Theoretical Computer Science 1040 (2025) 115191

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Convex language semantics for nondeterministic probabilistic automata [☆,☆☆,☆☆☆]

Gerco van Heerdt [a,*], Justin Hsu [b], Joël Ouaknine [c], Alexandra Silva [b]

[a] Droit Financial Technologies, United Kingdom
[b] Cornell University, United States
[c] MPI-SWS, Germany

A B S T R A C T

We explore language semantics for automata combining probabilistic and nondeterministic behaviors. We first show that there are precisely two natural semantics for probabilistic automata with nondeterminism. For both choices, we show that these automata are strictly more expressive than deterministic probabilistic automata, and we prove that the problem of checking language equivalence is undecidable by reduction from the threshold problem. However, we provide a discounted metric that can be computed to arbitrarily high precision.

1. Introduction

Probabilistic automata are fundamental models of randomized computation. They serve as useful tools to study the semantics and correctness of probabilistic programming languages [2,3], randomized algorithms [4,5], and machine learning [6,7]. Likewise, nondeterministic automata are well-studied models for concurrent and distributed systems [8].

Interest in systems that exhibit both random and nondeterministic behaviors goes back to Rabin's randomized techniques to increase the efficiency of distributed algorithms in the 1970s and 1980s [4,5]. This line of research yielded several automaton models supporting both nondeterministic and probabilistic choice [9–11]. Formal techniques and tools developed for these models have been successfully used in verification tasks [12,13,11,14], but there are multiple ways of combining nondeterminism and randomization, and there remains room for further investigation.

In this paper we study nondeterministic probabilistic automata (NPAs) and propose a novel probabilistic language semantics. NPAs are similar to Segala systems [9] in that transitions can make combined nondeterministic and probabilistic choices, but NPAs also have an output weight in $[0, 1]$ for each state, reminiscent of observations in Markov Decision Processes. This enables us to define the *expected* weight of a word in a similar way as in standard nondeterministic automata: the output of an NPA on an input word can be computed in a *deterministic* version of the automaton, using a careful choice of algebraic structure for the state space.

The resulting notion of equivalence is a type of language equivalence (also known as trace equivalence) and is coarser than probabilistic bisimulation [15–18], which distinguishes systems with different branching structure even if the total weight assigned

☆ This article belongs to Section B: Logic, semantics and theory of programming, Edited by Don Sannella.

☆☆ This is an extended version of the ICTAC'18 paper with the same title [1], including more examples, all proofs, and a new section (Section 3.3). This work was developed when the first and last author were at University College London. The work was partially supported by ERC starting grant ProFoundNet (679127), ERC consolidator grant AVS-ISS (648701), EPSRC Standard Grant CLeVer (EP/S028641/1), a Leverhulme Prize (PLP-2016-129), a Royal Society Wolfson Fellowship, and an NSF grant (1637532).

☆☆☆ Joël Ouaknine is also affiliated with Keble College, Oxford as emmy.network Fellow, and supported by DFG grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science).

* Corresponding author.
E-mail address: alexandra.silva@gmail.com (A. Silva).

https://doi.org/10.1016/j.tcs.2025.115191
Received 6 April 2019; Received in revised form 1 June 2023; Accepted 17 March 2025

Available online 21 March 2025

to a word is the same. This difference is well-known, even for purely probabilistic systems [19]; different target applications may call for different notions of equivalence.

After reviewing mathematical preliminaries in Section 2, we introduce the NPA model and explore its semantics in Section 3. We show that there are precisely two natural ways to define the language semantics of such systems—by either taking the maximum or the minimum of the weights associated with the different paths labeled by an input word. The proof of this fact relies on an abstract view on these automata generating probabilistic languages with algebraic structure. Specifically, probabilistic languages have the structure of a *convex algebra*, analogous to the join-semilattice structure of standard languages. These features can abstractly be seen as so-called *Eilenberg-Moore algebras* for a monad—the distribution and the powerset monads, respectively—which can support new semantics and proof techniques (see, e.g., [16,20]). In order to support the use of NPAs in practice and proof techniques for their equivalence we explore a finite representation of their semantics (Section 3.3). We also include a soundness theorem for the finitary determinization construction.

In Section 4, we compare NPAs with standard, deterministic probabilistic automata (DPAs) as formulated by Rabin [21]. Our semantics ensures that NPAs recover DPAs in the special case when there is no nondeterministic choice. Furthermore, we show that there are weighted languages accepted by NPAs that are not accepted by any DPA. We use the theory of linear recurrence sequences to give a separation even for weighted languages over a unary alphabet.

In Section 5, we turn to equivalence. We prove that language equivalence of NPAs is undecidable by reduction from so-called *threshold problems*, which are undecidable [22–24]. The hard instances encoding the threshold problem are equivalences between probabilistic automata over a two-letter alphabet. Thus, the theorem immediately implies that equivalence of NPAs is undecidable when the alphabet size is at least two. The situation for automata over unary alphabets is more subtle; in particular, the threshold problem over a unary alphabet is not known to be undecidable. However, we give a reduction from the Positivity problem on linear recurrence sequences, a problem where a decision procedure would likely entail breakthroughs in open problems in number theory [25]. Finally, we propose a discounted metric on languages. This metric cannot be computed exactly due to the undecidability result, but we show that the metric can be approximated to arbitrarily high precision.

We survey related work and conclude in Section 6.

2. Preliminaries

We start by introducing some mathematical background on convex algebras, monads, probabilistic automata, and language semantics.

2.1. Convex algebra

A set A is a *convex algebra*, or a *convex set*, if for all $n \in \mathbb{N}$ and tuples $(p_i)_{i=1}^n$ of numbers in $[0,1]$ summing up to 1 (that is, $\sum_{i=1}^n p_i = 1$) there is an operation denoted $\sum_{i=1}^n p_i(-)_i : A^n \to A$ satisfying the following properties for $(a_1, \dots, a_n) \in A^n$:

Projection. If $p_j = 1$ (and hence $p_i = 0$ for all $i \neq j$), we have $\sum_{i=1}^n p_i a_i = a_j$.
Barycenter. For any n tuples $(q_{i,j})_{j=1}^m$ in $[0,1]$ satisfying for all $i = 1, \cdots, n$, $\sum_{j=1}^m q_{i,j} = 1$, we have

$$\sum_{i=1}^n p_i \left(\sum_{j=1}^m q_{i,j} a_j \right) = \sum_{j=1}^m \left(\sum_{i=1}^n p_i q_{i,j} \right) a_j.$$

Informally, a convex algebra structure gives a way to take finite convex combinations of elements in a set A. Given this structure, we can define convex subsets and generate them by elements of A.

Example 2.1 *(Unit interval).* The interval $[0,1]$ has a standard convex structure which we use throughout this paper. Namely, for $A = [0,1]$, the operation $\sum_{i=1}^n p_i(-)_i : A^n \to A$ is simply defined using standard sum and multiplication operations over the elements of $[0,1]$. More generally, any interval $[a,b] \subset \mathbb{R}$ is a convex set using the operation as above.

Definition 2.2. A subset $S \subseteq A$ is *convex* if it is closed under all convex combinations. That is, it has to be a convex subalgebra. A convex set S is *generated* by a set $G \subseteq A$ if for all $s \in S$, there exist $n \in \mathbb{N}$, $(p_i)_{i=1}^n$, $(g_i)_{i=1}^n \in G^n$ such that $s = \sum_i p_i g_i$. When G is finite, we say that S is *finitely generated.*

We can also define morphisms between convex sets.

Definition 2.3. An *affine map* between two convex sets A and B is a function $h : A \to B$ commuting with convex combinations:

$$h\left(\sum_{i=1}^n p_i a_i\right) = \sum_{i=1}^n p_i h(a_i).$$

Example 2.4. For $A = [0, 1]$ and $B = [1/2, 3/2]$ the map $h : A \to B$ given by

$$h(a) = a + 1/2$$

is an affine map. We can show this using basic arithmetic on real numbers and the fact that $(p_i)_{i=1}^n$ is a sequence of numbers in $[0, 1]$ summing up to 1:

$$\begin{aligned} h\left(\sum_{i=1}^n p_i a_i\right) &= \left(\sum_{i=1}^n p_i a_i\right) + 1/2 \\ &= \left(\sum_{i=1}^n p_i a_i\right) + 1 \times 1/2 \\ &= \left(\sum_{i=1}^n p_i a_i\right) + \left(\sum_{i=1}^n p_i\right) \times 1/2 \\ &= \sum_{i=1}^n p_i(a_i + 1/2) \\ &= \sum_{i=1}^n p_i h(a_i). \end{aligned}$$

2.2. Monads and their algebras

Our definition of language semantics will be based on the category theoretic framework of monads and their algebras. Monads can be used to model computational side-effects such as nondeterminism and probabilistic choice. An algebra allows us to interpret such side-effects within an object of the category.

Definition 2.5. A *monad* (T, η, μ) consists of an endofunctor T and two natural transformations: a *unit* $\eta : Id \Rightarrow T$ and a *multiplication* $\mu : TT \Rightarrow T$, making the following diagrams commute.

$$\begin{tikzcd} T \arrow[r, "\eta"] \arrow[d, "T\eta"'] \arrow[dr, equal] & TT \arrow[d, "\mu"] \\ TT \arrow[r, "\mu"] & T \end{tikzcd} \qquad \begin{tikzcd} TTT \arrow[r, "T\mu"] \arrow[d, "\mu"'] & TT \arrow[d, "\mu"] \\ TT \arrow[r, "\mu"] & T \end{tikzcd}$$

When there is no risk of confusion, we identify a monad with its endofunctor.

Example 2.6 *(Monads)*. We give three examples of monads in the category of sets and functions.

1. The non-empty powerset monad is given by $(\mathcal{P}_f, \{-\}, \bigcup)$, where $\mathcal{P}_f$ denotes the finite nonempty powerset functor sending each set to the set of its finite nonempty subsets, $\{-\}$ is the singleton operation, and $\bigcup$ is set union.
2. The list monad is the triple $((-)^*, \mathsf{nil}, +\!+)$, where $(-)^*$ denotes the finite list functor sending each set A to the set A^* of finite lists over A, nil is the empty list, and $+\!+$ is list concatenation.
3. The vector space monad (over a field $\mathbb{F}$) is given by $(V(-), \mathsf{unit}, \mu)$, where $V(X)$ denotes the free vector space generated by X: the vector space with basis X, the set of abstract linear combinations of elements of X with coefficients in $\mathbb{F}$; unit maps an element of X to the corresponding basis vector, and $\mu : V(V(X)) \to V(X)$ sends an abstract linear combination of vectors to the actual linear combination which is a well-defined element of $V(X)$.

We will see in Section 2.3 two other examples of monads: the distribution monad (over sets) and the convex powerset functor (over convex sets).

Definition 2.7. An *algebra for a monad* (T, η, μ) is a pair (X, h) consisting of a carrier set X and a function $h : TX \to X$ making the following diagrams commute.

$$\begin{tikzcd} X \arrow[r, "\eta"] \arrow[dr, equal] & TX \arrow[d, "h"] \\ & X \end{tikzcd} \qquad \begin{tikzcd} TTX \arrow[r, "Th"] \arrow[d, "\mu"'] & TX \arrow[d, "h"] \\ TX \arrow[r, "h"] & X \end{tikzcd}$$

Definition 2.8. A homomorphism from an algebra (X, h) to an algebra (Y, k) for a monad T is a function $f : X \to Y$ making the diagram below commute.

$$\begin{tikzcd} TX \arrow[r, "Tf"] \arrow[d, "h"'] & TY \arrow[d, "k"] \\ X \arrow[r, "f"] & Y \end{tikzcd}$$

Example 2.9 *(Algebras for a monad).* The algebras for the finite powerset monad are precisely the join-semilattices with bottom, and their homomorphisms are maps that preserve finite joins. The algebras for the finite list monad are monoids, and their homomorphisms are maps that preserve the monoid operation. The algebras for the vector space monad are vector spaces and their homomorphisms are linear maps.

The algebras for any monad together with their homomorphisms form a category, the so-called category of *Eilenberg-Moore algebras* for a monad.

2.3. Distribution and convex powerset monads

We will work with two monads closely associated with convex sets.

In the category of sets, the *distribution monad* $(\mathcal{D}, \delta, m)$ maps a set X to the set of distributions over X with finite support. The support of a distribution d is given by

$$\mathsf{supp}(d) = \{a \in A \mid d(a) \neq 0\} \in \mathcal{P}_f A.$$

The unit $\delta\colon X \to \mathcal{D}X$ maps $x \in X$ to the point distribution at x. For the multiplication $m\colon \mathcal{D}\mathcal{D}X \to \mathcal{D}X$, let $d \in \mathcal{D}\mathcal{D}X$ be a finite distribution with support $\{d_1, \dots, d_n\} \in \mathcal{P}_f\mathcal{D}X$ and define

$$m(d)(x) = \sum_{i=1}^{n} d(d_i) \times d_i(x).$$

The category of algebras for the distribution monad is precisely the category of convex sets and affine maps—we will often implicitly identify these two representations.

