Decidable and Expressive Classes of Probabilistic Automata

Yue Ben[a], Rohit Chadha[b], A. Prasad Sistla[a], Mahesh Viswanathan[c]

[a] *University of Illinois at Chicago, USA*
[b] *University of Missouri, USA*
[c] *University of Illinois at Urbana-Champaign, USA*

Abstract

k-Hierarchical probabilistic automata (HPA) are probabilistic automata whose states are partitioned into $k + 1$ levels such that for any state and input symbol, at most one transition goes to a state at the same level, and others go to higher level states. We show that 1-HPA, with acceptance threshold $1/2$ (in the finite and infinite word cases) can recognize non-regular languages, and the non-emptiness and non-universality problems for 1-HPA with threshold $1/2$ are decidable in **EXPTIME** and are **PSPACE**-hard. We present a new sufficient condition when 1-HPA recognize regular languages. We show that the emptiness problem is undecidable for 2-HPAs.

Keywords: Hierarchical Probabilistic automata, regular languages, decidability, emptiness problem, universality problem

1. Introduction

Probabilistic automata (PA) [1, 2, 3, 4] are finite-state machines that have probabilistic transitions on input symbols. Such machines can either recognize a language of finite words (probabilistic finite automata PFA [1, 2]) or a language of infinite words (probabilistic Büchi/Rabin/Muller automata [3, 4, 5]) depending on the notion of accepting run; on finite input words, an accepting run is one that reaches a final state, while on an infinite input, an accepting run is one whose set of states visited infinitely often satisfy a Büchi, Rabin, or Muller acceptance condition. The set of accepting runs in all these cases can be shown to be measurable and the probability of this set is taken to be probability of accepting the input word. Given an acceptance threshold x, the language $\mathsf{L}_{>x}(\mathcal{A})$ ($\mathsf{L}_{\geq x}(\mathcal{A})$) of a PA $\mathcal{A}$ is the set of all inputs whose acceptance probability is $> x$ ($\geq x$). In this paper the threshold x is always a rational number in $(0, 1)$.

Hierarchical probabilistic automata (HPA) are a syntactic subclass of probabilistic automata that are computationally more tractable for extremal thresholds [6] — problems of emptiness and universality which are undecidable for PA on infinite words with threshold 0 become decidable for HPA. Over finite words, the problem of deciding whether the infimum of acceptance probabilities is 0 also becomes decidable for HPA [7, 8], even though it is undecidable for general PA [9]. Intuitively, a k-HPA is a PA whose states are stratified into totally-ordered $k + 1$ levels with the property that

from any state q, on any input a, the machine can transition with non-zero probability to at most one state in the same level as q, and all other probabilistic successors belong to a higher level. Such automata arise naturally as models of *client-server systems*. Consider such a system where clients can request services of multiple servers that can fail (catastrophically) with some probability. The state of the automaton models the global state of all the servers and inputs to the machine correspond to requests from the client to the servers. The levels of the automaton correspond to the number of failed servers, with the lowest level modeling no failures. Since failed servers can't come back, the transitions in such a system satisfy the hierarchical nature.

While HPAs are tractable with extremal thresholds [10], the emptiness and universality problems were shown to be undecidable for 6-HPA with threshold $\frac{1}{2}$ [10]. In this paper, we investigate how the landscape changes when we restrict our attention to HPAs at low level. Specifically, we study the expressiveness and the complexity of decision problems for 1-HPA and 2-HPAs.

1-HPA (henceforth simply called HPA) are machines whose states are partitioned into two levels (0 and 1), with initial state in level 0, and transitions satisfying the hierarchical structure. These automata model client-server systems where only one server failure is allowed. Despite their extremely simple structure, we show that 1-HPA turn out to be surprisingly powerful — they can recognize non-regular languages over finite and infinite words (even with threshold $\frac{1}{2}$). This result is significant because all earlier constructions of PFA [2, 1] and probabilistic Büchi automata [4, 11] recognizing non-regular languages use either more complex automata or irrational acceptance thresholds or HPA with more than two levels. Moreover, this result is also unexpected because it was previously shown that *simple probabilistic automata* only recognize regular languages [10, 6]. The only difference between 1-HPA and simple PA is that all accepting states of a simple PA are required to be in level 0 (same level as the initial state).

