# Convex Language Semantics for Nondeterministic Probabilistic Automata*

Gerco van Heerdt[1], Justin Hsu[2], Joël Ouaknine[3], and Alexandra Silva[1]

[1] University College London
[2] Cornell University
[3] MPI-SWS and Oxford University

**Abstract.** We explore language semantics for automata combining probabilistic and nondeterministic behavior. 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 have been used in the study of such topics as the semantics and correctness of probabilistic programming languages [18,20], randomized algorithms [17,3,28], and machine learning [4,24]. Removing randomness but adding nondeterminism, nondeterministic automata are established tools for describing concurrent and distributed systems [25].

Interest in systems that exhibit both random and nondeterministic behavior goes back to Rabin's randomized techniques to increase the efficiency of distributed algorithms in the 1970s and 1980s [22,23]. This line of research yielded several automata models supporting both nondeterministic and probabilistic choice [26,5,15]. Many formal techniques and tools were developed for these models, and they have been successfully used in verification tasks [14,19,15], but there are many ways of combining nondeterminism and randomization, and there remains plenty of 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 [26] 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 associated with a word in a similar way to what one

* This work was partially supported by ERC starting grant ProFoundNet (679127), ERC consolidator grant AVS-ISS (648701), a Leverhulme Prize (PLP-2016-129), and an NSF grant (1637532).

would do for 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.

Equivalence in our semantics is language equivalence (also known as trace equivalence), which is coarser than probabilistic bisimulation [30,8,9,16], which distinguishes systems with different branching structure even if the total weight assigned to a word is the same. This generalizes the classical difference between branching and linear semantics [29] to the probabilistic setting, with different target applications calling for different semantics.

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, [8,7]).

Then, we compare NPAs with standard, deterministic probabilistic automata (DPAs) (sometimes called *reactive systems* in the literature) in Section 4. Our semantics ensures that NPAs recover DPAs in the special case when there is no nondeterministic choice. More interestingly, 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 were shown to be undecidable by Blondel and Canterini [6]. 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 necessarily entail breakthroughs in open problems in number theory [21]. Finally, we show that despite the undecidability result we can provide a discounted metric that can be computed to arbitrarily high precision.

We survey related work and conclude in Section 6.

## 2 Preliminaries

Before we present our main technical results, we review some necessary mathematical background on convex algebra, 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 there is an operation denoted $\sum_{i=1}^n p_i(-)_i \colon A^n \to A$ satisfying the following properties for $(a_1, \ldots, a_n) \in A^n$:

**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 $[0,1]$ summing up to 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 $A$.

**Definition 1.** *A subset $S \subseteq A$ is* convex *if it is closed under all convex combinations. (Such a set can also be seen as 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.** *An* affine map *between two convex sets $A$ and $B$ is a function $h \colon A \to B$ commuting with convex combinations:*

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

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

$$\begin{array}{ccc} T & \xrightarrow{\eta} & TT \\ {\scriptstyle T\eta}\downarrow & \searrow & \downarrow{\scriptstyle \mu} \\ TT & \xrightarrow{\mu} & T \end{array} \qquad\qquad \begin{array}{ccc} TTT & \xrightarrow{T\mu} & TT \\ {\scriptstyle \mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ TT & \xrightarrow{\mu} & T \end{array}$$

When there is no risk of confusion, we identify a monad with its endofunctor. An example of a monad in the category of sets is the triple $(\mathcal{P}, \{-\}, \bigcup)$, where $\mathcal{P}$ denotes the finite powerset functor sending each set to the set of its finite subsets, $\{-\}$ is the singleton operation, and $\bigcup$ is set union.

**Definition 4.** *An* algebra for a monad $(T, \eta, \mu)$ *is a pair* $(X, h)$ *consisting of a carrier set* $X$ *and a function* $h\colon TX \to X$ *making the following diagrams commute.*

$$\begin{array}{ccc} X & \xrightarrow{\eta} & TX \\ & \searrow^{=} & \downarrow{\scriptstyle h} \\ & & X \end{array} \qquad\qquad \begin{array}{ccc} TTX & \xrightarrow{Th} & TX \\ {\scriptstyle \mu}\downarrow & & \downarrow{\scriptstyle h} \\ TX & \xrightarrow{h} & X \end{array}$$

**Definition 5.** *A homomorphism from an algebra* $(X, h)$ *to an algebra* $(Y, k)$ *for a monad* $T$ *is a function* $f\colon X \to Y$ *making the diagram below commute.*

$$\begin{array}{ccc} TX & \xrightarrow{Tf} & TY \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle k} \\ X & \xrightarrow{f} & Y \end{array}$$

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 any monad together with their homomorphisms form a category.

