mathcal{S}\}$ from Example 4.1. We present for each finite set X a finite set M and a map $\xi_0\colon M \to (X \mathrel{-\!\circ\!\!\!\rightarrow} X)$ such that $\xi = \xi_0^{\#}\colon TM \twoheadrightarrow (X \mathrel{-\!\circ\!\!\!\rightarrow} X)$ is surjective. The respective witnesses are given in Figure 4, where $X \rightharpoonup X$ denotes the set of all partial functions on X. For all three monads, the condition amounts to the observation that every effectful function $f\colon X \mathrel{-\!\circ\!\!\!\rightarrow} X$ can be built from (partial or total) *effect-free* functions using $\mathbb{T}$-operations.

▶ **Remark 4.14.** Using the abstract theory of syntactic structures [2, 14, 63], it is possible to associate a canonical algebraic recognizer to every language $L\colon \Sigma^* \to O$, namely its *syntactic* $(\mathbb{T}, \mathcal{M})$*-bialgebra* and, in the commutative case, its *syntactic* $\mathbb{T}$*-monoid.* The syntactic convex monoid (Theorem 3.21) is an instance of the latter.

Next, we show that, under conditions on $\mathbb{T}$, fg-carried $\mathbb{T}$-monoids are finitely presentable.

▶ **Definition 4.15.** Let $\mathbb{S}$ be a monad. An $\mathbb{S}$-algebra is *finitely presentable* if it is the coequalizer in $\mathsf{Alg}(\mathbb{S})$ of some pair $p, q\colon \mathbb{S}X \to \mathbb{S}Y$ of $\mathbb{S}$-morphisms for finite sets X and Y.

▶ **Remark 4.16.** For the monad $\mathbb{S}$ associated to an algebraic theory (Λ, E), an $\mathbb{S}$-algebra is finitely presentable iff its corresponding Λ-algebra in the variety defined by (Λ, E) admits a finite presentation by generators and relations [4, Prop. 11.28]. Instantiating $\mathbb{S}$ to the monad corresponding to the algebraic theory of $\mathbb{T}$-monoids, where $\mathbb{T}$ is finitary, we obtain a notion of *finitely presentable* $\mathbb{T}$*-monoid.*

▶ **Theorem 4.17.** *If $\mathbb{T}$ is finitary and commutative, and finitely generated $\mathbb{T}$-algebras are finitely presentable, then every fg-carried $\mathbb{T}$-monoid is finitely presentable.*

Note that Theorem 3.19 is the instance of this result for $\mathbb{T} = \mathcal{D}$.

4.4 Recognizing Effectful Languages by Finite Monoids

Our second mode of recognition of $\mathbb{T}$-FA-computable languages uses effectful monoid morphisms to finite monoids; recognition by probabilistic channels from Section 3.3 is an instance.

▶ **Notation 4.18.** Given $\mathcal{K}\ell(\mathbb{T})$-morphisms $f_i\colon X_i \mathrel{-\!\!\circ\!\!\!\to} Y_i$ $(i = 1, 2)$, we define the Kleisli morphism $f_1 \bar{\times} f_2\colon X_1 \times X_2 \mathrel{-\!\!\circ\!\!\!\to} Y_1 \times Y_2$ as in Notation 3.6, with $\mathbb{T}$ in lieu of $\mathcal{D}$.

▶ **Definition 4.19.** A ($\mathbb{T}$-)*effectful monoid morphism* from a monoid $(N, \cdot, n)$ to a monoid $(M, \cdot, e)$ is a $\mathcal{K}\ell(\mathbb{T})$-morphism $h\colon N \mathrel{-\!\!\circ\!\!\!\to} M$ such that the diagram in Figure 2 commutes. The monoid M *effectfully recognizes* the language $L\colon \Sigma^* \to O$ if there exists an effectful monoid morphism $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} M$ and a map $p\colon M \to O$ such that $L = h\,;p^{\#}$.