Next, we consider the canonical decision problems of emptiness and universality for 1-HPA with threshold x. Decision problems for PA with non-extremal thresholds are often computationally harder than similar questions when the threshold is extremal (either 0 or 1), and the problems are always undecidable [12, 6, 11, 2]. Even though 1-HPA are expressive, we show that both emptiness and universality problems for 1-HPA are decidable in **EXPTIME** and are **PSPACE**-hard. As far as we know, this is the first decidability result for any subclass of PA with non-extremal thresholds that can recognize non-regular languages. Our decision procedure relies on observing that when the language of an HPA $\mathcal{A}$ is non-empty (or non-universal), then there is an input whose length is exponentially bounded in the size of the HPA that witnesses this fact.

We introduce a special subclass of 1-HPA called *integer* HPA. Integer HPA are HPA where from any level 0 state q, on any input a, the probability of transitioning to a level 1 state is an integer multiple of the probability of the unique transition to a level 0 state on a from q. With this restriction, we can show that integer HPA with threshold x only recognize regular languages (over finite and infinite words). For integer HPA, we show that the canonical decision decision problems of emptiness and universality are **PSPACE**-complete.

Finally, we consider the problem of checking emptiness of 2-HPAs. As mentioned above, undecidability was established for 6-HPAs in [10]. There, we adapted the proof

of undecidability of the problem of checking emptiness of probabilistic automata given in [12]. The key ingredient in the undecidability proof in [10, 12] is the design of a probabilistic automaton that recognize the language $\{a^n b^n \mid n \in \mathbb{N}\}$. The undecidability proof for 6-HPAs in [10] exploits the fact that this probabilistic automaton can be seen as a 5-HPA. The undecidability result in this paper relies on the construction of a probabilistic automaton that can recognize the same language using a 2-HPA.

The rest of the paper is organized as follows. Section 2 has basic definitions, and introduces HPA along with some useful propositions. The results characterizing the expressiveness and decidability of HPA are presented in Section 3. The results on integer HPA are presented in Section 4. The undecidability result on 2-HPA is proved in Section 5. Section 6 discusses related work and Secction 7 contains concluding remarks.

Note. An extended abstract of the paper appeared in the 16th International Conference on Foundations of Software Science and Computation Structures [13]. This paper contains all the proofs that were not included in [13]. In addition, this paper also includes the proof of undecidability result on 2-HPA which was established by Yue Ben and A. Prasad Sistla in the 13th International Symposium on Automated Technology for Verification and Analysis [14].

2. Preliminaries

We assume that the reader is familiar with probability measures, Markov Chains, finite-state automata, regular languages, Büchi automata, Muller automata and ω-regular languages. A sub-Markov Chain is a Markov chain in which the transition relation is a sub-probability measure. The set of natural numbers will be denoted by $\mathbb{N}$, the closed unit interval by $[0, 1]$ and the open unit interval by $(0, 1)$. The power-set of a set X will be denoted by 2^X.

Sequences. Given a finite set S, $|S|$ denotes the cardinality of S. Given a sequence (finite or infinite) $\kappa = s_0 s_1 \ldots$ over S, $|\kappa|$ will denote the length of the sequence (for infinite sequence $|\kappa|$ will be ω), and $\kappa[i]$ will denote the ith element s_i of the sequence with $\kappa[0]$ being the first. We will use ϵ to denote the (unique) empty string/sequence. For natural numbers i, j, $i \leq j < |\kappa|$, $\kappa[i : j]$ is the sequence $s_i \ldots s_j$. For $i < |\kappa|$, $\kappa[i : \infty]$ is the sequence $s_i s_{i+1} \ldots$ if $|\kappa| = \omega$, and is the sequence $s_i \ldots s_{|\kappa|-1}$ if $|\kappa|$ is finite. The set of *finite prefixes* of κ is the set $Pref(\kappa) = \{\kappa[0 : j] \mid j \in \mathbb{N}, j \leq |\kappa|\}$. As usual S^* will denote the set of all finite sequences/strings/words over S, S^+ will denote the set of all finite non-empty sequences/strings/words over S and S^ω will denote the set of all infinite sequences/strings/words over S. We will use u, v, w to range over elements of S^*, α, β, γ to range over infinite words over S^ω.