### 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 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 $\{d_1, \ldots, d_n\} \subseteq \mathcal{D}X$ and define $m(d) = \sum_{i=1}^{n} p_i d_i$, where $p_i$ is the probability of producing $d_i$ under $d$. The category of algebras for the distribution monad is precisely the category of convex sets and affine maps—we will often convert between these two representations implicitly.

In the category of convex sets, the *finitely generated nonempty convex powerset monad* [8] $(\mathcal{P}_c, \{-\}, \bigcup)$ maps a convex set $A$ to the set of finitely generated nonempty convex subsets of $A$.[4] The convex algebra structure on $\mathcal{P}_c A$ is given by $\sum_{i=1}^{n} p_i U_i = \{\sum_{i=1}^{n} p_i u_i \mid u_i \in U_i \text{ for all } 1 \le i \le n\}$ 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.

[4] The monad defined in the paper cited does not have the restriction of containing only convex subsets that are finitely generated. However, since all the monad operations preserve this property, the restricted monad is also well-defined.

**Lemma 1.** *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 $[p,q] = p\{1\} + (q-p)[0,1] + (1-q)\{0\}$. □

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.

Explicitly, writing $\omega_A \colon \mathcal{D}\mathcal{P}_c A \to \mathcal{P}_c A$ for the (affine) convex algebra structure on $\mathcal{P}_c A$ for any convex algebra $A$, the composite monad $(\mathcal{P}_c\mathcal{D}, \hat{\delta}, \hat{m})$ is given by

$$\begin{array}{ccc} X & \overset{\hat{\delta}}{\dashrightarrow} & \\ {\scriptstyle\delta}\downarrow & & \\ \mathcal{D}X & \xrightarrow{\{-\}} & \mathcal{P}_c\mathcal{D}X \end{array} \qquad\qquad \begin{array}{ccc} \mathcal{P}_c\mathcal{D}\mathcal{P}_c\mathcal{D}X & & \\ {\scriptstyle\mathcal{P}_c\omega}\downarrow & \overset{\hat{m}}{\dashrightarrow} & \\ \mathcal{P}_c\mathcal{P}_c\mathcal{D}X & \xrightarrow{\bigcup} & \mathcal{P}_c\mathcal{D}X \end{array} \tag{1}$$

For all convex sets $A$ and finite nonempty subsets $S \subseteq A$, we can define the *convex closure* of $S$ (sometimes called the *convex hull*) $\mathsf{Conv}(S) \in \mathcal{P}_c A$ by

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

where $\alpha \colon \mathcal{D}A \to A$ is the convex algebra structure on $A$. $\mathsf{Conv}$ is in fact a natural transformation, a fact we will use later.

**Lemma 2.** *For all convex sets $(A,\alpha)$ and $(B,\beta)$, affine maps $f \colon A \to B$, and finite nonempty subsets $S \subseteq A$, $(\mathcal{P}_c f \circ \mathsf{Conv})(S) = (\mathsf{Conv} \circ \mathcal{P} f)(S)$.*

*Proof.* We will first show that

$$\{\mathcal{D}f(d) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\} = \{d \in \mathcal{D}B \mid \mathsf{supp}(d) \subseteq \{f(a) \mid a \in S\}\} \tag{2}$$

for all finite nonempty $S \subseteq A$. 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

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

Now we have

$$
\begin{aligned}
&(\mathcal{P}_c f \circ \mathsf{Conv})(S) \\
&= \mathcal{P}_c f(\{\alpha(d) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\}) && \text{(definition of } \mathsf{Conv}) \\
&= \{f(\alpha(d)) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\} && \text{(definition of } \mathcal{P}_c f) \\
&= \{\beta(\mathcal{D}f(d)) \mid d \in \mathcal{D}A, \mathsf{supp}(d) \subseteq S\} && (f \text{ is affine}) \\
&= \{\beta(d) \mid d \in \mathcal{D}B, \mathsf{supp}(d) \subseteq \{f(a) \mid a \in S\}\} && (2) \\
&= \mathsf{Conv}(\{f(a) \mid a \in S\}) && \text{(definition of } \mathsf{Conv}) \\
&= (\mathsf{Conv} \circ \mathcal{P}f)(S) && \text{(definition of } \mathcal{P}f).
\end{aligned}
$$

□

### 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., [2,27,13]). This categorical framework is the foundation of our language semantics for NPA.