The key condition on the monad $\mathbb{T}$ that makes effectful recognition work is isolated in Theorem 4.22. This requires some technical preparation:

▶ **Remark 4.20.** The natural transformations $\pi_{X,Y}\colon TX \times TY \to T(X \times Y)$ of (4.1) and $\eta_X\colon X \to TX$ make T a *monoidal functor*, that is, π and η satisfy coherence laws w.r.t. the natural isomorphisms $(X \times Y) \times Z \cong X \times (Y \times Z)$ and $1 \times X \cong X \cong X \times 1$ [34, Thm. 2.1]. Consequently, T preserves monoid structures: for every monoid $(M, \cdot, e)$, the set TM forms a monoid with neutral element and multiplication, respectively, defined by

$$1 \xrightarrow{e} M \xrightarrow{\eta_M} TM \qquad \text{and} \qquad TM \times TM \xrightarrow{\pi_{M,M}} T(M \times M) \xrightarrow{T\cdot} TM.$$

It is folklore that if $\mathbb{T}$ is commutative, then, for every $\mathbb{T}$-monoid N, the extension $h^{\#}\colon TM \to N$ of every monoid morphism $h\colon M \to N$ is also a monoid morphism. For non-commutative monads $\mathbb{T}$, this is not true in general.

▶ **Definition 4.21.** A $(\mathbb{T}, \mathcal{M})$-bialgebra N is *monoidally finitely generated* if there exists a finite monoid M and a monoid morphism $\xi_0\colon M \to N$ whose free extension $\xi = \xi_0^{\#}\colon TM \to N$ is a surjective monoid morphism.

We are now ready to state the desired general result on effectful recognition. Note that its condition implies that of Theorem 4.11; for a separating example see the full version [38].

▶ **Theorem 4.22.** *Suppose that for every finite set X the $(\mathbb{T}, \mathcal{M})$-bialgebra $X \mathrel{-\!\!\circ\!\!\!\to} X$ from Example 4.82 is monoidally finitely generated. Then for every language $L\colon \Sigma^* \to O$, there exists a $\mathbb{T}$-FA computing L iff there exists a finite monoid $\mathbb{T}$-effectfully recognizing L.*

Proof sketch. $(\Rightarrow)$ Suppose that $\mathcal{A} = (Q, \delta, i, f)$ is a $\mathbb{T}$-FA computing L. The proof of Theorem 4.11 shows that L is recognized by the $(\mathbb{T}, \mathcal{M})$-bialgebra $Q \mathrel{-\!\!\circ\!\!\!\to} Q$; say $L = g\,; p$ for some monoid morphism $g\colon \Sigma^* \to (Q \mathrel{-\!\!\circ\!\!\!\to} Q)$ and some $\mathbb{T}$-morphism $p\colon (Q \mathrel{-\!\!\circ\!\!\!\to} Q) \to O$. (The specific choices of g and p in that proof are not relevant here.) Since $Q \mathrel{-\!\!\circ\!\!\!\to} Q$ is monoidally finitely generated, there exists a monoid morphism $\xi_0\colon M \to (Q \mathrel{-\!\!\circ\!\!\!\to} Q)$ such that M is finite and $\xi = \xi_0^{\#}\colon TM \twoheadrightarrow (Q \mathrel{-\!\!\circ\!\!\!\to} Q)$ is a surjective monoid morphism. By the universal property of the free monoid Σ^* and the surjectivity of ξ, we obtain a (not necessarily unique) monoid morphism $h\colon \Sigma^* \to TM$ with $g = h\,; \xi$ to fill the commutative diagram (4.2). Moreover, one can show that $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\!\to} M$ is an *effectful* monoid morphism. Then the finite monoid M effectfully recognizes L via h and $\xi_0 \,;\, p$ because $L = g \,;\, p = h \,;\, (\xi_0 \,;\, p)^{\#}$.