Given $u \in S^*$ and $\kappa \in S^* \cup S^\omega$, $u\kappa$ is the sequence obtained by concatenating the two sequences in order. Given $\mathsf{L}_1 \subseteq S^*$ and $\mathsf{L}_2 \subseteq S^* \cup S^\omega$, the set $\mathsf{L}_1\mathsf{L}_2$ is defined to be $\{u\kappa \mid u \in \mathsf{L}_1 \text{ and } \kappa \in \mathsf{L}_2\}$. Given $u \in S^+$, the word u^ω is the unique infinite sequence formed by repeating u infinitely often. An infinite word $\alpha \in S^\omega$ is said to be *ultimately* periodic if there are finite words $u \in S^*$ and $v \in S^+$ such that $\alpha = uv^\omega$. For an infinite word $\alpha \in S^\omega$, we write $\mathsf{inf}(\alpha) = \{s \in S \mid s = \alpha[i] \text{ for infinitely many } i\}$.

Languages. Given a finite alphabet Σ, a language L of finite words is a subset of Σ^*. A language L of infinite words over a finite alphabet Σ is a subset of Σ^ω. We consider only finite alphabets.

Probabilistic automaton (PA). Informally, a PA is like a finite-state deterministic automaton except that the transition function from a state on a given input is described as a probability distribution which determines the probability of the next state.

Definition 2.1. A *finite-state probabilistic automaton (*PA*)* over a finite alphabet Σ is a tuple $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\delta : Q \times \Sigma \times Q \to [0, 1]$ is the *transition relation* such that for all $q \in Q$ and $a \in \Sigma$, $\delta(q, a, q')$ is a non-negative rational number and $\sum_{q' \in Q} \delta(q, a, q') = 1$, and Acc is an *acceptance condition*. The acceptance condition shall depend on the kind of automata that we shall be considering and shall be defined later.

Notation: The transition function δ of PA $\mathcal{A}$ on input a can be seen as a square matrix δ_a of order $|Q|$ with the rows labeled by "current" state, columns labeled by "next state" and the entry $\delta_a(q, q')$ equal to $\delta(q, a, q')$. Given a word $u = a_0 a_1 \dots a_n \in \Sigma^+$, δ_u is the matrix product $\delta_{a_0} \delta_{a_1} \dots \delta_{a_n}$. For an empty word $\epsilon \in \Sigma^*$ we take δ_ϵ to be the identity matrix. Finally for any $Q_0 \subseteq Q$, we say that $\delta_u(q, Q_0) = \sum_{q' \in Q_0} \delta_u(q, q')$. Given a state $q \in Q$ and a word $u \in \Sigma^+$, $\mathsf{post}(q, u) = \{q' \mid \delta_u(q, q') > 0\}$. For a set $C \subseteq Q$, $post(C, u) = \cup_{q \in C}\, post(q, u)$.

Intuitively, the PA starts in the initial state q_0 and if after reading $a_0, a_1 \dots, a_i$ it is in state q, then the PA moves to state q' with probability $\delta_{a_{i+1}}(q, q')$ on symbol a_{i+1}. A *run* of the PA $\mathcal{A}$ starting in a state $q \in Q$ on an input $\kappa \in \Sigma^* \cup \Sigma^\omega$ is a sequence $\rho \in Q^* \cup Q^\omega$ such that $|\rho| = 1 + |\kappa|, \rho[0] = q$ and for each $i \geq 0$, $\delta_{\kappa[i]}(\rho[i], \rho[i+1]) > 0$. Unless otherwise stated, a *run* for us will mean a run starting in the initial state q_0.