**Definition 6.** *Given a monad $(T, \eta, \mu)$ in the category of sets, an output set $O$, and a (finite) alphabet $A$, a* $T$-automaton *is defined by a tuple $(S, s_0, \gamma, \{\tau_a\}_{a \in A})$, where $S$ is the set of* states*, $s_0 \in S$ is the* initial state*, $\gamma \colon S \to O$ is the* output function*, and $\tau_a \colon S \to TS$ for $a \in A$ 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.

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

**Definition 7.** *Given a monad $(T, \eta, \mu)$ in the category of sets, an output set $O$ with a $T$-algebra structure $o \colon TO \to O$, and a (finite) alphabet $A$, a $T$-automaton $(S, s_0, \gamma, \{\tau_a\}_{a \in A})$ can be determinized into the deterministic automaton $(TS, s_0', \gamma', \{\tau_a'\}_{a \in A})$ given by $s_0' = \eta(s_0) \in TS$ and*

$$
\begin{aligned}
\gamma' &\colon TS \to O & \qquad \tau_a' &\colon TS \to TS \\
\gamma' &= o \circ T\gamma & \qquad \tau_a' &= \mu \circ T\tau_a.
\end{aligned}
$$

This construction allows us to define the language semantics of any $T$-automaton as the semantics of its determinization. More formally, we have the following definition.

**Definition 8.** *Given a monad $(T, \eta, \mu)$ in the category of sets, an output set $O$ with a $T$-algebra structure $o \colon TO \to O$, and a (finite) alphabet $A$, the* language

*accepted by a $T$-automaton $\mathcal{A} = (S, s_0, \gamma, \{\tau_a\}_{a\in A})$ is the function $\mathcal{L}_\mathcal{A} \colon A^* \to O$ given by $\mathcal{L}_\mathcal{A} = (l_\mathcal{A} \circ \eta)(s_0)$, where $l_\mathcal{A} \colon TS \to O^{A^*}$ is defined inductively by*

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

As an example, we recover deterministic probabilistic automata (DPAs) by taking $T$ to be the distribution monad $\mathcal{D}$ and letting the output set be the interval $[0, 1]$. That is, a DPA with finite[5] state space $S$ has an output function of type $S \to [0, 1]$, and each of its transition functions is of type $S \to \mathcal{D}S$. In order to derive a semantics for this automata, we use the usual $\mathcal{D}$-algebra structure $\mathbb{E} \colon \mathcal{D}[0, 1] \to [0, 1]$ computing the expected weight.

More concretely, the semantics works as follows. Let $(S, s_0, \gamma, \{\tau_a\}_{a\in A})$ 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 sum $\sum_{i=1}^n p_i \gamma(s_i)$. On reading a symbol $a \in A$, 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 1.* One may wonder if the automaton model would be more expressive if the initial state $s_0$ in an automaton $(S, s_0, \gamma, \{\tau_a\}_{a\in A})$ 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 $a \in A$ 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 $\mu$, an output vector $\gamma$, and transitions $\tau_a$. In typical presentations, $\mu$ and $\gamma$ are represented as weight vectors over states, and the $\tau_a$ are encoded by stochastic matrices.

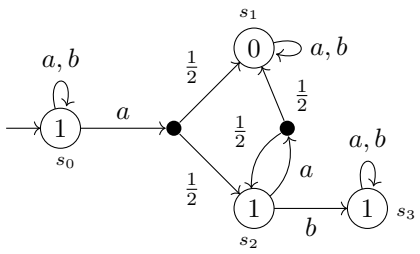
## 3 Nondeterministic Probabilistic Automata

We work with an automaton model supporting probabilistic and nondeterministic behavior, inspired by Segala [26]. 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 makes a probabilistic transition to the following state. Each state has an output weight in $[0, 1]$. The following formalization is an instantiation of Definition 6 with the monad $\mathcal{P}_c\mathcal{D}$.

**Definition 9.** *A* nondeterministic probabilistic automaton *(NPA) over a (finite) alphabet $A$ is defined by a tuple $(S, s_0, \gamma, \{\tau_a\}_{a\in A})$, 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*.*

[5] Concrete automata considered in this paper will have a finite state space, but the general automaton in Definition 6 does not have this restriction. The distribution monad, for example, does not preserve finite sets in general.