$$\begin{tikzcd}
\Sigma^* \arrow[r, dashed, "h"] \arrow[dr, "g"'] & TM \arrow[r, "(\xi_0;p)^{\pi}"] \arrow[d, two heads, "\xi"] & O \\
& Q \mathrel{-\!\!\circ\!\!\to} Q \arrow[ur, "p"'] &
\end{tikzcd} \qquad (4.2)
\qquad
\begin{tikzcd}
M \times \Sigma \arrow[r, "\eta_M \times h_0"] \arrow[d, "\circ" marking, "\delta"] & TM \times TM \arrow[d, "\circ" marking, "\pi_{M,M}"] \\
M & M \times M \arrow[l, "\cdot"']
\end{tikzcd} \qquad (4.3)$$

($\Leftarrow$) Suppose that the finite monoid $(M, \cdot, e)$ effectfully recognizes L via $h\colon \Sigma^* \mathrel{-\!\!\circ\!\!\to} M$ and $p\colon M \to O$. Then the $\mathbb{T}$-FA $\mathcal{A} = (M, \delta, e, p)$ whose transition map $\delta\colon M \times \Sigma \mathrel{-\!\!\circ\!\!\to} M$ is given by the composite (4.3) (where $h_0(a) = h(a)$ for $a \in \Sigma$) computes L. ◀

▶ **Example 4.23.** The condition of Theorem 4.22 holds for $\mathbb{T} \in \{\mathcal{D}, \mathcal{C}, \mathcal{S}\}$, as the maps $\xi_0\colon M \to (X \mathrel{-\!\!\circ\!\!\to} X)$ from Figure 4 witnessing that $X \mathrel{-\!\!\circ\!\!\to} X$ is finitely generated also witness that $X \mathrel{-\!\!\circ\!\!\to} X$ is *monoidally* finitely generated. We note that the verification that $\xi\colon TM \to (X \mathrel{-\!\!\circ\!\!\to} X)$ is a monoid morphism is non-trivial unless $\mathbb{T}$ is commutative (see Remark 4.20). Proposition 4.24 below simplifies the computations for non-commutative $\mathbb{T}$, and also explains conceptually *why* the non-commutative monads $\mathcal{C}$ and $\mathcal{S}$ satisfy the condition: for a finite set X, the $(\mathbb{T}, \mathcal{M})$-bialgebra $X \mathrel{-\!\!\circ\!\!\to} X$ is generated by *central* morphisms.

Recall that a Kleisli morphism $f\colon X \mathrel{-\!\!\circ\!\!\to} Y$ is *central* [22, 49] if for all Kleisli morphisms $f'\colon X' \mathrel{-\!\!\circ\!\!\to} Y'$, the following diagram commutes in $\mathcal{K}\ell(\mathbb{T})$.

$$\begin{tikzcd}
& TY \times X' \arrow[r, "\circ" marking, "\mathsf{rs}_{Y,X'}"] & Y \times X' \arrow[r, "1\times f'"] & Y \times TY' \arrow[dr, "\circ" marking, "\mathsf{ls}_{Y,Y'}"] & \\
X \times X' \arrow[ur, "f\times 1"] \arrow[dr, "1\times f'"'] & & & & Y \times Y' \\
& X \times TY' \arrow[r, "\circ" marking, "\mathsf{ls}_{X,Y'}"] & X \times Y' \arrow[r, "f\times 1"] & TY \times Y' \arrow[ur, "\circ" marking, "\mathsf{rs}_{Y,Y'}"'] &
\end{tikzcd} \qquad (4.4)$$

▶ **Proposition 4.24.** *If $\xi_0\colon M \to (X \mathrel{-\!\!\circ\!\!\to} X)$ is a monoid morphism whose uncurried form $X \times M \mathrel{-\!\!\circ\!\!\to} X$ is central, then its free extension $\xi\colon TM \to (X \mathrel{-\!\!\circ\!\!\to} X)$ is a monoid morphism.*

▶ **Remark 4.25.** The witness $\xi_0\colon (X \to X) \to (X \mathrel{-\!\!\circ\!\!\to} X)$ defined by $f \mapsto f \,;\, \eta_X$ for the monad $\mathcal{D}$ from Definition 3.4 works more generally for all *affine monads* [35] (i.e. monads $\mathbb{T}$ satisfying $T1 \cong 1$). Hence, our Theorems 4.11 and 4.22 apply to all affine monads $\mathbb{T}$.