Given a word $\kappa \in \Sigma^* \cup \Sigma^\omega$, the PA $\mathcal{A}$ can be thought of as a (possibly infinite-state) (sub)-Markov chain. The set of states of this (sub)-Markov Chain is the set $\{(q, v) \mid q \in Q, v \in Pref(\kappa)\}$ and the probability of transitioning from (q, v) to (q', u) is $\delta_a(q, q')$ if $u = va$ for some $a \in \Sigma$ and 0 otherwise. This gives rise to the standard σ-algebra on Q^ω defined using cylinders and the standard probability measure on (sub)-Markov chains [15, 16]. We shall henceforth denote the σ-algebra as $\mathcal{F}_{\mathcal{A},\kappa}$ and the probability measure as $\mu_{\mathcal{A},\kappa}$.

Acceptance conditions and PA **languages**. The language of a PA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ over an alphabet Σ is defined with respect to the acceptance condition Acc and a threshold $x \in [0, 1]$. We consider three kinds of acceptance conditions.

Finite acceptance: When defining languages over finite words, the acceptance condition Acc is given in terms of a finite set $Q_f \subseteq Q$. In this case we call the PA $\mathcal{A}$, a probabilistic finite automaton (PFA). Given a finite acceptance condition $Q_f \subseteq Q$ and a finite word $u \in \Sigma^*$, a run ρ of $\mathcal{A}$ on u is said to be accepting if the last state of ρ is in Q_f. The set of accepting runs on $u \in \Sigma^*$ is measurable [15] and we denote its measure by $\mu^{acc,\mathsf{f}}_{\mathcal{A},u}$. Note that $\mu^{acc,\mathsf{f}}_{\mathcal{A},u} = \delta_u(q_0, Q_f)$. Given a rational threshold $x \in [0, 1]$ and $\rhd \in \{\geq, >\}$, the language of finite words $\mathsf{L}^{\mathsf{f}}{}_{\rhd x}(\mathcal{A}) = \{u \in \Sigma^* \mid \mu^{acc,\mathsf{f}}_{\mathcal{A},u} \rhd x\}$ is the set of finite words accepted by $\mathcal{A}$ with probability $\rhd x$.

Büchi acceptance: Büchi acceptance condition defines languages over infinite words. For Büchi acceptance, the acceptance condition Acc is given in terms of a finite set $Q_f \subseteq Q$. In this case, we call the PA $\mathcal{A}$, a probabilistic Büchi automaton (PBA). Given a Büchi acceptance condition Q_f, a run ρ of $\mathcal{A}$ on an infinite word $\alpha \in \Sigma^\omega$ is said to be *accepting* if $\inf(\rho) \cap Q_f \neq \emptyset$. The set of accepting runs on $\alpha \in \Sigma^\omega$ is once again measurable [15] and we shall denote its measure by $\mu^{acc,\mathsf{b}}_{\mathcal{A},\alpha}$. Given a rational threshold $x \in [0,1]$ and $\rhd \in \{\geq, >\}$, the language of infinite words $\mathsf{L}^{\mathsf{b}}{}_{\rhd x}(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mu^{acc,\mathsf{b}}_{\mathcal{A},\alpha} \rhd x\}$ is the set of infinite words accepted by PBA $\mathcal{A}$ with probability $\rhd x$.