As an example, consider the NPA below.




(3)

States are labeled by their direct output (i.e., their weight from $\gamma$) 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, from which the outgoing transitions represent the distribution. Those edges are labeled with probabilities.

Our NPAs recognize weighted languages. The rest of the section is concerned with formally defining this 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 $\mathbb{E}$, making the diagram below commute.

$$\begin{array}{ccc} \mathcal{D}[0,1] & & \\ {\scriptstyle \{-\}}\downarrow & \searrow^{\mathbb{E}} & \\ \mathcal{P}_c\mathcal{D}[0,1] & \xrightarrow{o} & [0,1] \end{array} \tag{4}$$

### 3.2 Characterizing the Convex Algebra on $[0,1]$

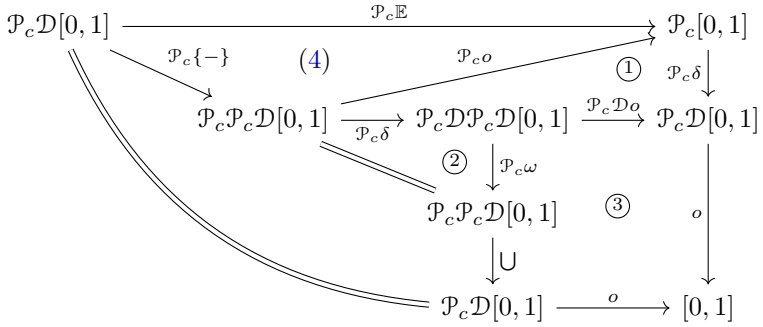
While in principle there could be many different $\mathcal{P}_c\mathcal{D}$-algebras on $[0,1]$ leading to different language semantics for NPAs, we show that (i) each algebra extending the $\mathcal{D}$-algebra on $[0,1]$ is fully determined by a $\mathcal{P}_c$-algebra on $[0,1]$, and (ii) there are exactly two $\mathcal{P}_c$-algebras on $[0,1]$: the map computing the minimum and the map computing the maximum.

**Proposition 1.** *Any $\mathcal{P}_c\mathcal{D}$-algebra on $[0,1]$ extending $\mathbb{E}\colon \mathcal{D}[0,1] \to [0,1]$ is of the form $\mathcal{P}_c\mathcal{D}[0,1] \xrightarrow{\mathcal{P}_c\mathbb{E}} \mathcal{P}_c[0,1] \xrightarrow{\alpha} [0,1]$, where $\alpha$ is a $\mathcal{P}_c$-algebra.*

*Proof.* Let $o\colon \mathcal{P}_c\mathcal{D}[0,1] \to [0,1]$ be a $\mathcal{P}_c\mathcal{D}$-algebra extending $\mathbb{E}$. We define

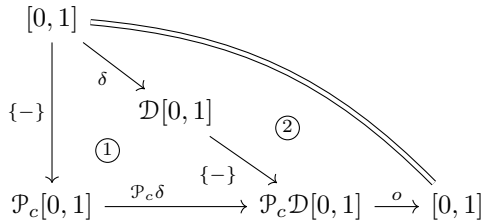
$$\alpha = \mathcal{P}_c[0,1] \xrightarrow{\mathcal{P}_c\delta} \mathcal{P}_c\mathcal{D}[0,1] \xrightarrow{o} [0,1].$$

Indeed, the diagram

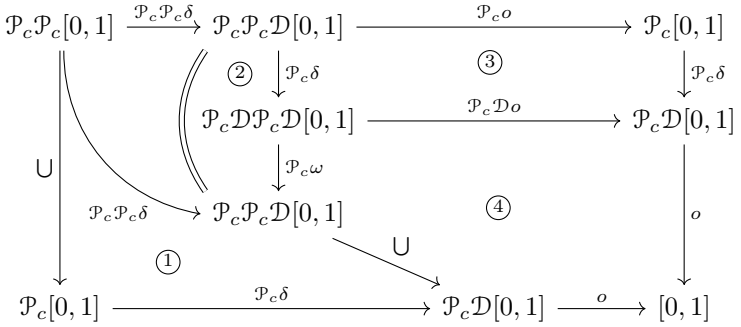



① naturality of $\delta$ ② $\omega$ is a convex algebra ③ $o$ is a $\mathcal{P}_c\mathcal{D}$-algebra

commutes, so it only remains to show that $\alpha$ is a $\mathcal{P}_c$-algebra. This can be seen from the commutative diagrams below.