In the category of convex sets, the *finitely generated nonempty convex powerset monad* [16] $(\mathcal{P}_c, \{-\}, \bigcup)$ maps a convex set A to the set of finitely generated nonempty convex subsets of A.[1] The convex algebra structure on $\mathcal{P}_c A$ is given by $\sum_{i=1}^n p_i U_i = \{\sum_{i=1}^n p_i u_i \mid u_i \in U_i \text{ for all } 1 \leq i \leq n\}$ with every $U_i \in \mathcal{P}_c A$. The unit map $\{-\}\colon A \to \mathcal{P}_c A$ maps $a \in A$ to a singleton convex set $\{a\}$, and the multiplication $\bigcup\colon \mathcal{P}_c\mathcal{P}_c A \to \mathcal{P}_c A$ is again the union operation, which collapses nested convex sets.

As an example, we can consider this monad on the convex algebra $[0,1]$. The result is a finitely generated convex set.

Lemma 2.10. *The convex set* $\mathcal{P}_c[0,1]$ *is generated by its elements* $\{0\}$, $\{1\}$, *and* $[0,1]$, *i.e.,* $\mathsf{conv}(\{\{0\},\{1\},[0,1]\}) = \mathcal{P}_c[0,1]$.

Proof. The finitely generated nonempty convex subsets of $[0,1]$ are of the form $[p,q]$ for $p,q \in [0,1]$, and $[p,q] = p\{1\} + (q-p)[0,1] + (1-q)\{0\}$. $\square$

To describe automata with both nondeterministic and probabilistic transitions, we will work with convex powersets of distributions. The functor $\mathcal{P}_c\mathcal{D}$ taking sets X to the set of finitely generated nonempty convex sets of distributions over X can be given a monad structure.[2]

Explicitly, the (affine) convex algebra structure on $\mathcal{P}_c A$ for any convex algebra (A, α) is the map $\omega_A\colon \mathcal{D}\mathcal{P}_c A \to \mathcal{P}_c A$ given by

$$\omega_A(d) = \{(\alpha \circ \mathcal{D}c)(d) \mid c \in \mathsf{choice}_A\},$$

where

$$\mathsf{choice}_A = \{c\colon \mathcal{P}_c A \to A, \forall U \in \mathcal{P}_c A.\, c(U) \in U\},$$

which induces the composite monad $(\mathcal{P}_c\mathcal{D}, \hat{\delta}, \hat{m})$ defined by

$$\begin{tikzcd} X \arrow[d, "\delta"'] \arrow[dr, dashed, "\hat{\delta}"] & \\ \mathcal{D}X \arrow[r, "\{-\}"] & \mathcal{P}_c\mathcal{D}X \end{tikzcd} \qquad \begin{tikzcd} \mathcal{P}_c\mathcal{D}\mathcal{P}_c\mathcal{D}X \arrow[d, "\mathcal{P}_c\omega"'] \arrow[dr, dashed, "\hat{m}"] & \\ \mathcal{P}_c\mathcal{P}_c\mathcal{D}X \arrow[r, "\bigcup"] & \mathcal{P}_c\mathcal{D}X \end{tikzcd} \tag{1}$$

[1] In prior work [16], the monad was defined to take all convex subsets rather than just the finitely generated ones. However, since all the monad operations preserve finiteness of the generators, the restricted monad we consider is also well-defined.

[2] Formally, the functor $\mathcal{D}$ in this composition is a functor from the category of sets to the category of convex sets. To simplify notation, we identify this functor with the distribution functor $\mathcal{D}$ we defined above since they are defined in the same way on sets. One can show that the codomain of the set endofunctor $\mathcal{D}$ is indeed convex sets.

Note that ω is natural since $\mathcal{P}_c$ is a functor on the category of convex sets.

For all convex sets (A, α) and $S \in \mathcal{P}_f A$, we can define the *convex closure* of S (sometimes called the *convex hull*) $\mathsf{conv}(S) \in \mathcal{P}_c A$ by

$$\mathsf{conv}(S) = \{\alpha(d) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\}.$$

The conv maps form a natural transformation, a fact we will use later. To show that this is the case, we first note that $\mathsf{conv} : \mathcal{P}_f A \to \mathcal{P}_c A$ can be written as

$$\mathsf{conv} = \mathcal{P}_f A \xrightarrow{\mathsf{dis}} \mathcal{P}_c \mathcal{D} A \xrightarrow{\mathcal{P}_c \alpha} \mathcal{P}_c A, \tag{2}$$

where

$$\mathsf{dis}(S) = \{d \in \mathcal{D}A \mid \mathsf{supp}(d) \subseteq S\}$$

can be defined on any set A. We start by showing that dis is natural.

Lemma 2.11. *For all sets A and B and functions $f : A \to B$, the diagram below commutes.*

$$\begin{tikzcd}
\mathcal{P}_f A \arrow[r, "\mathsf{dis}"] \arrow[d, "\mathcal{P}_f f"'] & \mathcal{P}_c \mathcal{D} A \arrow[d, "\mathcal{P}_c \mathcal{D} f"] \\
\mathcal{P}_f B \arrow[r, "\mathsf{dis}"] & \mathcal{P}_c \mathcal{D} B
\end{tikzcd}$$

Proof. Consider any $S \in \mathcal{P}_f A$. We will first show that

$$\{\mathcal{D}f(d) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\} = \{d \in \mathcal{D}B \mid \mathsf{supp}(d) \subseteq \{f(a) \mid a \in S\}\}. \tag{3}$$

For the inclusion from left to right, note that for each $d \in \mathcal{D}A$ such that $\mathsf{supp}(d) \subseteq S$ we have $b \in \mathsf{supp}(\mathcal{D}f(d))$ only if there exists $a \in S$ such that $f(a) = b$. Thus, $\mathsf{supp}(\mathcal{D}f(d)) \subseteq \{f(a) \mid a \in S\}$. Conversely, consider $d \in \mathcal{D}B$ such that $\mathsf{supp}(d) \subseteq \{f(a) \mid a \in S\}$. We define $d' \in \mathcal{D}A$ by

$$d'(a) = \frac{d(f(a))}{|\{a' \in S \mid f(a') = f(a)\}|}.$$

Then

$$\begin{aligned}
\mathcal{D}f(d')(b) &= \sum_{a \in A, f(a) = b} d'(a) && \text{(definition of } \mathcal{D}) \\
&= \sum_{a \in A, f(a) = b} \frac{d(f(a))}{|\{a' \in S \mid f(a') = f(a)\}|} && \text{(definition of } d') \\
&= \sum_{a \in A, f(a) = b} \frac{d(b)}{|\{a' \in S \mid f(a') = b\}|} \\
&= d(b).
\end{aligned}$$

Now we have

$$\begin{aligned}
(\mathcal{P}_c \mathcal{D} f \circ \mathsf{dis})(S) &= \mathcal{P}_c \mathcal{D} f(\{d \in \mathcal{D}A \mid \mathsf{supp}(d) \subseteq S\}) && \text{(definition of dis)} \\
&= \{\mathcal{D}f(d) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\} && \text{(definition of } \mathcal{P}_c) \\
&= \{d \in \mathcal{D}B \mid \mathsf{supp}(d) \subseteq \{f(a) \mid a \in S\}\} && (3) \\
&= \mathsf{dis}(\{f(a) \mid a \in S\}) && \text{(definition of dis)} \\
&= (\mathsf{dis} \circ \mathcal{P}_c f)(S) && \text{(definition of } \mathcal{P}_c). \qquad \square
\end{aligned}$$

Lemma 2.12. *For all convex sets (A, α) and (B, β) and affine maps $f : A \to B$, the diagram below commutes.*

$$\begin{tikzcd}
\mathcal{P}_f A \arrow[r, "\mathsf{conv}"] \arrow[d, "\mathcal{P}_f f"'] & \mathcal{P}_c A \arrow[d, "\mathcal{P}_c f"] \\
\mathcal{P}_f B \arrow[r, "\mathsf{conv}"] & \mathcal{P}_c B
\end{tikzcd}$$

Proof. As seen below, this follows almost directly from the previous result.

$$\begin{tikzcd}
\mathcal{P}_f A \arrow[rr, "\mathsf{conv}"] \arrow[rr, phantom, "(2)"', shift right=3] \arrow[dr, "\mathsf{dis}"'] \arrow[ddd, "\mathcal{P}_f f"'] & & \mathcal{P}_c A \arrow[ddd, "\mathcal{P}_c f"] \\
{} \arrow[d, phantom, "\textcircled{1}"] & \mathcal{P}_c\mathcal{D}A \arrow[ur, "\mathcal{P}_c\alpha"'] \arrow[d, "\mathcal{P}_c\mathcal{D}f"] & {} \arrow[d, phantom, "\textcircled{2}"] \\
{} & \mathcal{P}_c\mathcal{D}B \arrow[dr, "\mathcal{P}_c\beta"] & {} \\
\mathcal{P}_f B \arrow[rr, "\mathsf{conv}"'] \arrow[rr, phantom, "(2)", shift left=3] \arrow[ur, "\mathsf{dis}"] & & \mathcal{P}_c B
\end{tikzcd}$$

$\textcircled{1}$ Lemma 2.11 $\quad\textcircled{2}$ f is affine $\square$

2.4. Automata and language semantics

In this section we review the general language semantics for automata with side-effects provided by a monad (see, e.g., [26–28]). This categorical framework is the foundation of our language semantics for NPAs.

Definition 2.13 *(T-automaton).* Given a monad (T, η, μ) in the category of sets, an output set O, and a (finite) alphabet Σ, a T-*automaton* is defined by a tuple $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$, where S is the set of *states*, $s_0 \in S$ is the *initial state*, $\gamma : S \to O$ is the *output function*, and $\tau_a : S \to TS$ for $a \in \Sigma$ are the *transition functions*.

This abstract formulation encompasses many standard notions of automata. For instance, we recover deterministic (Moore) automata by letting T be the identity monad; deterministic acceptors are a further specialization where the output set is the set $2 = \{0, 1\}$, with 0 modeling rejecting states and 1 modeling accepting states. If we use the powerset monad, we recover nondeterministic acceptors and with the vector space monad one recovers weighted automata.

Any T-automaton can be determinized, using a categorical generalization of the powerset construction [27].

Definition 2.14 *(Determinization).* Given a monad (T, η, μ) in the category of sets, an output set O with a T-algebra structure $o : TO \to O$, and a (finite) alphabet Σ, a T-automaton $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ can be determinized into the deterministic automaton $(TS, s_0', \gamma', \{\tau_a'\}_{a\in\Sigma})$ given by $s_0' = \eta(s_0) \in TS$ and

$$\gamma' : TS \to O \qquad \tau_a' : TS \to TS$$
$$\gamma' = o \circ T\gamma \qquad \tau_a' = \mu \circ T\tau_a.$$

This construction allows us to define the language semantics of any T-automaton as the semantics of its determinization.

Definition 2.15. Given a monad (T, η, μ) in the category of sets, an output set O with a T-algebra structure $o : TO \to O$, and a (finite) alphabet Σ, the *language* accepted by a T-automaton $\mathcal{A} = (S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ is the function $\mathcal{L}_\mathcal{A} : \Sigma^* \to O$ given by $\mathcal{L}_\mathcal{A} = (l_\mathcal{A} \circ \eta)(s_0)$, where $l_\mathcal{A} : TS \to O^{\Sigma^*}$ is defined inductively by

$$l_\mathcal{A}(s)(\varepsilon) = (o \circ T\gamma)(s) \qquad l_\mathcal{A}(s)(av) = l_\mathcal{A}((\mu \circ T\tau_a)(s))(v).$$

Example 2.16 *(Deterministic probabilistic automata).* As an example of a T-automaton and its semantics, take T to be the distribution monad $\mathcal{D}$ and the output set O to be the interval $[0, 1]$. Instantiating the definition of T−automata (for finite[3] state space S) yields deterministic probabilistic automata (DPAs): automata with a finite state space S, an output function of type $S \to [0, 1]$, and transition functions of type $S \to \mathcal{D}S$ for each $a \in \Sigma$. To instantiate the language semantics of such automata, we use the usual $\mathcal{D}$-algebra structure $\mathbb{E} : \mathcal{D}[0, 1] \to [0, 1]$ computing the expected weight:

$$\mathbb{E}(d) = \sum_{a\in\mathsf{supp}(d)} d(a) \cdot a.$$

More concretely, the semantics works as follows. Let $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ be a DPA. At any time while reading a word, we are in a convex combination of states $\sum_{i=1}^n p_i s_i$ (equivalently, a distribution over states). The current output is given by evaluating the

[3] All concrete automata considered in this paper will have a finite state space (cf. Section 4), but this is not required by Definition 2.13. The reason for this is two-fold. First, the determinization procedure (Definition 2.14) often results in an automaton with an infinite state space even when starting from a T-automaton with a finite state space: e.g. the distribution monad, for example, does not preserve finite sets in general. Second, the semantics of each T-automaton is given in terms of a mapping to an automaton of languages (Definition 2.15) and that language automaton has an infinite state space. Even when T is the identity monad and we recover deterministic Moore automata the language automaton has as state space all power series O^{A^*} which is an infinite set.

sum $\sum_{i=1}^{n} p_i \gamma(s_i)$. On reading a symbol $a \in \Sigma$, we transition to the convex combination of convex combinations $\sum_{i=1}^{n} p_i \tau_a(s_i)$, say $\sum_{i=1}^{n} p_i \sum_{j=1}^{m_i} q_{i,j} s_{i,j}$, which is collapsed to the final convex combination $\sum_{i=1}^{n}\sum_{j=1}^{m_i} p_i q_{i,j} s_{i,j}$ (again, a distribution over states).