Muller acceptance: For Muller acceptance, the acceptance condition Acc is given in terms of a finite set $F \subseteq 2^Q$. In this case, we call the PA $\mathcal{A}$, a probabilistic Muller automaton (PMA). Given a Muller acceptance condition $F \subseteq 2^Q$, a run ρ of $\mathcal{A}$ on an infinite word $\alpha \in \mathcal{A}$ is said to be *accepting* if $\mathsf{inf}(\rho) \in F$. Once again, the set of accepting runs are measurable [15]. Given a word α, the measure of the set of accepting runs is denoted by $\mu^{acc,\mathsf{m}}_{\mathcal{A},\alpha}$. Given a threshold $x \in [0,1]$ and $\rhd \in \{\geq, >\}$, the language of infinite words $\mathsf{L}^{\mathsf{m}}{}_{\rhd x}(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mu^{acc,\mathsf{m}}_{\mathcal{A},\alpha} \rhd x\}$ is the set of infinite words accepted by PMA $\mathcal{A}$ with probability $\rhd x$.

2.1. Hierarchical Probabilistic Automata

Intuitively, a hierarchical probabilistic automaton is a PA such that the set of its states can be stratified into totally-ordered levels. From a state q on each letter a, the machine can transit with non-zero probability to at most one state in the same level as q, and all other probabilistic successors belong to a higher level. We define such automata for the special case when the states are partitioned into two levels (level 0 and level 1).

Definition 2.2. A 1-*level hierarchical probabilistic automaton* HPA is a probabilistic automaton $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ over alphabet Σ such that Q can be partitioned into two sets Q_0 and Q_1 with the following properties.

- $q_0 \in Q_0$,
- For every $q \in Q_0$ and $a \in \Sigma$, $|\mathsf{post}(q,a) \cap Q_0| \leq 1$
- For every $q \in Q_1$ and $a \in \Sigma$, $\mathsf{post}(q,a) \subseteq Q_1$ and $|\mathsf{post}(q,a)| = 1$.

Given a 1-level HPA $\mathcal{A}$, we will denote the level 0 and level 1 states by the sets Q_0 and Q_1 respectively.

Notation: For the rest of the paper, by an HPA we shall mean 1-HPA, unless otherwise stated.

Example 2.3. Consider the PA $\mathcal{A}_{\mathsf{int}}$, $\mathcal{A}_{\frac{1}{3}}$, and $\mathcal{A}_{\mathsf{Rabin}}$ shown in Figs. 1, 2, and 3 respectively.

All three automata have the same set of states ($\{q_0, \mathsf{q_{acc}}, \mathsf{q_{rej}}\}$), same initial state ($q_0$), same alphabet ($\{\mathbf{0}, \mathbf{1}\}$), the same acceptance condition ($Q_f = \{\mathsf{q_{acc}}\}$ if finite/Büchi, and $F = \{\{\mathsf{q_{acc}}\}\}$ if Muller) and the same transition structure. The only difference is in the probability of transitions out of q_0. All three of these automata are 1-HPA; we