① naturality of $\{-\}$
② $o$ is a $\mathcal{P}_c\mathcal{D}$-algebra




① naturality of $\bigcup$
② $\omega$ is a convex algebra
③ naturality of $\delta$
④ $o$ is a $\mathcal{P}_c\mathcal{D}$-algebra

□

**Proposition 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.

$$\begin{array}{ccc} \mathcal{P}_c\mathcal{P}_c[0,1] & \xrightarrow{\mathcal{P}_c\alpha} & \mathcal{P}_c[0,1] \\ \cup\downarrow & & \downarrow\alpha \\ \mathcal{P}_c[0,1] & \xrightarrow{\alpha} & [0,1] \end{array} \tag{5}$$

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

$$\begin{aligned} \alpha([0,1]) &= \alpha\left(\bigcup\{[0,p] \mid p \in [0,1]\}\right) \\ &= (\alpha \circ \bigcup \circ \mathsf{Conv})(\{\{0\},[0,1]\}) \\ &= (\alpha \circ \mathcal{P}_c\alpha \circ \mathsf{Conv})(\{\{0\},[0,1]\}) && (5) \\ &= (\alpha \circ \mathsf{Conv} \circ \mathcal{P}\alpha)(\{\{0\},[0,1]\}) && (\text{Lemma } 2) \\ &= (\alpha \circ \mathsf{Conv})(\{\alpha(\{0\}),\alpha([0,1])\}) && (\text{definition of } \mathcal{P}_c\alpha) \\ &= (\alpha \circ \mathsf{Conv})(\{0,\alpha([0,1])\}) \\ &= \alpha([0,\alpha([0,1])]) \\ &= \alpha(\alpha([0,1])[0,1] + (1-\alpha([0,1]))\{0\}) \\ &= \alpha([0,1]) \cdot \alpha([0,1]) + (1-\alpha([0,1])) \cdot \alpha(\{0\}) && (\alpha \text{ is affine}) \\ &= \alpha([0,1])^2 + (1-\alpha([0,1])) \cdot 0 \\ &= \alpha([0,1])^2. \end{aligned}$$

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

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

□

**Corollary 1.** *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 (3). Since we can always choose to remain in the initial state, the $\mathsf{max}$ semantics assigns 1 to each word for this automaton. The $\mathsf{min}$ 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 $u \in A^*$ the automaton outputs $2^{-n}$, where $n$ is the length of the longest sequence of $a$'s occurring in $u$.

The semantics coming from $\mathsf{max}$ and $\mathsf{min}$ are highly symmetrical; in a sense, they are two representations of the same semantics.[6] Technically, we establish the following relation between the two semantics—this will be useful to avoid repeating proofs twice for each property.

[6] The $\mathsf{max}$ semantics is perhaps preferable since it recovers standard nondeterministic finite automata when there is no probabilistic choice and the output weights are in $\{0,1\}$, but this is a minor point.

**Proposition 3.** *Consider an NPA $\mathcal{A} = (S, s_0, \gamma, \{\tau_a\}_{a\in A})$ under the* min *semantics. Define $\gamma' \colon S \to [0,1]$ by $\gamma'(s) = 1 - \gamma(s)$, and consider the NPA $\mathcal{A}' = (S, s_0, \gamma', \{\tau_a\}_{a\in A})$ under the* max *semantics. Then $\mathcal{L}_{\mathcal{A}'}(u) = 1 - \mathcal{L}_{\mathcal{A}}(u)$ for all $u \in A^*$.*