Remark 2.17. One may wonder if the automaton model would be more expressive if the initial state s_0 in an automaton $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ would be an element of TS rather than S. This is not the case, since we can always add a new element to S that simulates s_0 by setting its output to $(o \circ T\gamma)(s_0)$ and its transition on $a \in \Sigma$ to $(\mu \circ T\tau_a)(s_0)$.

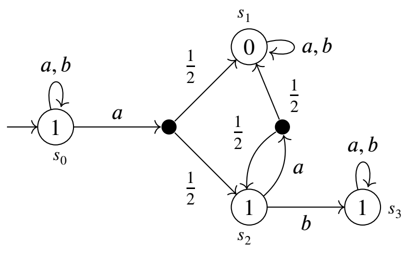
For instance, DPAs allowing a distribution over states as the initial state can be represented by an initial state distribution μ, an output vector γ, and transitions τ_a. In typical presentations, μ and γ are represented as weight vectors over states, and the τ_a are encoded by stochastic matrices.

3. Nondeterministic probabilistic automata

We now introduce an automaton model supporting probabilistic and nondeterministic behaviors, inspired by Segala [9]. On each input letter, the automaton can choose from a finitely generated nonempty convex set of distributions over states. After selecting a distribution, the automaton then transitions to its next state probabilistically. Each state has an output weight in $[0,1]$. The following formalization is an instantiation of Definition 2.13 with the monad $\mathcal{P}_c\mathcal{D}$.

Definition 3.1. A *nondeterministic probabilistic automaton* (NPA) over a (finite) alphabet Σ is defined by a tuple $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$, where S is a finite set of *states*, $s_0 \in S$ is the *initial state*, $\gamma\colon S \to [0,1]$ is the *output function*, and $\tau_a\colon S \to \mathcal{P}_c\mathcal{D}S$ are the *transition functions* indexed by inputs $a \in \Sigma$.

As an example, consider the NPA below.



(4)

States are labeled by their direct output (i.e., their weight from γ) while outgoing edges represent transitions. Additionally, we write the state name next to each state. We only indicate a set of generators of the convex subset that a state transitions into. If one of these generators is a distribution with nonsingleton support, then a transition into a black dot is depicted. Outgoing transitions represent the distribution, and are labeled with probabilities.

Our NPAs recognize weighted languages. The rest of the section formalizes the language semantics, based on the general framework from Section 2.4.

3.1. From convex algebra to language semantics

To define language semantics for NPAs, we will use the monad structure of $\mathcal{P}_c\mathcal{D}$. To be able to use the semantics from Section 2.4, we need to specify a $\mathcal{P}_c\mathcal{D}$-algebra structure $o\colon \mathcal{P}_c\mathcal{D}[0,1] \to [0,1]$. Moreover, our model should naturally coincide with DPAs when transitions make no nondeterministic choices, i.e., when each transition function maps each state to a singleton distribution over states. Thus, we require the $\mathcal{P}_c\mathcal{D}$-algebra o to extend the expected weight function $\mathbb{E}$. That is, we want o to be of the form

$$o = \mathcal{P}_c\mathcal{D}[0,1] \xrightarrow{\mathcal{P}_c\mathbb{E}} \mathcal{P}_c[0,1] \xrightarrow{\alpha} [0,1] \tag{5}$$

for some $\mathcal{P}_c$-algebra structure α on $[0,1]$. This in particular makes the diagram below commute, as a result of naturality of $\{-\}$ and the fact that $\alpha \circ \{-\} = \mathsf{id}$ by one of the algebra laws.[4]

$$\begin{tikzcd}
\mathcal{D}[0,1] \arrow[d, "\{-\}"'] \arrow[dr, "\mathbb{E}"] & \\
\mathcal{P}_c\mathcal{D}[0,1] \arrow[r, "o"] & {[0,1]}
\end{tikzcd} \tag{6}$$

[4] In the conference version of the present paper, we defined o extending $\mathbb{E}$ as the commutativity of (6). We then included a result [1, Proposition 1] implying that the two definitions—the one above given by (5) and the commutativity of (6)—are equivalent, but the proof given in the paper is incorrect: it applies the functor $\mathcal{P}_c$ to δ, which is not an affine map in general. We do not know whether the equivalence holds. We believe that our new notion of o extending $\mathbb{E}$ is more natural: if one thinks of α as a nondeterministic choice between expectations, the corresponding o is the function first calculating the expectation of each choice, followed by the appropriate choice itself.

3.2. Characterizing the $\mathcal{P}_c\mathcal{D}$-algebra on $[0,1]$

While in principle there could be many different $\mathcal{P}_c\mathcal{D}$-algebras on $[0,1]$ leading to different language semantics for NPAs, we will show that there are exactly two $\mathcal{P}_c$-algebras on $[0,1]$: the map computing the minimum and the map computing the maximum. This leads to two $\mathcal{P}_c\mathcal{D}$-algebras on $[0,1]$ that extend $\mathbb{E}$.

Proposition 3.2. *The only $\mathcal{P}_c$-algebras on the convex set $[0,1]$ are* min *and* max.

Proof. Let $\alpha\colon \mathcal{P}_c[0,1] \to [0,1]$ be a $\mathcal{P}_c$-algebra. Then for any $r \in [0,1]$, $\alpha(\{r\}) = r$, and the diagram below must commute.

$$\begin{tikzcd}
\mathcal{P}_c\mathcal{P}_c[0,1] \arrow[r, "\mathcal{P}_c\alpha"] \arrow[d, "\bigcup"'] & \mathcal{P}_c[0,1] \arrow[d, "\alpha"] \\
\mathcal{P}_c[0,1] \arrow[r, "\alpha"] & {[0,1]}
\end{tikzcd} \tag{7}$$

Furthermore, α is an affine map. Since $\mathsf{conv}(\{\{0\},\{1\},[0,1]\}) = \mathcal{P}_c[0,1]$ by Lemma 2.10, $\alpha(\{0\}) = 0$, and $\alpha(\{1\}) = 1$, α is completely determined by $\alpha([0,1])$. We now calculate that

$$\begin{aligned}
\alpha([0,1]) &= \alpha\left(\bigcup\{[0,p] \mid p \in [0,1]\}\right) \\
&= \left(\alpha \circ \bigcup \circ \mathsf{conv}\right)(\{\{0\},[0,1]\}) \\
&= (\alpha \circ \mathcal{P}_c\alpha \circ \mathsf{conv})(\{\{0\},[0,1]\}) && (7) \\
&= (\alpha \circ \mathsf{conv} \circ \mathcal{P}_f\alpha)(\{\{0\},[0,1]\}) && \text{(Lemma 2.12)} \\
&= (\alpha \circ \mathsf{conv})(\{\alpha(\{0\}),\alpha([0,1])\}) && \text{(definition of } \mathcal{P}_f\text{)} \\
&= (\alpha \circ \mathsf{conv})(\{0,\alpha([0,1])\}) \\
&= \alpha([0,\alpha([0,1])]) \\
&= \alpha(\alpha([0,1])[0,1] + (1-\alpha([0,1]))\{0\}) \\
&= \alpha([0,1]) \cdot \alpha([0,1]) + (1-\alpha([0,1])) \cdot \alpha(\{0\}) && (\alpha \text{ is affine}) \\
&= \alpha([0,1])^2 + (1-\alpha([0,1])) \cdot 0 \\
&= \alpha([0,1])^2.
\end{aligned}$$

Thus, we have either $\alpha([0,1]) = 0$ or $\alpha([0,1]) = 1$. Consider any finitely generated nonempty convex subset $[p,q] \subseteq [0,1]$. If $\alpha([0,1]) = 0$, then Lemma 2.10 gives

$$\begin{aligned}
\alpha([p,q]) &= \alpha(p\{1\} + (q-p)[0,1] + (1-q)\{0\}) \\
&= p \cdot \alpha(\{1\}) + (q-p) \cdot \alpha([0,1]) + (1-q) \cdot \alpha(\{0\}) \\
&= p \cdot 1 + (q-p) \cdot 0 + (1-q) \cdot 0 = p = \mathsf{min}([p,q]);
\end{aligned}$$

if $\alpha([0,1]) = 1$, then

$$\begin{aligned}
\alpha([p,q]) &= \alpha(p\{1\} + (q-p)[0,1] + (1-q)\{0\}) \\
&= p \cdot \alpha(\{1\}) + (q-p) \cdot \alpha([0,1]) + (1-q) \cdot \alpha(\{0\}) \\
&= p \cdot 1 + (q-p) \cdot 1 + (1-q) \cdot 0 = q = \mathsf{max}([p,q]).
\end{aligned}$$

We now show that min is an algebra; the case for max is analogous. We have

$$\begin{aligned}
\mathsf{min}\left(\sum_{i=1}^{n} r_i[p_i,q_i]\right) &= \mathsf{min}\left(\left[\sum_{i=1}^{n} r_i \cdot p_i, \sum_{i=1}^{n} r_i \cdot q_i\right]\right) \\
&= \sum_{i=1}^{n} r_i \cdot p_i \\
&= \sum_{i=1}^{n} r_i \cdot \mathsf{min}([p_i,q_i]),
\end{aligned}$$

so min is an affine map. Furthermore, clearly $\mathsf{min}(\{r\}) = r$ for all $r \in [0,1]$, and for all $S \in \mathcal{P}_c\mathcal{P}_c[0,1]$,

$$\mathsf{min}\left(\bigcup S\right) = \mathsf{min}(\{\mathsf{min}(T) \mid T \in S\}) = (\mathsf{min} \circ \mathcal{P}_c\mathsf{min})(S). \qquad \square$$

Corollary 3.3. *The only $\mathcal{P}_c\mathcal{D}$-algebras on $[0,1]$ extending $\mathbb{E}$ are $\mathcal{P}_c\mathcal{D}[0,1] \xrightarrow{\mathcal{P}_c\mathbb{E}} \mathcal{P}_c[0,1] \xrightarrow{\mathsf{min}} [0,1]$ and $\mathcal{P}_c\mathcal{D}[0,1] \xrightarrow{\mathcal{P}_c\mathbb{E}} \mathcal{P}_c[0,1] \xrightarrow{\mathsf{max}} [0,1]$.*

Consider again the NPA (4). Since we can always choose to remain in the initial state, the max semantics assigns 1 to each word for this automaton. The min semantics is more interesting. Consider reading the word aa. On the first a, we transition from s_0 to $\mathsf{conv}\{s_0, \frac{1}{2}s_1 + \frac{1}{2}s_2\} \in \mathcal{P}_c\mathcal{D}S$. Reading the second a gives

$$\mathsf{conv}\left\{\mathsf{conv}\left\{s_0, \tfrac{1}{2}s_1 + \tfrac{1}{2}s_2\right\}, \tfrac{1}{2}\{s_1\} + \tfrac{1}{2}\left\{\tfrac{1}{2}s_1 + \tfrac{1}{2}s_2\right\}\right\} \in \mathcal{P}_c\mathcal{D}\mathcal{P}_c\mathcal{D}S.$$

Now we first apply $\mathcal{P}_c\omega$ to eliminate the outer distribution, arriving at

$$\mathsf{conv}\left\{\mathsf{conv}\left\{s_0, \tfrac{1}{2}s_1 + \tfrac{1}{2}s_2\right\}, \left\{\tfrac{3}{4}s_1 + \tfrac{1}{4}s_2\right\}\right\} \in \mathcal{P}_c\mathcal{P}_c\mathcal{D}S.$$

Taking the union yields

$$\mathsf{conv}\left\{s_0, \tfrac{1}{2}s_1 + \tfrac{1}{2}s_2, \tfrac{3}{4}s_1 + \tfrac{1}{4}s_2\right\} \in \mathcal{P}_c\mathcal{D}S,$$

which leads to the convex subset of distributions over outputs

$$\mathsf{conv}\left\{1, \tfrac{1}{2}\cdot 0 + \tfrac{1}{2}\cdot 1, \tfrac{3}{4}\cdot 0 + \tfrac{1}{4}\cdot 1\right\} \in \mathcal{P}_c\mathcal{D}[0,1].$$

Calculating the expected weights gives $\mathsf{conv}\{1, \frac{1}{2}, \frac{1}{4}\} \in \mathcal{P}_c[0,1]$, which has a minimum of $\frac{1}{4}$. One can show that on reading any word $u \in \Sigma^*$ the automaton outputs 2^{-n}, where n is the length of the longest sequence of a's occurring in u.