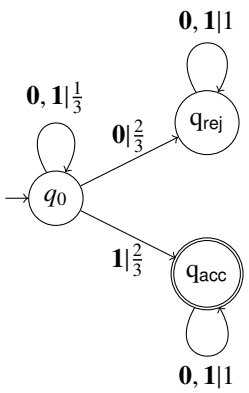

Figure 1: HPA $\mathcal{A}_{\mathsf{int}}$

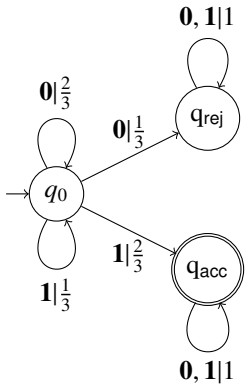

Figure 2: HPA $\mathcal{A}_{\frac{1}{3}}$

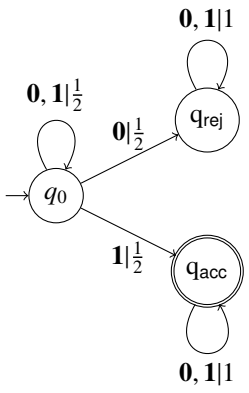

Figure 3: HPA $\mathcal{A}_{\mathrm{Rabin}}$

can take $Q_0 = \{q_0\}$, and $Q_1 = \{\mathsf{q_{acc}}, \mathsf{q_{rej}}\}$. Though all three are very similar automata, we will show that $\mathcal{A}_{\mathsf{int}}$ and $\mathcal{A}_{\mathsf{Rabin}}$ are symptomatic of automata that accept only regular languages (with rational thresholds), while the $\mathcal{A}_{\frac{1}{3}}$ accepts non-regular languages (with rational thresholds). The automaton $\mathcal{A}_{\mathsf{Rabin}}$ was originally presented in [1] and it is known to accept a non-regular language with an *irrational threshold* [1, 17]. Similarly it can be shown that $\mathcal{A}_{\mathsf{int}}$ also accepts a non-regular language with an irrational threshold.

Let us fix an HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ over alphabet Σ. Observe that given any state $q \in Q_0$ and any word $\kappa \in \Sigma^* \cup \Sigma^\omega$, $\mathcal{A}$ has at most one run ρ on α where all states in ρ belong to Q_0. We now present a couple of useful definitions. A set $W \subseteq Q$ is said to be a *witness* set if W has at most one level 0 state, i.e., $|W \cap Q_0| \leq 1$. Observe that for any word $u \in \Sigma^*$, $\mathsf{post}(q_0, u)$ is a witness set, i.e., $|\mathsf{post}(q_0, u) \cap Q_0| \leq 1$. We will say a word $\kappa \in \Sigma^* \cup \Sigma^\omega$ (depending on whether $\mathcal{A}$ is an automaton on finite or infinite words) is *definitely accepted* from witness set W iff for every $q \in W$ with $q \in Q_i$ (for $i \in \{0, 1\}$) there is an accepting run ρ on κ starting from q such that for every j, $\rho[j] \in Q_i$ and $\delta_{\kappa[j]}(\rho[j], \rho[j+1]) = 1$. In other words, κ is definitely accepted from witness set W if and only if κ is accepted from every state q in W by a run which only visits states at the same level as q, and all transitions in the run are taken with probability 1. Observe that the set of all words definitely accepted from a witness set W is regular.

Proposition 2.4. *For any* HPA $\mathcal{A}$ *and witness set* W*, the language*

$$\mathsf{L}_W = \{\kappa \mid \kappa \text{ is definitely accepted by } \mathcal{A} \text{ from } W\}$$

is regular. For any HPA $\mathcal{A}$ *and witness set* W*, the problem of checking the emptiness of* L_W *is in* **PSPACE**.

Proof. Observe that $\mathsf{L}_W = \cap_{q \in W} \mathsf{L}_{\{q\}}$ and $\mathsf{L}_\emptyset$ (as defined above) is the set of all strings. The language $\mathsf{L}_{\{q\}}$ is the language recognized by the deterministic automaton M_q defined as follows. M_q has the same set of states as $\mathcal{A}$ and q is its initial state. The

acceptance condition of M_q is the same as $\mathcal{A}$ .There is a transition from q_1 to q_2 in M_q on input a iff probability of transitioning from q_1 to q_2 in $\mathcal{A}$ is 1. Thus, L_W is the finite intersection of regular languages L_q, and its emptiness can be checked in **PSPACE**. □

Definition 2.5. A set W is said to be *good* if W is a witness set and the language L_W (defined in Proposition 2.4) is non-empty.

Example 2.6. Consider the HPAs $\mathcal{A}_{\mathsf{int}}, \mathcal{A}_{\frac{1}{3}}$ and $\mathcal{A}_{\mathrm{Rabin}}$ considered in Example 2.3. For each one of these automata, any subset of the set $\{q_0, \mathsf{q_{rej}}, \mathsf{q_{acc}}\}$ is a witness set. For each of these automata, the only non-empty good witness set is the set $\{\mathsf{q_{acc}}\}$. The language $\mathsf{L}_{\{\mathsf{q_{acc}}\}}$ is the set of all words.