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

$$
\begin{aligned}
& l_{\mathcal{A}'}(x)(\varepsilon) \\
&= (\mathsf{max} \circ \mathcal{P}_c\mathbb{E} \circ \mathcal{P}_c\mathcal{D}\gamma')(x) && \text{(Definition 8)} \\
&= (\mathsf{max} \circ \mathcal{P}_c\mathbb{E}) \left( \left\{ \lambda p. \sum_{s\in S, \gamma'(s)=p} d(s) \;\middle|\; d \in x \right\} \right) && \text{(definition of } \mathcal{P}_c\mathcal{D}\gamma') \\
&= \mathsf{max} \left( \left\{ \sum_{p\in[0,1]} p \sum_{s\in S, \gamma'(s)=p} d(s) \;\middle|\; d \in x \right\} \right) && \text{(definition of } \mathcal{P}_c\mathbb{E}) \\
&= \mathsf{max} \left( \left\{ \sum_{p\in[0,1]} p \sum_{s\in S, \gamma(s)=1-p} d(s) \;\middle|\; d \in x \right\} \right) && \text{(definition of } \gamma') \\
&= \mathsf{max} \left( \left\{ \sum_{p\in[0,1]} (1-p) \sum_{s\in S, \gamma(s)=p} d(s) \;\middle|\; d \in x \right\} \right) \\
&= \mathsf{max} \left( \left\{ \sum_{p\in[0,1]} (1-p) \cdot \mathcal{D}\gamma(d)(p) \;\middle|\; d \in x \right\} \right) \\
&= \mathsf{max} \left( \left\{ 1 - \sum_{p\in[0,1]} p \cdot \mathcal{D}\gamma(d)(p) \;\middle|\; d \in x \right\} \right) \\
&= 1 - \mathsf{min} \left( \left\{ \sum_{p\in[0,1]} p \cdot \mathcal{D}\gamma(d)(p) \;\middle|\; d \in x \right\} \right) \\
&= 1 - (\mathsf{min} \circ \mathcal{P}_c\mathbb{E})(\{\mathcal{D}\gamma(d) \mid d \in x\}) && \text{(definition of } \mathcal{P}_c\mathbb{E}) \\
&= 1 - (\mathsf{min} \circ \mathcal{P}_c\mathbb{E} \circ \mathcal{P}_c\mathcal{D}\gamma)(x) && \text{(definition of } \mathcal{P}_c\mathcal{D}\gamma') \\
&= 1 - l_{\mathcal{A}}(x)(\varepsilon) && \text{(Definition 8).}
\end{aligned}
$$

Furthermore,

$$
\begin{aligned}
l_{\mathcal{A}'}(x)(av) &= l_{\mathcal{A}'} \left( \left( \bigcup \circ \mathcal{P}_c\omega \circ \mathcal{P}_c\mathcal{D}\tau_a \right)(x) \right)(v) && \text{(Definition 8)} \\
&= 1 - l_{\mathcal{A}} \left( \left( \bigcup \circ \mathcal{P}_c\omega \circ \mathcal{P}_c\mathcal{D}\tau_a \right)(x) \right)(v) && \text{(induction hypothesis)} \\
&= 1 - l_{\mathcal{A}}(x)(av) && \text{(Definition 8).}
\end{aligned}
$$

This concludes the proof. □

## 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 the Hankel matrix, and actually shows that NPAs are strictly more expressive than weighted finite automata (WFAs).

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 (3) using the $\mathsf{min}$ algebra.

**Theorem 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, and hence the WFA cannot exist.

The proof is by induction on a natural number $m$. Assume that for all natural numbers $i \leq m$ the languages $l(a^i b)$ are linearly independent. For all $i \leq m$ 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 $i \leq m$, then there are constants $c_1, \ldots, c_m \in \mathbb{R}$ 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 $i \leq m + 1$ 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 $i \leq m$ 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 $\mathsf{max}$ algebra semantics—one can easily repeat the argument in the above theorem for the language accepted by the NPA resulting from applying Proposition 3 to the NPA (3).

### 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 $\langle u_i \rangle = u_0, u_1, \ldots$ 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 $u_0, \ldots, u_{k-1} \in \mathbb{R}$ (the *initial conditions*), and coefficients $b_0, \ldots, b_{k-1} \in \mathbb{R}$, 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. [11] for details.

**Theorem 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 (see, e.g., [21]).** *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 3 (Akshay, et al. [1, Theorem 3, Corollary 4]).** *Given any LRS $\langle u_i \rangle$, there exists a stochastic matrix $M$ such that*

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

*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 1). 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.** *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

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

with $x_0 = 1$ and $x_1 = a$, so $\langle x_n \rangle$ is an order-2 rational LRS. By Theorem 3, 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 1). 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) and hence so is $\langle t_n \rangle$ with $t_n = w_n - y_n$. If we now consider the zero set

$$\begin{aligned} S &= \{n \in \mathbb{N} \mid t_n = 0\} \\ &= \{n \in \mathbb{N} \mid y_n > 1/4\} && (y_n \neq 1/4) \\ &= \{n \in \mathbb{N} \mid x_n > 0\} && (\text{Theorem } 3) \\ &= \{n \in \mathbb{N} \mid \mathrm{Re}(z^n) > 0\} && (\text{by definition}), \end{aligned}$$

Theorem 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. □

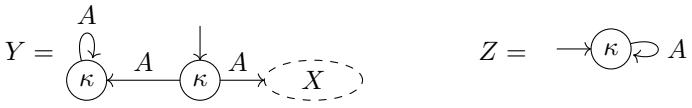
## 5 Checking Language Equivalence of NPAs

Given the coalgebraic NPA model, 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 in finite time.