The semantics coming from max and min are highly symmetrical; in a sense, they are two representations of the same semantics.[5] Technically, we establish the following relation between the two semantics—this will be useful to avoid repeating proofs twice for each property.

Proposition 3.4. *Consider an NPA $\mathcal{A} = (S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ under the* min *semantics. Define $\gamma' : S \to [0,1]$ by $\gamma'(s) = 1 - \gamma(s)$, and consider the NPA $\mathcal{A}' = (S, s_0, \gamma', \{\tau_a\}_{a\in\Sigma})$ under the* max *semantics. Then $\mathcal{L}_{\mathcal{A}'}(u) = 1 - \mathcal{L}_{\mathcal{A}}(u)$ for all $u \in \Sigma^*$.*

Proof. We prove a stronger property by induction on u: for all $x \in \mathcal{P}_c\mathcal{D}S$ and $u \in \Sigma^*$, we have $l_{\mathcal{A}'}(x)(u) = 1 - l_{\mathcal{A}}(x)(u)$. This is sufficient because $\mathcal{A}$ and $\mathcal{A}'$ have the same initial state. We have

$$\begin{aligned}
& l_{\mathcal{A}'}(x)(\varepsilon) \\
&= (\mathsf{max} \circ \mathcal{P}_c\mathbb{E} \circ \mathcal{P}_c\mathcal{D}\gamma')(x) && \text{(Definition 2.15)} \\
&= (\mathsf{max} \circ \mathcal{P}_c\mathbb{E})\left(\left\{\lambda p.\sum_{s\in S, \gamma'(s)=p} d(s) \;\middle|\; d \in x\right\}\right) && \text{(definition of } \mathcal{P}_c\mathcal{D}) \\
&= \mathsf{max}\left(\left\{\sum_{p\in[0,1]} p \cdot \sum_{s\in S,\gamma'(s)=p} d(s) \;\middle|\; d \in x\right\}\right) && \text{(definition of } \mathcal{P}_c\mathbb{E}) \\
&= \mathsf{max}\left(\left\{\sum_{p\in[0,1]} p \cdot \sum_{s\in S,\gamma(s)=1-p} d(s) \;\middle|\; d \in x\right\}\right) && \text{(definition of } \gamma') \\
&= \mathsf{max}\left(\left\{\sum_{p\in[0,1]} (1-p) \cdot \sum_{s\in S,\gamma(s)=p} d(s) \;\middle|\; d \in x\right\}\right) \\
&= \mathsf{max}\left(\left\{\sum_{p\in[0,1]} (1-p) \cdot \mathcal{D}\gamma(d)(p) \;\middle|\; d \in x\right\}\right) \\
&= \mathsf{max}\left(\left\{1 - \sum_{p\in[0,1]} p \cdot \mathcal{D}\gamma(d)(p) \;\middle|\; d \in x\right\}\right) \\
&= 1 - \mathsf{min}\left(\left\{\sum_{p\in[0,1]} p \cdot \mathcal{D}\gamma(d)(p) \;\middle|\; d \in x\right\}\right) \\
&= 1 - (\mathsf{min} \circ \mathcal{P}_c\mathbb{E})(\{\mathcal{D}\gamma(d) \mid d \in x\}) && \text{(definition of } \mathcal{P}_c\mathbb{E})
\end{aligned}$$

[5] The max semantics is perhaps slightly preferable since it recovers standard nondeterministic finite automata when there is no probabilistic choice and the output weights are in $\{0,1\}$.

$$= 1 - (\min \circ \mathcal{P}_c\mathbb{E} \circ \mathcal{P}_c\mathcal{D}\gamma)(x) \qquad \text{(definition of } \mathcal{P}_c\text{)}$$

$$= 1 - l_{\mathcal{A}}(x)(\varepsilon) \qquad \text{(Definition 2.15).}$$

Furthermore,

$$\begin{aligned} l_{\mathcal{A}'}(x)(av) &= l_{\mathcal{A}'}\left(\left(\bigcup \circ \mathcal{P}_c\omega \circ \mathcal{P}_c\mathcal{D}\tau_a\right)(x)\right)(v) && \text{(Definition 2.15)} \\ &= 1 - l_{\mathcal{A}}\left(\left(\bigcup \circ \mathcal{P}_c\omega \circ \mathcal{P}_c\mathcal{D}\tau_a\right)(x)\right)(v) && \text{(induction hypothesis)} \\ &= 1 - l_{\mathcal{A}}(x)(av) && \text{(Definition 2.15).} \end{aligned}$$

□

3.3. Using the finite representation

In this section we will show how to compute the semantics of nondeterministic probabilistic automata using a finite representation. Note that each transition of the NPAs we consider goes to a potentially infinite set of distributions over states. However, this set is finitely generated and can thus be finitely presented. In order for NPAs to be used in practice it is essential to have this finite presentation extended to the automaton level. This requires simulating the determinization construction of Definition 2.14 on the finite presentation and proving a soundness result, which we will do in this section. We start by making the finite presentation explicit in the following definition.

Definition 3.5. A *finitary NPA* over a (finite) alphabet Σ is defined by a tuple $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$, where S is a finite set of *states*, $s_0 \in S$ is the *initial state*, $\gamma : S \to [0,1]$ is the *output function*, and $\tau_a : S \to \mathcal{P}_f\mathcal{D}S$ are the *transition functions* indexed by inputs $a \in \Sigma$.

To run these automata we need to determinize them, but there is no monad structure on the functor $\mathcal{P}_f\mathcal{D}$ [29–31]. However, we do know what the semantics should be, since a finitary NPA $\mathcal{A} = (S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ extends to the NPA $\hat{\mathcal{A}} = (S, s_0, \gamma, \{\overline{\tau}_a\}_{a\in\Sigma})$, where for each $a \in \Sigma$,

$$\hat{\tau}_a = S \xrightarrow{\tau_a} \mathcal{P}_f\mathcal{D}S \xrightarrow{\mathsf{conv}} \mathcal{P}_c\mathcal{D}S,$$

and this NPA can be determinized as $\mathcal{P}_c\mathcal{D}$ is a monad. Our plan now is to simulate the determinization of the extension NPA on the finitary NPA and show that the map $\mathsf{conv} : \mathcal{P}_f\mathcal{D}S \to \mathcal{P}_c\mathcal{D}S$ forms an automaton homomorphism between the two, which guarantees language equivalence (and hence soundness of the construction).

We mirror $\omega_A : \mathcal{D}\mathcal{P}_cA \to \mathcal{P}_cA$ for a convex set (A, α) in the finite case by the function $\psi_A : \mathcal{D}\mathcal{P}_fA \to \mathcal{P}_fA$ given by

$$\psi_A(d) = \{(\alpha \circ \mathcal{D}c')(d) \mid c' : \mathsf{supp}(d) \to A, \forall U \in \mathsf{supp}(d).\, c'(U) \in U\}.$$

This allows us to define a determinization that is not an instance of Definition 2.14 but has the crucial property having a finite description and being equivalent to the aforementioned infinite determinization construction.

Definition 3.6 *(Finitary NPAs determinization)*. The determinization of a finitary NPA $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$ is the automaton $(\mathcal{P}_f\mathcal{D}S, \{\delta(s_0)\}, \hat{\gamma}, \{\hat{\tau}_a\}_{a\in\Sigma})$, where

$$\hat{\gamma} = \mathcal{P}_f\mathcal{D}S \xrightarrow{\mathcal{P}_f\mathcal{D}\gamma} \mathcal{P}_f\mathcal{D}[0,1] \xrightarrow{\mathcal{P}_f\mathbb{E}} \mathcal{P}_f[0,1] \xrightarrow{\alpha} [0,1]$$

$$\hat{\tau}_a = \mathcal{P}_f\mathcal{D}S \xrightarrow{\mathcal{P}_f\mathcal{D}\tau_a} \mathcal{P}_f\mathcal{D}\mathcal{P}_f\mathcal{D}S \xrightarrow{\mathcal{P}_f\psi_{\mathcal{D}S}} \mathcal{P}_f\mathcal{P}_f\mathcal{D}S \xrightarrow{\bigcup} \mathcal{P}_f\mathcal{D}S.$$

and α can be either min or max.

The following results will be used to prove Theorem 3.12, which states that conv is an automaton homomorphism from the finitary determinization to the (convex) determinization obtained through Definition 2.14. This guarantees both automata accept the same language and therefore implies that the construction above is sound. The results needed in the proof of the soundness theorem include several properties about the support of a distribution (supp) and the convex closure of a set (conv):

1. supp is a natural transformation from $\mathcal{D}$ to $\mathcal{P}_f$ (Lemma 3.7).
2. supp is a monad morphism from $\mathcal{D}$ to $\mathcal{P}_f$ (Lemma 3.8).
3. conv is a monad morphism from $\mathcal{P}_f$ to $\mathcal{P}_c$ (up to the forgetful functor from the category of convex sets, Lemma 3.9).[6]
4. ω, the (affine) convex algebra structure on $\mathcal{P}_cA$, and its analogue for the finite powerset case—ψ_A—are compatible via conv (Lemma 3.10).
5. The output maps of the determinizations are compatible via conv (Lemma 3.11).

[6] For both of these monad morphisms we do not include the compatibility with the units. It would be trivial to extend the proofs, but these particular properties are not needed in the final theorem.

Lemma 3.7. *For each function $f : X \to Y$, the diagram below commutes.*

$$\begin{tikzcd}
\mathcal{D}X \arrow[r, "\mathcal{D}f"] \arrow[d, "\mathsf{supp}"'] & \mathcal{D}Y \arrow[d, "\mathsf{supp}"] \\
\mathcal{P}_f X \arrow[r, "\mathcal{P}_f f"] & \mathcal{P}_f Y
\end{tikzcd}$$

Proof. For each $d \in \mathcal{D}X$,

$$\begin{aligned}
(\mathsf{supp} \circ \mathcal{D}f)(d) &= \mathsf{supp}\left(\lambda y \in Y.\ \sum_{x\in X, f(x)=y} d(x)\right) && \text{(definition of } \mathcal{D}) \\
&= \{y \in Y \mid \exists x \in \mathsf{supp}(d).\ f(x) = y\} && \text{(definition of supp)} \\
&= \{f(x) \mid x \in \mathsf{supp}(d)\} \\
&= (\mathcal{P}_f f \circ \mathsf{supp})(d) && \text{(definition of } \mathcal{P}_f). \qquad \square
\end{aligned}$$

Lemma 3.8. *For each set X, the diagram below commutes.*

$$\begin{tikzcd}
\mathcal{D}\mathcal{D}X \arrow[r, "\mathsf{supp}"] \arrow[d, "m"'] & \mathcal{P}_f\mathcal{D}X \arrow[r, "\mathcal{P}_f\mathsf{supp}"] & \mathcal{P}_f\mathcal{P}_f X \arrow[d, "\bigcup"] \\
\mathcal{D}X \arrow[rr, "\mathsf{supp}"] & & \mathcal{P}_f X
\end{tikzcd}$$

Proof. For each $d \in \mathcal{D}\mathcal{D}X$,

$$\begin{aligned}
&\left(\bigcup \circ \mathcal{P}_f\mathsf{supp} \circ \mathsf{supp}\right)(d) \\
&= \bigcup\{\mathsf{supp}(e) \mid e \in \mathsf{supp}(d)\} && \text{(definition of } \mathcal{P}_f) \\
&= \{x \in \mathsf{supp}(e) \mid e \in \mathsf{supp}(d)\} && \text{(definition of } \textstyle\bigcup) \\
&= \{x \in X \mid \exists e \in \mathsf{supp}(d).\ x \in \mathsf{supp}(e)\} \\
&= \mathsf{supp}\left(\lambda x \in X.\ \sum_{e\in\mathcal{D}X} d(e)\cdot e(x)\right) && \text{(definition of supp)} \\
&= (\mathsf{supp} \circ m)(d) && \text{(definition of } m). \qquad \square
\end{aligned}$$

Lemma 3.9. *For each convex set A, the diagram below commutes.*

$$\begin{tikzcd}
\mathcal{P}_f\mathcal{P}_f A \arrow[r, "\mathcal{P}_f\mathsf{conv}"] \arrow[d, "\bigcup"'] & \mathcal{P}_f\mathcal{P}_c A \arrow[r, "\mathsf{conv}"] & \mathcal{P}_c\mathcal{P}_c A \arrow[d, "\bigcup"] \\
\mathcal{P}_f A \arrow[rr, "\mathsf{conv}"] & & \mathcal{P}_c A
\end{tikzcd}$$