4.5 Applications

We proceed to instantiate Theorems 4.11 and 4.22 to derive algebraic characterizations for several effectful automata models. For $\mathbb{T} = \mathcal{D}$, we recover Theorems 3.10 and 3.18 for PFAs:

▶ **Theorem 4.26.** *A probabilistic language is PFA-computable iff it is $\mathcal{D}$-effectfully recognized by a finite monoid iff it is recognized by an fg-carried convex monoid.*

For $\mathbb{T} = \mathcal{C}$, we obtain a new algebraic characterization of NPFAs. Note that a $(\mathcal{C}, \mathcal{M})$*-bialgebra* is a convex semilattice with an additional monoid structure.

▶ **Theorem 4.27.** *A probabilistic language is NPFA-computable iff it is $\mathcal{C}$-effectfully recognized by a finite monoid iff it is recognized by an fg-carried $(\mathcal{C}, \mathcal{M})$-bialgebra.*

Finally, $\mathbb{T} = \mathcal{S}$ gives a new algebraic characterization of WFAs. Note that an $(\mathcal{S}, \mathcal{M})$*-bialgebra* is an S-semimodule with an additional monoid structure. For a commutative semiring S, the notion of an $\mathcal{S}$-monoid is that of an *(associative) S-algebra.*

▶ **Theorem 4.28.** *An S-weighted language is WFA-computable iff it is $\mathcal{S}$-effectfully recognized by a finite monoid iff it is recognized by an fg-carried $(\mathcal{S}, \mathcal{M})$-bialgebra, and, for a commutative semiring S, iff it is recognized by an fg-carried S-algebra.*

For the case where S is a commutative ring (i.e. has additive inverses), we recover the correspondence between WFAs and fg-carried S-algebras due to Reutenauer [52]. However, Theorem 4.28 applies to every semiring, so it also yields novel algebraic characterizations of min- and max-plus automata and transducers (Example 4.53) as special instances.

5 Conclusions and Future Work

We have developed the foundations of an algebraic theory of automata with generic computational effects. Under suitable conditions on the effect monad $\mathbb{T}$, we characterized $\mathbb{T}$-FA-computable languages by algebraic modes of effectful recognition. As special cases, this entails the first algebraic characterizations of probabilistic automata and weighted automata over unrestricted semirings. We proceed to give some prospects for future work.

Proving finite presentability of syntactic convex monoids is challenging even for simple probabilistic languages, as Example 3.24 illustrates. Identifying conditions on the language ensuring this property is thus a natural question.

We aim to extend our theory beyond the category of sets. Of particular interest are *nominal sets*, which would allow us to capture effectful (e.g. probabilistic or weighted) register automata [15]. The main technical hurdle lies in the construction $Q \mapsto (Q \to Q)$ of function spaces, which does not preserve orbit-finite (i.e. finitely presentable) nominal sets. This might be overcome via the recently proposed restriction to *single-use* functions [16, 17].

An orthogonal generalization of our theory concerns effectful languages beyond finite words. This requires switching from monoids to algebras for a monad $\mathbb{S}$ [14]. For example, taking the monad $\mathbb{S}$ corresponding to *ω-semigroups* [45] could lead to algebraic recognition of effectful languages over *infinite words*. For a generalization of the recognition by $\mathbb{T}$-monoids, we expect that the interaction between the monads $\mathbb{S}$ and $\mathbb{T}$ be given by some form of distributive law similar to the interaction of $\mathcal{M}$ and $\mathbb{T}$.

Lastly, our effectful automata/monoid correspondence could pave the way to a *topological* account of effectful languages based on effectful profinite monoids. A first glimpse in this direction is given by Fijalkow [27] who analyzes the value-1 problem for PFAs in terms of a notion of *free prostochastic monoid*. We aim to study effectful versions of the duality theory of profinite monoids [11, 28] and its applications, notably variety theorems [29, 63].