### 5.1 Undecidability and Hardness

**Theorem 5.** *Equivalence of NPAs is undecidable when* $|A| \geq 2$ *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 $A$ to indicate a transition for every element of $A$. We see that $\mathcal{L}_Y(\varepsilon) = \kappa = \mathcal{L}_Z(\varepsilon)$ and (for $\alpha$ either $\mathsf{min}$ or $\mathsf{max}$, as follows from Corollary 1)

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

Thus, if $\alpha = \mathsf{min}$, then $\mathcal{L}_Y = \mathcal{L}_Z$ if and only if $\mathcal{L}_X(v) \geq \kappa$ for all $v \in A^*$; if $\alpha = \mathsf{max}$, then $\mathcal{L}_Y = \mathcal{L}_Z$ if and only if $\mathcal{L}_X(v) \leq \kappa$ for all $v \in A^*$. Both of these threshold problems were shown to be undecidable, for alphabets of size at least 2, by Blondel and Canterini [6, Theorem 2.1]. □

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., [21]). 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 [21]. Furthermore, the Positivity problem can be reduced to the threshold problem on unary probabilistic automata. Putting everything together, we have the following reduction.

**Corollary 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 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 3 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 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 $l_1, l_2 \colon A^* \to [0, 1]$, we define

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

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 6.** *There is a procedure that given $c \in [0, 1)$, $\kappa > 0$, and computable functions $l_1, l_2 \colon A^* \to [0, 1]$ 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

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

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

$$\begin{aligned}
|d_c(l_1, l_2) - x| &= \sum_{u \in A^*, |u| \geq n} |l_1(u) - l_2(u)| \cdot \left(\frac{c}{|A|}\right)^{|u|} \\
&\leq \sum_{u \in A^*, |u| \geq n} \left(\frac{c}{|A|}\right)^{|u|} \\
&= \sum_{i \in \mathbb{N}, i \geq n} |A|^i \cdot \left(\frac{c}{|A|}\right)^i \\
&= \sum_{i \in \mathbb{N}, i \geq n} c^i \\
&= \frac{c^n}{1-c} \leq \kappa,
\end{aligned}$$

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 two 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. [12,10]. Finally, it would be interesting to investigate the class of languages recognizable by our NPAs.

*Related Work.* There are many papers studying probabilitic automata and variants thereof. The work in our paper is closest to the work of Segala [26] 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 [26] and bisimilarity on distributions [16]. 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 [16] 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 [8].