Proof. We will first show that the diagram

$$\begin{tikzcd}
\mathcal{P}_f\mathcal{P}_f A \arrow[r, "\mathcal{P}_f\mathsf{dis}"] \arrow[d, "\bigcup"'] & \mathcal{P}_f\mathcal{P}_c\mathcal{D}A \arrow[r, "\mathsf{dis}"] & \mathcal{P}_c\mathcal{D}\mathcal{P}_c\mathcal{D}A \arrow[r, "\mathcal{P}_c\omega"] & \mathcal{P}_c\mathcal{P}_c\mathcal{D}A \arrow[d, "\bigcup"] \\
\mathcal{P}_f A \arrow[rrr, "\mathsf{dis}"] & & & \mathcal{P}_c\mathcal{D}A
\end{tikzcd} \tag{8}$$

commutes. Consider any $U \in \mathcal{P}_f\mathcal{P}_f A$. We define

$$X = \left(\mathsf{dis} \circ \bigcup\right)(U) = \left\{d \in \mathcal{D}A \mid \mathsf{supp}(d) \subseteq \bigcup U\right\},$$

rewritten by expanding the definition of dis, and

$$\begin{aligned}
&Y \\
&= \left(\bigcup \circ \mathcal{P}_c\omega \circ \mathsf{dis} \circ \mathcal{P}_f\mathsf{dis}\right)(U) \\
&\overset{①}{=} \left(\bigcup \circ \mathcal{P}_c\omega \circ \mathsf{dis}\right)(\{\mathsf{dis}(V) \mid V \in U\})
\end{aligned}$$

$$\overset{②}{=} \left(\bigcup \circ \mathcal{P}_c \omega\right)(\{e \in \mathcal{D}\mathcal{P}_c\mathcal{D}A \mid \mathsf{supp}(e) \subseteq \{\mathsf{dis}(V) \mid V \in U\}\})$$
$$\overset{③}{=} \bigcup\{\{(m \circ \mathcal{D}c)(e) \mid c \in \mathsf{choice}_{\mathcal{D}A}\} \mid e \in \mathcal{D}\mathcal{P}_c\mathcal{D}A, \mathsf{supp}(e) \subseteq \{\mathsf{dis}(V) \mid V \in U\}\}$$
$$= \{(m \circ \mathcal{D}c)(e) \mid c \in \mathsf{choice}_{\mathcal{D}A}, e \in \mathcal{D}\mathcal{P}_c\mathcal{D}A, \mathsf{supp}(e) \subseteq \{\mathsf{dis}(V) \mid V \in U\}\}.$$

Here ① applies $\mathcal{P}_f\mathsf{dis}$, ② expands the definition of the outer dis, ③ expands the definition of $\mathcal{P}_c$, and finally we take the union of the resulting set of sets. We will show that $X \subseteq Y$ and $Y \subseteq X$. Regarding the former inclusion, consider any $d \in \mathcal{D}A$ such that $\mathsf{supp}(d) \subseteq \bigcup U$. For each $V \in U$ we define $d_V \in \mathcal{D}A$ by

$$d_V(a) = \begin{cases} \frac{d(a)}{|\{V' \in U \mid a \in V'\}| \sum_{b \in V} \frac{d(b)}{|\{V' \in U \mid b \in V'\}|}} & \text{if } a \in V \\ 0 & \text{if } a \notin V. \end{cases}$$

This is indeed a distribution, as

$$\sum_{a \in V} \frac{d(a)}{|\{V' \in U \mid a \in V'\}| \sum_{b \in V} \frac{d(b)}{|\{V' \in U \mid b \in V'\}|}}$$
$$= \sum_{a \in V} \frac{d(a)}{|\{V' \in U \mid a \in V'\}|} \cdot \frac{1}{\sum_{b \in V} \frac{d(b)}{|\{V' \in U \mid b \in V'\}|}}$$
$$= \left(\sum_{a \in V} \frac{d(a)}{|\{V' \in U \mid a \in V'\}|}\right) \cdot \frac{1}{\sum_{b \in V} \frac{d(b)}{|\{V' \in U \mid b \in V'\}|}}$$
$$= 1.$$

Now we define $c : \mathcal{P}_c\mathcal{D}A \to \mathcal{D}A$ for $W \in \mathcal{P}_c\mathcal{D}A$, which is nonempty and thus contains an element $w \in W$, by

$$c(W) = \begin{cases} d_V & \text{if } W = \mathsf{dis}(V) \text{ for some } V \in U \\ w & \text{otherwise.} \end{cases}$$

Note that $c \in \mathsf{choice}_{\mathcal{D}A}$. We further define $e \in \mathcal{D}\mathcal{P}_c\mathcal{D}A$ by

$$e(W) = \begin{cases} \sum_{a \in V} \frac{d(a)}{|\{V' \in U \mid a \in V'\}|} & \text{if } W = \mathsf{dis}(V) \text{ for some } V \in U \\ 0 & \text{otherwise.} \end{cases}$$

By definition, $\mathsf{supp}(e) \subseteq \{\mathsf{dis}(V) \mid V \in U\}$. It follows from the definitions of e and d_V that for all $V \in U$ and $a \in V$,

$$(e \circ \mathsf{dis})(V) \cdot d_V(a) = \frac{d(a)}{|\{V' \in U \mid a \in V'\}|}. \tag{9}$$

Thus, for all $a \in A$,

$$\begin{aligned}
&(m \circ \mathcal{D}c)(e)(a) \\
&= \sum_{f \in \mathcal{D}A} \mathcal{D}c(e)(f) \cdot f(a) && \text{(definition of } m\text{)} \\
&= \sum_{f \in \mathcal{D}A, W \in \mathcal{P}_c\mathcal{D}A, c(W) = f} e(W) \cdot f(a) && \text{(definition of } \mathcal{D}\text{)} \\
&= \sum_{W \in \mathcal{P}_c\mathcal{D}A} e(W) \cdot c(W)(a) \\
&= \sum_{V \in U} (e \circ \mathsf{dis})(V) \cdot (c \circ \mathsf{dis})(V)(a) && \text{(definition of } e\text{)} \\
&= \sum_{V \in U} (e \circ \mathsf{dis})(V) \cdot d_V(a) && \text{(definition of } c\text{)} \\
&= \sum_{V \in U, a \in V} (e \circ \mathsf{dis})(V) \cdot d_V(a) && \text{(definition of } d_V\text{)} \\
&= \sum_{V \in U, a \in V} \frac{d(a)}{|\{V' \in U \mid a \in V'\}|} && (9) \\
&= d(a),
\end{aligned}$$

so $(m \circ \mathcal{D}c)(e) = d$, which concludes the first inclusion.

For the inclusion $Y \subseteq X$, consider any $c \in \mathsf{choice}_{\mathcal{D}A}$ and $e \in \mathcal{D}\mathcal{P}_c\mathcal{D}A$ such that $\mathsf{supp}(e) \subseteq \{\mathsf{dis}(V) \mid V \in U\}$. We need to show that $\mathsf{supp}((m \circ \mathcal{D}c)(e)) \subseteq \bigcup U$. We have

$$\begin{aligned}
&\mathsf{supp}((m \circ \mathcal{D}c)(e)) \\
&= \left(\bigcup \circ \mathcal{P}_f\mathsf{supp}\right)(\mathsf{supp}(\mathcal{D}c(e))) && \text{(Lemma 3.8)} \\
&= \left(\bigcup \circ \mathcal{P}_f\mathsf{supp} \circ \mathcal{P}_f c\right)(\mathsf{supp}(e)) && \text{(Lemma 3.7)} \\
&\subseteq \left(\bigcup \circ \mathcal{P}_f\mathsf{supp} \circ \mathcal{P}_f c\right)(\{\mathsf{dis}(V) \mid V \in U\}) && \text{(monotonicity)} \\
&= \bigcup\{(\mathsf{supp} \circ c \circ \mathsf{dis})(V) \mid V \in U\} && \text{(definition of } \mathcal{P}_f\mathsf{supp} \circ \mathcal{P}_f c\text{)} \\
&= \bigcup\{W \mid W \subseteq V \in U\} && \text{(definition of dis)} \\
&\subseteq \bigcup\{V \mid V \in U\} \\
&= \bigcup U.
\end{aligned}$$

Note that both $\bigcup$ and any $\mathcal{P}_f f$ for any function f are monotone. This concludes commutativity of (8).

Below we show commutativity of the original diagram, using the factorization of conv through dis.

$$\begin{tikzcd}
\mathcal{P}_f\mathcal{P}_f A \arrow[rrr, "\bigcup"] \arrow[d, "\mathcal{P}_f\mathsf{dis}"'] \arrow[drrr, phantom, "(8)"] & & & \mathcal{P}_f A \arrow[d, "\mathsf{dis}"] \\
\mathcal{P}_f\mathcal{P}_c\mathcal{D}A \arrow[r, "\mathsf{dis}"] \arrow[d, "\mathcal{P}_f\mathcal{P}_c\alpha"'] \arrow[dr, phantom, "\textcircled{1}"] & \mathcal{P}_c\mathcal{D}\mathcal{P}_c\mathcal{D}A \arrow[r, "\mathcal{P}_c\omega"] \arrow[d, "\mathcal{P}_c\mathcal{D}\mathcal{P}_c\alpha"'] \arrow[dr, phantom, "\textcircled{2}"] & \mathcal{P}_c\mathcal{P}_c\mathcal{D}A \arrow[r, "\bigcup"] \arrow[d, "\mathcal{P}_c\mathcal{P}_c\alpha"] \arrow[dr, phantom, "\textcircled{3}"] & \mathcal{P}_c\mathcal{D}A \arrow[d, "\mathcal{P}_c\alpha"] \\
\mathcal{P}_f\mathcal{P}_c A \arrow[r, "\mathsf{dis}"] & \mathcal{P}_c\mathcal{D}\mathcal{P}_c A \arrow[r, "\mathcal{P}_c\omega"] & \mathcal{P}_c\mathcal{P}_c A \arrow[r, "\bigcup"] & \mathcal{P}_c A
\end{tikzcd}$$

$\textcircled{1}$ Lemma 2.11 $\textcircled{2}$ naturality of ω $\textcircled{3}$ naturality of $\bigcup$ □

Lemma 3.10. *For each convex set (A, α), the diagram below commutes.*

$$\begin{tikzcd}
\mathcal{D}\mathcal{P}_f A \arrow[r, "\mathcal{D}\mathsf{conv}"] \arrow[d, "\psi_A"'] & \mathcal{D}\mathcal{P}_c A \arrow[d, "\omega"] \\
\mathcal{P}_f A \arrow[r, "\mathsf{conv}"] & \mathcal{P}_c A
\end{tikzcd}$$

Proof. Note that ψ_A can be written as

$$\psi_A = \mathcal{D}\mathcal{P}_f A \xrightarrow{\rho} \mathcal{P}_f\mathcal{D}A \xrightarrow{\mathcal{P}_f\alpha} \mathcal{P}_f A, \tag{10}$$

where

$$\rho(d) = \{d_{c'} \mid c' \in C\} \qquad C = \{c' : \mathsf{supp}(d) \to A, \forall U \in \mathsf{supp}(d).\, c'(U) \in U\}$$

and $d_{c'} \in \mathcal{D}A$ for any $c' \in C$ is given by

$$d_{c'}(a) = \sum_{U \in \mathsf{supp}(d), c'(U) = a} d(U).$$

We will first show that the diagram

$$\begin{tikzcd}
\mathcal{D}\mathcal{P}_f A \arrow[r, "\mathcal{D}\mathsf{dis}"] \arrow[d, "\rho"'] & \mathcal{D}\mathcal{P}_c\mathcal{D}A \arrow[d, "\omega"] \\
\mathcal{P}_f\mathcal{D}A \arrow[r, "\mathsf{conv}"] & \mathcal{P}_c\mathcal{D}A
\end{tikzcd} \tag{11}$$

commutes. Given $d \in \mathcal{D}\mathcal{P}_f A$, let us define the sets

$$\begin{aligned}
X &= (\omega \circ \mathcal{D}\mathsf{dis})(d) \\
&\overset{\textcircled{1}}{=} \{(m \circ \mathcal{D}(c \circ \mathsf{dis}))(d) \mid c \in \mathsf{choice}_{\mathcal{D}A}\} \\
&\overset{\textcircled{2}}{=} \left\{ \lambda a \in A.\, \sum_{f \in \mathcal{D}A} \mathcal{D}(c \circ \mathsf{dis})(d)(f) \cdot f(a) \,\middle|\, c \in \mathsf{choice}_{\mathcal{D}A} \right\}
\end{aligned}$$

$$\begin{aligned}
&\overset{③}{=} \left\{ \lambda a \in A.\ \sum_{f\in\mathcal{D}A, U\in\mathsf{supp}(d), (c\circ\mathsf{dis})(U)=f} d(U)\cdot f(a) \,\middle|\, c \in \mathsf{choice}_{\mathcal{D}A} \right\} \\
&= \left\{ \lambda a \in A.\ \sum_{U\in\mathsf{supp}(d)} d(U)\cdot (c\circ\mathsf{dis})(U)(a) \,\middle|\, c \in \mathsf{choice}_{\mathcal{D}A} \right\}
\end{aligned}$$

and

$$\begin{aligned}
Y &= (\mathsf{conv}\circ\rho)(d) \\
&\overset{④}{=} \{ m(e) \mid e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \} \\
&\overset{②}{=} \left\{ \lambda a \in A.\ \sum_{f\in\mathcal{D}A} e(f)\cdot f(a) \,\middle|\, e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \right\} \\
&= \left\{ \lambda a \in A.\ \sum_{f\in\rho(d)} e(f)\cdot f(a) \,\middle|\, e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \right\} \\
&\overset{⑤}{=} \left\{ \lambda a \in A.\ \sum_{f\in\mathcal{D}A, \exists c'\in C.\, f=d_{c'}} e(f)\cdot f(a) \,\middle|\, e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \right\} \\
&= \left\{ \lambda a \in A.\ \sum_{f\in\mathcal{D}A, c'\in C, f=d_{c'}} \frac{e(f)\cdot f(a)}{|\{c''\in C \mid d_{c''}=f\}|} \,\middle|\, e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \right\} \\
&= \left\{ \lambda a \in A.\ \sum_{c'\in C} \frac{e(d_{c'})\cdot d_{c'}(a)}{|\{c''\in C \mid d_{c''}=d_{c'}\}|} \,\middle|\, e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \right\} \\
&\overset{⑥}{=} \left\{ \lambda a \in A.\ \sum_{c'\in C, U\in\mathsf{supp}(d), c'(U)=a} \frac{e(d_{c'})\cdot d(U)}{|\{c''\in C \mid d_{c''}=d_{c'}\}|} \,\middle|\, e \in \mathcal{D}\mathcal{D}A, \mathsf{supp}(e) \subseteq \rho(d) \right\}.
\end{aligned}$$