References

1 Jiří Adámek, Stefan Milius, Nathan J. Bowler, and Paul Blain Levy. Coproducts of monads on set. In *27th Annual IEEE Symposium on Logic in Computer Science, (LICS 2012)*, pages 45–54. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.16.

2 Jiří Adámek, Stefan Milius, and Henning Urbat. A categorical approach to syntactic monoids. *Log. Methods Comput. Sci.*, 14(2):9:1–9:34, 2018. doi:10.23638/LMCS-14(2:9)2018.

3 Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories.* London Mathematical Society Lecture Note Series. Cambridge University Press, 1994. doi:10.1017/CBO9780511600579.

4 Jiří Adámek, Jiří Rosický, and E. M. Vitale. *Algebraic Theories: A Categorical Introduction to General Algebra.* Cambridge Tracts in Mathematics. Cambridge University Press, 2010. doi:10.1017/CBO9780511760754.

5 Jiří Adámek and Vera Trnková. *Automata and Algebras in Categories.* Springer, 1990.

6 Jorge Almeida. On pseudovarieties, varieties of languages, filters of congruences, pseudoidentities and related topics. *Algebra Universalis*, 27(3):333–350, 1990. doi:10.1007/BF01190713.

7 Michael A. Arbib and Ernest G. Manes. Machines in a category: An expository introduction. *SIAM Review*, 16(2):163–192, 1974. doi:10.1137/1016026.

8 Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

9 Bernhard Banaschewski and Evelyn Nelson. Tensor products and bimorphisms. *Canadian Mathematical Bulletin*, 19(4):385–402, 1976. doi:10.4153/CMB-1976-060-2.

10 Nicolas Bedon and Chloé Rispal. Schützenberger and Eilenberg theorems for words on linear orderings. *J. Comput. Syst. Sci.*, 78(2):517–536, 2012. doi:10.1016/J.JCSS.2011.06.003.

11 Fabian Birkmann, Henning Urbat, and Stefan Milius. Monoidal extended Stone duality. In *27th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2024)*, volume 14574 of *Lect. Notes Comput. Sci.*, pages 144–165. Springer, 2024. doi:10.1007/978-3-031-57228-9_8.

12 Achim Blumensath. Regular tree algebras. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:16)2020.

13 Achim Blumensath. Algebraic language theory for Eilenberg–Moore algebras. *Log. Methods Comput. Sci.*, 17(2), 2021. doi:10.23638/LMCS-17(2:6)2021.

14 Mikołaj Bojańczyk. Recognisable languages over monads. In Igor Potapov, editor, *Proc. 19th International Conference on Developments in Language Theory (DLT)*, volume 9168 of *Lect. Notes Comput. Sci.*, pages 1–13. Springer, 2015. doi:10.1007/978-3-319-21500-6_1.

15 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.

16 Mikołaj Bojańczyk, Lê Thành Dung Nguyên, and Rafal Stefański. Function spaces for orbit-finite sets. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *LIPIcs*, pages 130:1–130:20. Schloss Dagstuhl, 2024. doi:10.4230/LIPICS.ICALP.2024.130.

17 Mikołaj Bojańczyk and Rafal Stefański. Single-use automata and transducers for infinite alphabets. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl, 2020. doi:10.4230/LIPIcs.ICALP.2020.113.

18 Mikołaj Bojanczyk and Igor Walukiewicz. Forest algebras. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 107–132. Amsterdam University Press, 2008.

19 Benedikt Bollig, Paul Gastin, Benjamin Monmege, and Marc Zeitoun. A probabilistic Kleene theorem. In *10th International Symposium on Automated Technology for Verification and Analysis (ATVA 2012)*, volume 7561 of *Lect. Notes Comput. Sci.*, pages 400–415. Springer, 2012. doi:10.1007/978-3-642-33386-6_31.

20 Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. The theory of traces for systems with nondeterminism, probability, and termination. *Log. Methods Comput. Sci.*, 18(2), 2022. doi:10.46298/LMCS-18(2:21)2022.