## References


1. S Akshay, Timos Antonopoulos, Joël Ouaknine, and James Worrell. Reachability problems for Markov chains. *Information Processing Letters*, 115(2):155–158, 2015.
2. Michael A. Arbib and Ernest G. Manes. Fuzzy machines in a category. *Bulletin of the AMS*, 13:169–210, 1975.
3. Christel Baier. Quantitative analysis of randomized distributed systems and probabilistic automata. In *International Conference on Algebraic Informatics (CAI), Porquerolles, France*, volume 8080 of *Lecture Notes in Computer Science*, pages 4–5. Springer-Verlag, 2013.
4. Borja Balle, Jorge Castro, and Ricard Gavaldà. Adaptively learning probabilistic deterministic automata from data streams. *Machine Learning*, 96(1–2):99–127, 2014.
5. Marco Bernardo, Rocco De Nicola, and Michele Loreti. Revisiting trace and testing equivalences for nondeterministic and probabilistic processes. *Logical Methods in Computer Science*, 10(1), 2014.
6. Vincent D. Blondel and Vincent Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theory of Computing Systems*, 36:231–245, 2003.
7. Filippo Bonchi and Damien Pous. Hacking nondeterminism with induction and coinduction. *Communications of the ACM*, 58(2):87–95, 2015.
8. Filippo Bonchi, Alexandra Silva, and Ana Sokolova. The power of convex algebras. In *International Conference on Concurrency Theory (CONCUR), Berlin, Germany*, volume 85 of *Leibniz International Proceedings in Informatics*, pages 23:1–23:18. Schloss Dagstuhl–Leibniz Center for Informatics, 2017.
9. Yuxin Deng, Rob J. van Glabbeek, Matthew Hennessy, and Carroll Morgan. Testing finitary probabilistic processes. In *International Conference on Concurrency Theory (CONCUR), Bologna, Italy*, volume 5710 of *Lecture Notes in Computer Science*, pages 274–288. Springer-Verlag, 2009.
10. Josee Desharnais, Abbas Edalat, and Prakash Panangaden. A logical characterization of bisimulation for labeled Markov processes. In *IEEE Symposium on Logic in Computer Science (LICS), Indianapolis, Indiana*, pages 478–487, 1998.
11. Graham Everest, Alfred J. van der Poorten, Igor E. Shparlinski, and Thomas Ward. *Recurrence Sequences*, volume 104 of *Mathematical surveys and monographs*. American Mathematical Society, 2003.
12. Nathanaël Fijalkow, Bartek Klin, and Prakash Panangaden. Expressiveness of probabilistic modal logics, revisited. In *International Colloquium on Automata, Languages and Programming (ICALP), Warsaw, Poland*, volume 80 of *LIPIcs*, pages 105:1–105:12. Schloss Dagstuhl–Leibniz Center for Informatics, 2017.
13. Sergey Goncharov, Stefan Milius, and Alexandra Silva. Towards a coalgebraic Chomsky hierarchy. In *IFIP International Conference on Theoretical Computer Science (TCS), Rome, Italy*, pages 265–280. Springer-Verlag, 2014.
14. Thomas A. Henzinger. Quantitative reactive modeling and verification. *Computer Science – Research and Development*, 28(4):331–344, 2013.
15. Holger Hermanns and Joost-Pieter Katoen. The how and why of interactive Markov chains. In *International Symposia on Formal Methods for Components and Objects (FMCO), Eindhoven, The Netherlands*, volume 6286 of *Lecture Notes in Computer Science*, pages 311–337. Springer-Verlag, 2009.
16. Holger Hermanns, Jan Krcál, and Jan Kretínský. Probabilistic bisimulation: Naturally on distributions. In *International Conference on Concurrency Theory (CONCUR), Rome, Italy*, volume 8704 of *Lecture Notes in Computer Science*, pages 249–265. Springer-Verlag, 2014.

17. Andrew Hinton, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM: A tool for automatic verification of probabilistic systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Vienna, Austria*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer-Verlag, 2006.
18. Dexter Kozen. Semantics of probabilistic programs. In *IEEE Symposium on Foundations of Computer Science (FOCS), San Juan, Puerto Rico*, pages 101–114, 1979.
19. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification (CAV), Snowbird, Utah*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer-Verlag, 2011.
20. Axel Legay, Andrzej S. Murawski, Joël Ouaknine, and James Worrell. On automated verification of probabilistic programs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Budapest, Hungary*, volume 4963 of *Lecture Notes in Computer Science*, pages 173–187. Springer-Verlag, 2008.
21. Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *ACM–SIAM Symposium on Discrete Algorithms (SODA), Portland, Oregon*, pages 366–379, 2014.
22. Michael O. Rabin. Probabilistic algorithms. *Algorithms and Complexity: New directions and results*, pages 21–39, 1976.
23. Michael O. Rabin. $N$-process mutual exclusion with bounded waiting by $4 \log_2 N$-valued shared variable. *Journal of Computer and System Sciences*, 25(1):66–75, 1982.
24. Dana Ron, Yoram Singer, and Naftali Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2):117–149, November 1996.
25. Vladimiro Sassone, Mogens Nielsen, and Glynn Winskel. Models for concurrency: Towards a classification. *Theoretical Computer Science*, 170(1–2):297–348, 1996.
26. Roberto Segala. *Modeling and Verification of Randomized Distributed Real-time Systems*. PhD thesis, Cambridge, MA, USA, 1995.
27. Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013.
28. Mani Swaminathan, Joost-Pieter Katoen, and Ernst-Rüdiger Olderog. Layered reasoning for randomized distributed algorithms. *Formal Aspects of Computing*, 24(4):477–496, July 2012.
29. Moshe Y. Vardi. Branching vs. linear time: Final showdown. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Genova, Italy*, volume 2031 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2001.
30. Valeria Vignudelli. *Behavioral Equivalences for Higher-Order Languages with Probabilities*. PhD thesis, University of Bologna, Italy, 2017.