Here ①, ②, ③, and ④ expand the definition of ω, m, $\mathcal{D}$, and conv, respectively. Step ⑤ uses the definition of ρ, and ⑥ expands the definition of $d_{c'}$. We need to show $X \subseteq Y$ and $Y \subseteq X$. For the former, consider any $c \in \mathsf{choice}_{\mathcal{D}A}$. We define $e \in \mathcal{D}\mathcal{D}A$ by

$$e(f) = \sum_{c'\in C, d_{c'}=f} \ \prod_{U\in\mathsf{supp}(d)} (c\circ\mathsf{dis})(U)(c'(U)).$$

To show that e is a distribution, we define for any $Z \in \mathcal{P}_f A$ such that $Z \subseteq \mathsf{supp}(d)$,

$$C_Z = \{c' : Z \to A, \forall U \in Z. c'(U) \in U\}.$$

We will show that

$$\sum_{c'\in C_Z} \prod_{U\in Z} (c\circ\mathsf{dis})(U)(c'(U)) = 1. \tag{12}$$

First suppose that Z is a singleton $\{U\}$, and note that $C_{\{U\}} \cong U$ via the element picked by the choice function. Thus,

$$\sum_{c'\in C_{\{U\}}} (c\circ\mathsf{dis})(U)(c'(U)) = \sum_{a\in U} (c\circ\mathsf{dis})(U)(a) = 1.$$

Now assume that $Z = Z' \uplus \{U\}$ with $|Z| \geq 2$ and that we have

$$\sum_{c'\in C_{Z'}} \prod_{V\in Z'} (c\circ\mathsf{dis})(V)(c'(V)) = 1.$$

Observe that $C_Z \cong C_{Z'} \times U$ via the element picked from U. Then

$$\begin{aligned}
&\sum_{c'\in C_{Z'\cup\{U\}}} \ \prod_{V\in Z'\cup\{U\}} (c\circ\mathsf{dis})(V)(c'(V)) \\
&= \sum_{c'\in C_{Z'}, a\in U} (c\circ\mathsf{dis})(U)(a) \cdot \prod_{V\in Z'} (c\circ\mathsf{dis})(V)(c'(V))
\end{aligned}$$

$$= \sum_{c' \in C_{Z'}} \prod_{V \in Z'} (c \circ \mathsf{dis})(V)(c'(V))$$

$$= 1,$$

using the induction hypothesis in the last step. Finally, to see that e is a distribution, note that $C = C_{\mathsf{supp}(d)}$ and thus

$$\sum_{f \in \mathcal{D}A, c' \in C, d_{c'} = f} \prod_{U \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(U)(c'(U))$$

$$= \sum_{c' \in C} \prod_{U \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(U)(c'(U))$$

$$= 1.$$

We are now ready to show that this definition of e induces the right element of Y. For all $a \in A$,

$$\sum_{c' \in C, U \in \mathsf{supp}(d), c'(U) = a} \frac{e(d_{c'}) \cdot d(U)}{|\{c'' \in C \mid d_{c''} = d_{c'}\}|}$$

$$= \sum_{\substack{c' \in C, U \in \mathsf{supp}(d), \\ c'(U) = a, c'' \in C, d_{c''} = d_{c'}}} \frac{\prod_{V \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(V)(c''(V))}{|\{c''' \in C \mid d_{c'''} = d_{c'}\}|} \cdot d(U)$$

$$= \sum_{\substack{c' \in C, \\ c'' \in C, d_{c''} = d_{c'}}} \frac{\prod_{V \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(V)(c''(V))}{|\{c''' \in C \mid d_{c'''} = d_{c'}\}|} \cdot \sum_{\substack{U \in \mathsf{supp}(d), \\ c'(U) = a}} d(U)$$

$$= \sum_{\substack{c' \in C, \\ c'' \in C, d_{c''} = d_{c'}}} \frac{\prod_{V \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(V)(c''(V))}{|\{c''' \in C \mid d_{c'''} = d_{c'}\}|} \cdot \sum_{\substack{U \in \mathsf{supp}(d), \\ c''(U) = a}} d(U)$$

$$= \sum_{\substack{c' \in C, c'' \in C, d_{c''} = d_{c'}, \\ U \in \mathsf{supp}(d), c''(U) = a}} d(U) \cdot \frac{\prod_{V \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(V)(c''(V))}{|\{c''' \in C \mid d_{c'''} = d_{c'}\}|}$$

$$= \sum_{\substack{c'' \in C, \\ U \in \mathsf{supp}(d), c''(U) = a}} d(U) \cdot \prod_{V \in \mathsf{supp}(d)} (c \circ \mathsf{dis})(V)(c''(V))$$

$$= \sum_{\substack{c'' \in C_{\mathsf{supp}(d) \setminus \{U\}}, \\ U \in \mathsf{supp}(d)}} d(U) \cdot (c \circ \mathsf{dis})(U)(a) \cdot \prod_{V \in \mathsf{supp}(d) \setminus \{U\}} (c \circ \mathsf{dis})(V)(c''(V))$$

$$= \sum_{U \in \mathsf{supp}(d)} d(U) \cdot (c \circ \mathsf{dis})(U)(a).$$

Here we expanded the definition of e in the first step and applied (12) in the last step.

Regarding the other inclusion $Y \subseteq X$, consider any $e \in \mathcal{D}\mathcal{D}A$ such that $\mathsf{supp}(e) \subseteq \{d_{c'} \mid c' \in C\}$. We define $c : \mathcal{P}_c\mathcal{D}A \to \mathcal{D}A$ by

$$c(V)(a) = \sum_{U \in \mathsf{supp}(d), V = \mathsf{dis}(U)} \sum_{c' \in C, c'(U) = a} \frac{e(d_{c'})}{|\{c'' \in C \mid d_{c''} = d_{c'}\}|}$$

To see that $c(V)$ is a distribution for any $V \in \mathcal{P}_c\mathcal{D}A$, note that

$$\sum_{a \in A, c' \in C, c'(U) = a} \frac{e(d_{c'})}{|\{c'' \in C \mid d_{c''} = d_{c'}\}|} = \sum_{c' \in C} \frac{e(d_{c'})}{|\{c'' \in C \mid d_{c''} = d_{c'}\}|} = 1.$$

Furthermore, for all $a \in A$,

$$\sum_{U \in \mathsf{supp}(d)} d(U) \cdot (c \circ \mathsf{dis})(U)(a) = \sum_{U \in \mathsf{supp}(d), c' \in C, c'(U) = a} \frac{e(d_{c'}) \cdot d(U)}{|\{c'' \in C \mid d_{c''} = d_{c'}\}|}$$

by the definition of c, so this defines the right element of X. This concludes the commutativity of (11), which completes the diagram below.

$$\begin{tikzcd}
\mathcal{D}\mathcal{P}_f A \arrow[rrrr, "\mathcal{D}\mathsf{conv}"] \arrow[rrd, "\mathcal{D}\mathsf{dis}"] \arrow[rd, "\rho"'] \arrow[dddd, "\psi"'] & & (2) & & \mathcal{D}\mathcal{P}_c A \arrow[dddd, "\omega"] \\
 & (11) & \mathcal{D}\mathcal{P}_c\mathcal{D}A \arrow[rru, "\mathcal{D}\mathcal{P}_c\alpha"'] \arrow[d, "\omega"] & & \\
(10) & \mathcal{P}_f\mathcal{D}A \arrow[r, "\mathsf{conv}"] \arrow[ldd, "\mathcal{P}_f\alpha"'] & \mathcal{P}_c\mathcal{D}A \arrow[rrdd, "\mathcal{P}_c\alpha"] & \textcircled{2} & \\
 & & \textcircled{1} & & \\
\mathcal{P}_f A \arrow[rrrr, "\mathsf{conv}"] & & & & \mathcal{P}_c A
\end{tikzcd}$$

① Lemma 2.12 ② naturality of ω $\square$

Lemma 3.11. *The diagrams below commute.*

$$\begin{tikzcd}
\mathcal{P}_f[0,1] \arrow[r, "\mathsf{conv}"] \arrow[rd, "\mathsf{min}"'] & \mathcal{P}_c[0,1] \arrow[d, "\mathsf{min}"] \\
 & {[0,1]}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{P}_f[0,1] \arrow[r, "\mathsf{conv}"] \arrow[rd, "\mathsf{max}"'] & \mathcal{P}_c[0,1] \arrow[d, "\mathsf{max}"] \\
 & {[0,1]}
\end{tikzcd}$$

Proof. We only show the case for min, as the one for max is analogous. For each $U \in \mathcal{P}_f[0,1]$, we have

$$\begin{aligned}
(\mathsf{min} \circ \mathsf{conv})(U) &= \mathsf{min}(\{\mathbb{E}(d) \mid d \in \mathcal{D}([0,1]), \mathsf{supp}(d) \subseteq U\}) \\
&= \mathsf{min}\left(\left\{ \sum_{a\in\mathsf{supp}(d)} d(a)\cdot a \;\middle|\; d \in [0,1], \mathsf{supp}(d) \subseteq U \right\}\right) \\
&= \mathsf{min}(U),
\end{aligned}$$

where in the first two steps we expand the definitions of $\mathsf{conv}(U)$ and $\mathbb{E}(d)$, respectively, and for the last step we note that to minimize the sum, the distribution d should be chosen so as to concentrate all probability mass at the minimum element of U. $\square$

We are now ready to prove the main result of this section: soundness of the finitary version of the determinization construction.

Theorem 3.12 *(Soundness of finitary determinization). Given any finitary NPA $(S, s_0, \gamma, \{\tau_a\}_{a\in\Sigma})$, the map $\mathsf{conv} : \mathcal{P}_f\mathcal{D}S \to \mathcal{P}_c\mathcal{D}S$ is an automaton homomorphism from its determinization to the determinization of its convex extension.*

Proof. Noting $\mathsf{conv} \circ \{-\} = \{-\}$, the initial state is trivially preserved. For the output maps, we have commutativity of the following diagram.

$$\begin{tikzcd}
\mathcal{P}_f\mathcal{D}S \arrow[r, "\mathcal{P}_f\mathcal{D}\gamma"] \arrow[d, "\mathsf{conv}"'] \arrow[rd, phantom, "\textcircled{1}"] & \mathcal{P}_f\mathcal{D}[0,1] \arrow[r, "\mathcal{P}_f\mathbb{E}"] \arrow[d, "\mathsf{conv}"'] \arrow[rd, phantom, "\textcircled{1}"] & \mathcal{P}_f[0,1] \arrow[d, "\mathsf{conv}"'] \arrow[rd, "\alpha"] \arrow[rd, phantom, "\textcircled{2}"'] & \\
\mathcal{P}_c\mathcal{D}S \arrow[r, "\mathcal{P}_c\mathcal{D}\gamma"] & \mathcal{P}_c\mathcal{D}[0,1] \arrow[r, "\mathcal{P}_c\mathbb{E}"] & \mathcal{P}_c[0,1] \arrow[r, "\alpha"] & {[0,1]}
\end{tikzcd}$$

① Lemma 2.12 ② Lemma 3.11

To conclude, conv commutes with each of the transition functions for $a \in \Sigma$.