21 Rais G. Bukharaev. *Theorie der stochastischen Automaten.* Vieweg+Teubner, 1995.

22 Titouan Carette, Louis Lemonnier, and Vladimir Zamdzhiev. Central submonads and notions of computation: Soundness, completeness and internal languages. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2023)*, pages 1–13. IEEE, 2023. doi:10.48550/arXiv.2207.09190.

23 Thomas Colcombet. Regular cost functions, part i: Logic and algebra over words. *Log. Methods Comput. Sci.*, 9(3), 2013. doi:10.2168/LMCS-9(3:3)2013.

24 Thomas Colcombet and Daniela Petrişan. Automata minimization: a functorial approach. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:32)2020.

25 Volker Diekert, Paul Gastin, and Manfred Kufleitner. A survey on small fragments of first-order logic over finite words. *Int. J. Found. Comput. Sci.*, 19(3):513–548, 2008. doi:10.1142/S0129054108005802.

26 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata.* Springer, 2009. doi:10.1007/978-3-642-01492-5.

27 Nathanaël Fijalkow. Profinite techniques for probabilistic automata and the Markov monoid algorithm. *Theor. Comput. Sci.*, 680:1–14, 2017. doi:10.1016/j.tcs.2017.04.006.

28 Mai Gehrke. Stone duality, topological algebra, and recognition. *Journal of Pure and Applied Algebra*, 220(7):2711–2747, 2016. doi:10.1016/j.jpaa.2015.12.007.

29 Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin. Duality and equational theory of regular languages. In *35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5126 of *Lect. Notes Comput. Sci.*, pages 246–257. Springer, 2008. doi:10.1007/978-3-540-70583-3_21.

30 Joseph A. Goguen. Minimal realization of machines in closed categories. *Bulletin of the American Mathematical Society*, 78(5):777–783, 1972. doi:10.1090/S0002-9904-1972-13032-5.

31 Sergey Goncharov, Stefan Milius, and Alexandra Silva. Towards a coalgebraic Chomsky hierarchy. In *8th International Conference on Theoretical Computer Science (TCS 2014)*, volume 8705 of *Lect. Notes Comput. Sci.*, pages 265–280. Springer, 2014. doi:10.1007/978-3-662-44602-7_21.

32 Branko Grünbaum, Volker Kaibel, Victor Klee, and Günter M. Ziegler. *Convex Polytopes.* Springer, 2003. doi:10.1007/978-1-4613-0019-9.

33 Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610, 2022. doi:10.1007/s10009-021-00633-z.

34 Anders Kock. Monads on symmetric monoidal closed categories. *Arch. Math*, 21:1–10, 1970. doi:10.1007/BF01220868.

35 Anders Kock. Bilinearity and cartesian closed monads. *Mathematica Scandinavica*, 29:161–174, 1971. doi:10.7146/math.scand.a-11042.

36 Anders Kock. Closed categories generated by commutative monads. *Journal of the Australian Mathematical Society*, 12(4):405–424, 1971. doi:10.1017/S1446788700010272.

37 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011. doi:10.1007/978-3-642-22110-1_47.

38 Fabian Lenke, Stefan Milius, Henning Urbat, and Thorsten Wißmann. Algebraic language theory with effects. *CoRR*, 2024. doi:10.48550/arXiv.2410.12569.

39 Saunders Mac Lane. *Categories for the Working Mathematician.* Springer, 1998. doi:10.1007/978-1-4757-4721-8.

40 Ernest G. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics.* Springer, 1976. doi:10.1007/978-1-4612-9860-1.

41 Ernest G. Manes and Philip Mulry. Monad compositions I: General constructions and recursive distributive laws. *Theory and Applications of Categories*, 18:172–208, 2007.

42 Robert McNaughton and Seymour A. Papert. *Counter-Free Automata (M.I.T. Research Monograph No. 65).* The MIT Press, 1971. doi:10.5555/1097043.

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

44 Azaria Paz. *Introduction to probabilistic automata.* Elsevier, 1971. doi:10.1016/c2013-0-11297-4.

45 Dominique Perrin and Jean-Éric Pin. *Infinite words - automata, semigroups, logic and games*, volume 141 of *Pure and applied mathematics series.* Elsevier Morgan Kaufmann, 2004.