$$\begin{tikzcd}
\mathcal{P}_f\mathcal{D}S \arrow[r, "\mathcal{P}_f\mathcal{D}\tau_a"] \arrow[dd, "\mathsf{conv}"] \arrow[rdd, phantom, "\textcircled{1}"] & \mathcal{P}_f\mathcal{D}\mathcal{P}_f\mathcal{D}S \arrow[rr, "\mathcal{P}_f\psi"] \arrow[rd, "\mathcal{P}_f\mathcal{D}\mathsf{conv}"] \arrow[dd, "\mathsf{conv}"] \arrow[rdd, phantom, "\textcircled{1}"] & \textcircled{2} & \mathcal{P}_f\mathcal{P}_f\mathcal{D}S \arrow[r, "\bigcup"] \arrow[d, "\mathcal{P}_f\mathsf{conv}"] \arrow[rdd, phantom, "\textcircled{3}"] & \mathcal{P}_f\mathcal{D}S \arrow[dd, "\mathsf{conv}"'] \\
 & & \mathcal{P}_f\mathcal{D}\mathcal{P}_c\mathcal{D}S \arrow[r, "\mathcal{P}_f\omega"] \arrow[d, "\mathsf{conv}"] \arrow[rd, phantom, "\textcircled{1}"] & \mathcal{P}_f\mathcal{P}_c\mathcal{D}S \arrow[d, "\mathsf{conv}"] & \\
\mathcal{P}_c\mathcal{D}S \arrow[r, "\mathcal{P}_c\mathcal{D}\tau_a"] & \mathcal{P}_c\mathcal{D}\mathcal{P}_f\mathcal{D}S \arrow[r, "\mathcal{P}_c\mathcal{D}\mathsf{conv}"] & \mathcal{P}_c\mathcal{D}\mathcal{P}_c\mathcal{D}S \arrow[r, "\mathcal{P}_c\omega"] & \mathcal{P}_c\mathcal{P}_c\mathcal{D}S \arrow[r, "\bigcup"] & \mathcal{P}_c\mathcal{D}S
\end{tikzcd}$$

① Lemma 2.12 ② Lemma 3.10 ③ Lemma 3.9 $\square$

4. Expressive power of NPAs

Our convex language semantics for NPAs coincides with the standard semantics for DPAs when all convex sets in the transition functions are singleton sets. In this section, we show that NPAs are in fact strictly more expressive than DPAs. We give two results. First, we exhibit a concrete language over a binary alphabet that is recognizable by a NPA, but not recognizable by any DPA. This argument uses elementary facts about weighted languages, and actually shows that NPAs are strictly more expressive than weighted finite automata (WFAs). A WFA (over a field $\mathbb{F}$) is a T-automaton over the free vector space monad V (see Example 2.6). More concretely, a WFA is a finite set of states S together with two linear maps $o\colon S \to \mathbb{F}$ and $t\colon S \to V(S)^A$ and an initial vector $i \in V(S)$. A WFA accepts weighted languages $\mathbb{F}^{A^*}$, as per Definition 2.15.

Next, we separate NPAs and DPAs over a unary alphabet. This argument is substantially more technical, relying on deeper results from number theory about linear recurrence sequences.

4.1. Separating NPAs and DPAs: binary alphabet

Consider the language $\mathcal{L}_a\colon \{a,b\}^* \to [0,1]$ by $\mathcal{L}_a(u) = 2^{-n}$, where n is the length of the longest sequence of a's occurring in u. Recall that this language is accepted by the NPA (4) using the min algebra.

Theorem 4.1. *NPAs are more expressive than DPAs. Specifically, there is no DPA, or even WFA, accepting $\mathcal{L}_a$.*

Proof. Assume there exists a WFA accepting $\mathcal{L}_a$, and let $l(u)$ for $u \in \{a,b\}^*$ be the language of the linear combination of states reached after reading the word u. We will show that the languages $l(a^n b)$ for $n \in \mathbb{N}$ are linearly independent. Since the function that assigns to each linear combination of states its accepted language is a linear map, this implies that the set of linear combinations of states of the WFA is a vector space of infinite dimension, a contradiction.

The proof is by induction on a natural number m. Assume that for all natural numbers $i \le m$ the languages $l(a^i b)$ are linearly independent. For all $i \le m$ we have $l(a^i b)(a^m) = 2^{-m}$ and $l(a^i b)(a^{m+1}) = 2^{-m-1}$; however, $l(a^{m+1} b)(a^m) = l(a^{m+1} b)(a^{m+1}) = 2^{-m-1}$. If $l(a^{m+1} b)$ is a linear combination of the languages $l(a^i b)$ for $i \le m$, then there are constants $c_1, \dots, c_m \in \mathbb{R}$ such that in particular

$$(c_1 + \cdots + c_m)2^{-m} = 2^{-m-1} \qquad \text{and} \qquad (c_1 + \cdots + c_m)2^{-m-1} = 2^{-m-1}.$$

These equations cannot be satisfied. Therefore, for all natural numbers $i \le m+1$ the languages $l(a^i b)$ are linearly independent. We conclude by induction that for all $m \in \mathbb{N}$ the languages $l(a^i b)$ for $i \le m$ are linearly independent, which implies that all languages $l(a^n b)$ for $n \in \mathbb{N}$ are linearly independent. □

A similar argument works for NPAs under the max algebra semantics—one can easily repeat the argument in the above theorem for the language accepted by the NPA resulting from applying Proposition 3.4 to the NPA (4).

4.2. Separating NPAs and DPAs: unary alphabet

We now turn to the unary case. A weighted language over a unary alphabet can be represented by a sequence $\langle u_i \rangle = u_0, u_1, \dots$ of real numbers. We will give such a language that is recognizable by a NPA but not recognizable by any WFA (and in particular, any DPA) using results on *linear recurrence sequences*, an established tool for studying unary weighted languages.

We begin with some mathematical preliminaries. A sequence of real numbers $\langle u_i \rangle$ is a *linear recurrence sequence* (LRS) if for some integer $k \in \mathbb{N}$ (the *order*), constants $u_0, \dots, u_{k-1} \in \mathbb{R}$ (the *initial conditions*), and coefficients $b_0, \dots, b_{k-1} \in \mathbb{R}$, we have

$$u_{n+k} = b_{k-1}u_{n-1} + \cdots + b_0 u_n$$

for every $n \in \mathbb{N}$. A well-known example of an LRS is the *Fibonacci sequence*, an order-2 LRS satisfying the recurrence $f_{n+2} = f_{n+1} + f_n$. Another example of an LRS is any constant sequence, i.e., $\langle u_i \rangle$ with $u_i = c$ for all i.

Linear recurrence sequences are closed under linear combinations: for any two LRS $\langle u_i \rangle, \langle v_i \rangle$ and constants $\alpha, \beta \in \mathbb{R}$, the sequence $\langle \alpha u_i + \beta v_i \rangle$ is again an LRS (possibly of larger order). We will use one important theorem about LRSs. See the monograph by Everest et al. [32] for details.

Theorem 4.2 *(Skolem-Mahler-Lech). If $\langle u_i \rangle$ is an LRS, then its* zero set *$\{i \in \mathbb{N} \mid u_i = 0\}$ is the union of a finite set along with finitely many arithmetic progressions (i.e., sets of the form $\{p + kn \mid n \in \mathbb{N}\}$ with $k \neq 0$).*

This is a celebrated result in number theory and not at all easy to prove. To make the connection to probabilistic and weighted automata, we will use two results. The first proposition follows from the Cayley-Hamilton Theorem.

Proposition 4.3 *(see, e.g., [25]). Let $\mathcal{L}$ be a weighted unary language recognizable by a weighted automaton W. Then the sequence of weights $\langle u_i \rangle$ with $u_i = \mathcal{L}(a^i)$ is an LRS, where the order is at most the number of states in W.*

While not every LRS can be recognized by a DPA, it is known that DPAs can recognize a weighted language encoding the sign of a given LRS.

Theorem 4.4 *(Akshay, et al. [33, Theorem 3, Corollary 4]). Given any LRS $\langle u_i \rangle$, there exists a stochastic matrix M such that*

$$u_n \ge 0 \iff u^T M^n v \ge 1/4$$

for all n, where $u = (1, 0, \ldots, 0)$ and $v = (0, 1, 0, \ldots, 0)$. Equality holds on the left if and only if it holds on the right. The language $\mathcal{L}(a^n) = u^T M^n v$ is recognizable by a DPA with input vector u, output vector v, and transition matrix M (Remark 2.17). If the LRS is rational, M can be taken to be rational as well.

We are now ready to separate NPAs and WFAs over a unary alphabet.

Theorem 4.5. *There is a language over a unary alphabet that is recognizable by an NPA but not by any WFA (and in particular any DPA).*

Proof. We will work in the complex numbers $\mathbb{C}$, with i being the positive square root of -1 as usual. Let $a, b \in \mathbb{Q}$ be nonzero such that $z \triangleq a + bi$ is on the unit circle in $\mathbb{C}$, for instance $a = 3/5, b = 4/5$ so that $|a + bi| = a^2 + b^2 = 1$. Let $\bar{z} = a - bi$ denote the complex conjugate of z and let $\mathrm{Re}(z)$ denote the real part of a complex number. It is possible to show that z is not a root of unity, i.e., $z^k \neq 1$ for all $k \in \mathbb{N}$. Let $\langle x_n \rangle$ be the sequence $x_n \triangleq (z^n + \bar{z}^n)/2 = \mathrm{Re}(z^n)$. By direct calculation, this sequence has imaginary part zero and satisfies the recurrence

$$x_{n+2} = 2a x_{n+1} - (a^2 + b^2) x_n$$

with $x_0 = 1$ and $x_1 = a$, so $\langle x_n \rangle$ is an order-2 rational LRS. By Theorem 4.4, there exists a stochastic matrix M and non-negative vectors u, v such that

$$x_n \geq 0 \iff u^T M^n v \geq 1/4$$

for all n, where equality holds on the left if and only if equality holds on the right. Note that $x_n = \mathrm{Re}(z^n) \neq 0$ since z is not a root of unity (so in particular $z^n \neq \pm i$), hence equality never holds on the right. Letting $\langle y_n \rangle$ be the sequence $y_n = u^T M^n v$, the (unary) language with weights $\langle y_n \rangle$ is recognized by the DPA with input u, output v and transition matrix M. Furthermore, the constant sequence $\langle 1/4 \rangle$ is recognizable by a DPA.

Now we define a sequence $\langle w_n \rangle$ with $w_n = \max(y_n, 1/4)$. Since $\langle y_n \rangle$ and $\langle 1/4 \rangle$ are recognizable by DPAs, $\langle w_n \rangle$ is recognizable by an NPA whose initial state nondeterministically chooses between the two DPAs (see Remark 2.17). Suppose for the sake of contradiction that it is also recognizable by a WFA. Then $\langle w_n \rangle$ is an LRS (by Proposition 4.3) and hence so is $\langle t_n \rangle$ with $t_n = w_n - y_n$. If we now consider the zero set

$$\begin{aligned}
S &= \{n \in \mathbb{N} \mid t_n = 0\} \\
&= \{n \in \mathbb{N} \mid y_n > 1/4\} && (y_n \neq 1/4) \\
&= \{n \in \mathbb{N} \mid x_n > 0\} && (\text{Theorem } 4.4) \\
&= \{n \in \mathbb{N} \mid \mathrm{Re}(z^n) > 0\} && (\text{by definition}),
\end{aligned}$$

Theorem 4.2 implies that S is the union of a finite set of indices and along with a finite number of arithmetic progressions. Note that S cannot be finite—in the last line, z^n is dense in the unit circle since z is not a root of unity—so there must be at least one arithmetic progression $\{p + kn \mid n \in \mathbb{N}\}$. Letting $\langle r_n \rangle$ be

$$r_n = (z^p \cdot (z^k)^n + \bar{z}^p \cdot (\bar{z}^k)^n)/2 = \mathrm{Re}(z^p \cdot (z^k)^n) = x_{p+kn},$$

we have $p + kn \in S$, so $r_n > 0$ for all $n \in \mathbb{N}$, but this is impossible since it is dense in $[-1, 1]$ (because z^k is not a root of unity for $k \neq 0$, so $z^p \cdot (z^k)^n$ is dense in the unit circle). Hence, the unary weighted language $\langle w_n \rangle$ can be recognized by an NPA but not by a WFA. $\square$

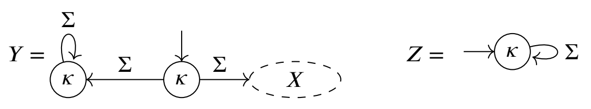
5. Checking language equivalence of NPAs

Now that we have a coalgebraic model for NPA, a natural question is whether there is a procedure to check language equivalence of NPAs. We will show that language equivalence of NPAs is undecidable by reduction from the *threshold problem* on DPAs. Nevertheless, we can define a metric on the set of languages recognized by NPAs to measure their similarity. While this metric cannot be computed exactly, it can be approximated to any given precision.

5.1. Undecidability and hardness

Theorem 5.1. *Equivalence of NPAs is undecidable when $|\Sigma| \geq 2$ and the $\mathcal{P}_c\mathcal{D}$-algebra on $[0, 1]$ extends the usual $\mathcal{D}$-algebra on $[0, 1]$.*

Proof. Let X be a DPA and $\kappa \in [0, 1]$. We define NPAs Y and Z as follows:



Here the node labeled X represents a copy of the automaton X—the transition into X goes into the initial state of X. Note that the edges are labeled by Σ to indicate a transition for every element of Σ. We see that $\mathcal{L}_Y(\varepsilon) = \kappa = \mathcal{L}_Z(\varepsilon)$ and (for α either min or max, as follows from Corollary 3.3)

$$\mathcal{L}_Y(av) = (\alpha \circ \mathsf{conv})(\{\kappa, \mathcal{L}_X(v)\}) \qquad \mathcal{L}_Z(av) = \kappa.$$

Thus, if $\alpha = \mathsf{min}$, then $\mathcal{L}_Y = \mathcal{L}_Z$ if and only if $\mathcal{L}_X(v) \geq \kappa$ for all $v \in \Sigma^*$; if $\alpha = \mathsf{max}$, then $\mathcal{L}_Y = \mathcal{L}_Z$ if and only if $\mathcal{L}_X(v) \leq \kappa$ for all $v \in \Sigma^*$. Both of these threshold problems are undecidable for alphabets of size at least 2 [22–24]. □

The situation for automata over unary alphabets is more subtle; in particular, the threshold problem is not known to be undecidable in this case. However, there is a reduction to a long-standing open problem on LRSs.

Given an LRS $\langle u_i \rangle$, the *Positivity* problem is to decide whether u_i is non-negative for all $i \in \mathbb{N}$ (see, e.g., [25]). While the decidability of this problem has remained open for more than 80 years, it is known that a decision procedure for Positivity would necessarily entail breakthroughs in open problems in number theory. That is, it would give an algorithm to compute the *homogeneous Diophantine approximation type* for a class of transcendental numbers [25]. Furthermore, the Positivity problem can be reduced to the threshold problem on unary probabilistic automata. Putting everything together, we have the following.

Corollary 5.2. *The Positivity problem for linear recurrence sequences can be reduced to the equivalence problem of NPAs over a unary alphabet.*

Proof. The construction in Theorem 5.1 shows that the lesser-than threshold problem can be reduced to the equivalence problem for NPAs with max semantics, so we show that Positivity can be reduced to the lesser-than threshold problem on probabilistic automata with a unary alphabet. Given any rational LRS $\langle u_i \rangle$, clearly $\langle -u_i \rangle$ is an LRS as well, so by Theorem 4.4 there exists a rational stochastic matrix M such that

$$-u_n > 0 \iff u^T M^n v > 1/4$$

for all n, where $u = (1, 0, \ldots, 0)$ and $v = (0, 1, 0, \ldots, 0)$. Taking M to be the transition matrix, v to be the input vector, and u to be the output vector, the probabilistic automaton corresponding to the right-hand side is a nonsatisfying instance to the threshold problem with threshold $\leq 1/4$ if and only if the $\langle u_i \rangle$ is a satisfying instance of the Positivity problem.

Applying Proposition 3.4 yields an analogous reduction from Positivity to the equivalence problem of NPAs with min semantics. □

5.2. Checking approximate equivalence

The previous negative results show that deciding exact equivalence of NPAs is computationally intractable (or at least very difficult, for a unary alphabet). A natural question is whether we might be able to check approximate equivalence. In this section, we show how to approximate a metric on weighted languages. Our metric will be *discounted*—differences in weights of longer words will contribute less to the metric than differences in weights of shorter words.

Given $c \in [0, 1)$ and two weighted languages $l_1, l_2 : \Sigma^* \to [0, 1]$, we define

$$d_c(l_1, l_2) = \sum_{u \in \Sigma^*} |l_1(u) - l_2(u)| \cdot \left(\frac{c}{|\Sigma|}\right)^{|u|}.$$

Suppose that l_1 and l_2 are recognized by given NPAs. Since $d_c(l_1, l_2) = 0$ if and only if the languages (and automata) are equivalent, we cannot hope to compute the metric exactly. We can, however, compute the weight of any finite word under l_1 and l_2. Combined with the discounting in the metric, we can approximate this metric d_c within any desired (nonzero) error.

Theorem 5.3. *There is a procedure that given $c \in [0, 1)$, $\kappa > 0$, and computable functions $l_1, l_2 : \Sigma^* \to [0, 1]$ outputs $x \in \mathbb{R}_+$ such that $|d_c(l_1, l_2) - x| \leq \kappa$.*

Proof. Let $n = \lceil \log_c((1 - c) \cdot \kappa) \rceil \in \mathbb{N}$ and define

$$x = \sum_{u \in \Sigma^*, |u| < n} |l_1(u) - l_2(u)| \cdot \left(\frac{c}{|\Sigma|}\right)^{|u|}.$$

This sum is over a finite set of finite strings and the weights of $l_1(u)$ and $l_2(u)$ can all be computed exactly, so x is computable as well. Now we can bound

$$|d_c(l_1, l_2) - x| = \sum_{u \in \Sigma^*, |u| \geq n} |l_1(u) - l_2(u)| \cdot \left(\frac{c}{|\Sigma|}\right)^{|u|} \leq \sum_{u \in \Sigma^*, |u| \geq n} \left(\frac{c}{|\Sigma|}\right)^{|u|}$$

$$= \sum_{i\in\mathbb{N}, i\geq n} |\Sigma|^i \cdot \left(\frac{c}{|\Sigma|}\right)^i = \sum_{i\in\mathbb{N}, i\geq n} c^i = \frac{c^n}{1-c} \leq \kappa,$$

where the last step is because $n \geq \log_c((1-c)\cdot\kappa)$, and thus $c^n \leq (1-c)\cdot\kappa$, noting that $c \in [0,1)$ and $\kappa > 0$. □

We leave approximating other metrics on weighted languages—especially nondiscounted metrics—as an intriguing open question.

6. Conclusions

We have defined a novel probabilistic language semantics for nondeterministic probabilistic automata (NPAs). We proved that NPAs are strictly more expressive than deterministic probabilistic automata, and that exact equivalence is undecidable. We have shown how to approximate the equivalence question to arbitrary precision using a discounted metric. There are several directions for future work that we would like to explore. First, it would be interesting to see if different metrics can be defined on probabilistic languages and what approximate equivalence procedures they give rise to. Second, we would like to explore whether we can extend logical characterization results in the style of Panangaden et al. [34,35]. Third, we believe that there is an abstract understanding on the relationship between $\mathcal{P}_f$ and $\mathcal{P}_c$ that is more general than what we presented in this paper and might lead to analogue results for further automata types (for instance, work developed after we first submitted this paper on weak distributive laws [36] seems like a promising direction to explore). Finally, it would be interesting to investigate the class of languages recognizable by our NPAs.

Related work Variants of probabilistic automata have a long history. Our work is closest to the work of Segala [9] in that our automaton model has both nondeterminism and probabilistic choice. However, we enrich the states with an output weight that is used in the definition of the language semantics. Our language semantics is coarser than probabilistic (convex) bisimilarity [9] and bisimilarity on distributions [18]. In fact, in contrast to the hardness and undecidability results we proved for probabilistic language equivalence, bisimilarity on distributions can be shown to be decidable [18] with the help of convexity. The techniques we use in defining the semantics are closely related to the recent categorical understanding of bisimilarity on distributions [16]. Bonchi, Sokolova, and Vignudelli [37] also consider automata based on the convex powerset monad. Using an algebraic presentation for this monad, they also define the max and min semantics, as well as a "min-max" semantics where outputs are closed intervals in $[0,1]$. The corresponding equivalences resulting from these semantics are then compared to existing trace equivalences from the literature.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Nathanaël Fijalkow and the anonymous reviewers for their useful suggestions to improve the paper.

References

[1] G. van Heerdt, J. Hsu, J. Ouaknine, A. Silva, Convex Language Semantics for Nondeterministic Probabilistic Automata, vol. 11187, Springer, 2018, pp. 472–492.
[2] D. Kozen, Semantics of probabilistic programs, 1979, pp. 101–114.
[3] A. Legay, A.S. Murawski, J. Ouaknine, J. Worrell, On automated verification of probabilistic programs 4963 (2008) 173–187.
[4] M.O. Rabin, Probabilistic algorithms, in: Algorithms and Complexity: New Directions and Results, 1976, pp. 21–39.
[5] M.O. Rabin, N-process mutual exclusion with bounded waiting by $4\log_2 N$-valued shared variable, J. Comput. Syst. Sci. 25 (1) (1982) 66–75.
[6] B. Balle, J. Castro, R. Gavaldà, Adaptively learning probabilistic deterministic automata from data streams, Mach. Learn. 96 (1–2) (2014) 99–127.
[7] D. Ron, Y. Singer, N. Tishby, The power of amnesia: learning probabilistic automata with variable memory length, Mach. Learn. 25 (2) (1996) 117–149.
[8] V. Sassone, M. Nielsen, G. Winskel, Models for concurrency: towards a classification, Theor. Comput. Sci. 170 (1–2) (1996) 297–348.
[9] R. Segala, Modeling and verification of randomized distributed real-time systems, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.
[10] M. Bernardo, R. De Nicola, M. Loreti, Revisiting trace and testing equivalences for nondeterministic and probabilistic processes, Log. Methods Comput. Sci. 10 (1) (2014).
[11] H. Hermanns, J. Katoen, The how and why of interactive Markov chains, in: LNCS, vol. 6286, 2009, pp. 311–337.
[12] T.A. Henzinger, Quantitative reactive modeling and verification, Comput. Sci. Res. Dev. 28 (4) (2013) 331–344.
[13] M.Z. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: LNCS, vol. 6806, 2011, pp. 585–591.
[14] M. Swaminathan, J.-P. Katoen, E.-R. Olderog, Layered reasoning for randomized distributed algorithms, Form. Asp. Comput. 24 (4) (2012) 477–496.
[15] V. Vignudelli, Behavioral equivalences for higher-order languages with probabilities, Ph.D. thesis, University of Bologna, Italy, 2017.
[16] F. Bonchi, A. Silva, A. Sokolova, The power of convex algebras, in: LIPIcs, vol. 85, 2017, pp. 23:1–23:18.
[17] Y. Deng, R.J. van Glabbeek, M. Hennessy, C. Morgan, Testing finitary probabilistic processes, in: LNCS, vol. 5710, 2009, pp. 274–288.
[18] H. Hermanns, J. Krcál, J. Kretínský, Probabilistic bisimulation: naturally on distributions, in: LNCS, vol. 8704, 2014, pp. 249–265.
[19] M.Y. Vardi, Branching vs. linear time: final showdown, in: LNCS, vol. 2031, 2001, pp. 1–22.
[20] F. Bonchi, D. Pous, Hacking nondeterminism with induction and coinduction, Commun. ACM 58 (2) (2015) 87–95.
[21] M.O. Rabin, Probabilistic automata, Inf. Control 6 (3) (1963) 230–245.
[22] A. Paz, Introduction to Probabilistic Automata, Academic Press, 1971.
[23] V.D. Blondel, V. Canterini, Undecidable problems for probabilistic automata of fixed dimension, Theory Comput. Syst. 36 (2003) 231–245.
[24] N. Fijalkow, Undecidability results for probabilistic automata, ACM SIGLOG News 4 (4) (2017) 10–17.

[25] J. Ouaknine, J. Worrell, Positivity problems for low-order linear recurrence sequences, 2014, pp. 366–379.

[26] M.A. Arbib, E.G. Manes, Fuzzy machines in a category, Bull. Am. Math. Soc. 13 (1975) 169–210.

[27] A. Silva, F. Bonchi, M.M. Bonsangue, J.J.M.M. Rutten, Generalizing determinization from automata to coalgebras, Log. Methods Comput. Sci. 9 (1) (2013).

[28] S. Goncharov, S. Milius, A. Silva, Towards a coalgebraic Chomsky hierarchy, in: IFIP International Conference on Theoretical Computer Science (TCS), Rome, Italy, 2014, pp. 265–280.

[29] D. Varacca, G. Winskel, Distributing probability over non-determinism, Math. Struct. Comput. Sci. 16 (1) (2006) 87–113, https://doi.org/10.1017/S0960129505005074.

[30] M. Zwart, D. Marsden, No-go theorems for distributive laws, in: 34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019, IEEE, 2019, pp. 1–13.

[31] F. Dahlqvist, R. Neves, Compositional semantics for new paradigms: probabilistic, hybrid and beyond, CoRR, arXiv:1804.04145 [abs], 2018, arXiv:1804.04145, http://arxiv.org/abs/1804.04145.

[32] G. Everest, A.J. van der Poorten, I.E. Shparlinski, T. Ward, Recurrence Sequences, Mathematical Surveys and Monographs, vol. 104, American Mathematical Society, 2003.

[33] S. Akshay, T. Antonopoulos, J. Ouaknine, J. Worrell, Reachability problems for Markov chains, Inf. Process. Lett. 115 (2) (2015) 155–158.

[34] N. Fijalkow, B. Klin, P. Panangaden, Expressiveness of probabilistic modal logics, revisited, in: LIPIcs, vol. 80, 2017, pp. 105:1–105:12.

[35] J. Desharnais, A. Edalat, P. Panangaden, A logical characterization of bisimulation for labeled Markov processes, 1998, pp. 478–487.

[36] A. Goy, D. Petrisan, Combining probabilistic and non-deterministic choice via weak distributive laws, in: H. Hermanns, L. Zhang, N. Kobayashi, D. Miller (Eds.), LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020, ACM, 2020, pp. 454–464.

[37] F. Bonchi, A. Sokolova, V. Vignudelli, The theory of traces for systems with nondeterminism and probability, arXiv:1808.00923, 2018.