46 Jean-Éric Pin. *Varieties Of Formal Languages.* Plenum Publishing Co., 1986.

47 Jean-Éric Pin, editor. *Handbook of Automata Theory.* European Mathematical Society Publishing House, Zürich, Switzerland, 2021. doi:10.4171/Automata.

48 Jean-Eric Pin. Mathematical foundations of automata theory. https://www.irif.fr/ jep/PDF/MPRI/MPRI.pdf, February 2022.

49 John Power. Premonoidal categories as categories with algebraic structure. *Theor. Comput. Sci.*, 278(1-2):303–321, 2002. doi:10.1016/S0304-3975(00)00340-6.

50 M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959. doi:10.1147/rd.32.0114.

51 Michael O. Rabin. Probabilistic automata. *Inform. Control*, 6(3):230–245, 1963. doi:10.1016/S0019-9958(63)90290-0.

52 C. Reutenauer. Séries formelles et algèbres syntactiques. *J. Algebra*, 66:448–483, 1980. doi:10.1016/0021-8693(80)90097-6.

53 Wojciech Różowski and Alexandra Silva. A completeness theorem for probabilistic regular expressions. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2024)*. ACM, 2024. doi:10.1145/3661814.3662084.

54 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

55 Saeed Salehi and Magnus Steinby. Tree algebras and varieties of tree languages. *Theoret. Comput. Sci.*, 377(1-3):1–24, 2007. doi:10.1016/j.tcs.2007.02.006.

56 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Inform. and Control*, 8:190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.

57 Gavin J. Seal. Tensors, monads and actions. *Theory and Applications of Categories*, 28:403–433, 2012. doi:10.48550/arXiv.1205.0101.

58 Roberto Segala. *Modeling and Verification of Randomized Distributed Real-time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995. URL: https://hdl.handle.net/1721.1/36560.

59 Alexandra Silva, Filippo Bonchi, Marcello Bonsangue, and Jan Rutten. Generalizing determinization from automata to coalgebras. *Log. Methods Comput. Sci.*, 9(1), 2013. doi:10.2168/LMCS-9(1:9)2013.

60 Ana Sokolova and Harald Woracek. Congruences of convex algebras. *Journal of Pure and Applied Algebra*, 219(8):3110–3148, 2015. doi:10.1016/j.jpaa.2014.10.005.

61 Marshall H. Stone. Postulates for the barycentric calculus. *Annali di Matematica Pura ed Applicata*, 29(1):25–30, 1949. doi:10.1007/BF02413910.

62 Tadeusz Swirszcz. Monadic functors and convexity. *Polonaise des Sciences. Sér. des sciences math., astr. et phys.*, 22, 1974.

63 Henning Urbat, Jiří Adámek, Liang-Ting Chen, and Stefan Milius. Eilenberg theorems for free. In *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, volume 83 of *LIPIcs*, pages 43:1–43:14. Schloss Dagstuhl, 2017. doi:10.4230/LIPIcs.MFCS.2017.43.

64 Gerco van Heerdt, Justin Hsu, Joël Ouaknine, and Alexandra Silva. Convex language semantics for nondeterministic probabilistic automata. In *15th International Colloquium on Theoretical Aspects of Computing (ICTAC 2018)*, volume 11187 of *lncs*, pages 472–492. Springer, 2018. doi:10.1016/j.tcs.2025.115191.

65 Thomas Weidner. Probabilistic automata and probabilistic logic. In *37th International Symposium on Mathematical Foundations of Computer Science (MFCS 2012)*, volume 7464 of *Lect. Notes Comput. Sci.*, pages 813–824. Springer, 2012. doi:10.1007/978-3-642-32589-2_70.

66 Thomas Wilke. An Eilenberg theorem for ∞-languages. In *Proc. 18th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 510 of *Lect. Notes Comput. Sci.*, pages 588–599. Springer, 1991. doi:10.5555/646245.756766.