For a set $C \subseteq Q_1$, a threshold $x \in (0, 1)$, and a word $u \in \Sigma^*$, we will find it useful to define the following quantity $\mathsf{val}(C, x, u)$ given as follows. If $\delta_u(q_0, Q_0) \neq 0$ then

$$\mathsf{val}(C, x, u) = \frac{x - \delta_u(q_0, C)}{\delta_u(q_0, Q_0)}.$$

On the other hand, if $\delta_u(q_0, Q_0) = 0$ then

$$\mathsf{val}(C, x, u) = \begin{cases} +\infty & \text{if } \delta_u(q_0, C) < x \\ 0 & \text{if } \delta_u(q_0, C) = x \\ -\infty & \text{if } \delta_u(q_0, C) > x \end{cases}.$$

The quantity $\mathsf{val}(C, x, u)$ measures the fraction of $\delta_u(q_0, Q_0)$ that still needs to move to C such that the probability of reaching C exceeds the threshold x. This intuition is captured by the following proposition whose proof follows immediately from the definition of $\mathsf{val}(C, x, u)$.

Proposition 2.7. *Consider an* HPA $\mathcal{A}$ *with threshold x, and words $u, v \in \Sigma^*$. Let $C, D \subseteq Q_1$ such that* $\mathsf{post}(C, v) = D$. *The following properties hold.*

- *If* $\mathsf{val}(C, x, u) < 0$ *then* $\delta_{uv}(q_0, D) > x$.
- *If* $\mathsf{val}(C, x, u) = 0$ *then* $\delta_u(q_0, C) = x$.

3. Expressiveness and decidability

1-HPA have a very simple transition structure. In spite of this, we will show that HPA can recognize non-regular languages (Section 3.1). Even though it has been shown before that PFA [2, 1] and PBA [4, 11] recognize non-regular languages, all the examples before, use either more complex automata or irrational acceptance thresholds or HPA with more than two levels. We then show that even though HPA can recognize non-regular languages, nevertheless the emptiness and universality problems of HPA are decidable (Section 3.2).

*3.1. Non-regular languages expressed by 1-*HPA

In this section, we show that HPA can recognize non-regular languages, under both finite acceptance and Büchi acceptance conditions. We consider a special type of HPA which we shall call *simple absorbing* HPA (SAHPA).

Definition 3.1. Let $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ be an HPA over an alphabet Σ with Q_0 and Q_1 as the sets of states at level 0 and 1 respectively. $\mathcal{A}$ is said to be a *simple absorbing* HPA (SAHPA) if

- $Q_0 = \{q_0\}, Q_1 = \{\mathsf{q_{acc}}, \mathsf{q_{rej}}\}$.
- The states $\mathsf{q_{acc}}, \mathsf{q_{rej}}$ are absorbing, i.e., for each $a \in \Sigma$, $\delta_a(\mathsf{q_{acc}}, \mathsf{q_{acc}}) = 1$ and $\delta_a(\mathsf{q_{rej}}, \mathsf{q_{rej}}) = 1$.

For a $\kappa \in \Sigma^* \cup \Sigma^\omega$, GoodRuns($\kappa$) is the set of runs ρ of $\mathcal{A}$ on κ such there is an $i \geq 0$ with $\rho(j) = \mathsf{q_{acc}}$ for all $i \leq j \leq |\kappa|$. A word $\alpha \in \Sigma^\omega$ is said to be *always alive* for $\mathcal{A}$ if for each $i > 0$, $\delta_{\alpha[0:i]}(q_0, q_0) > 0$.

Example 3.2. All three automata $\mathcal{A}_{\mathsf{int}}$, $\mathcal{A}_{\frac{1}{3}}$ and $\mathcal{A}_{\mathsf{Rabin}}$ (Example 2.3) shown in Figs. 1, 2, and 3 are simple absorbing HPA.

The following lemma states some important properties satisfied by SAHPA.

Lemma 3.3. *Let $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ be a* SAHPA *over an alphabet Σ with Q_0 and Q_1 as the sets of states at level* 0 *and* 1 *respectively. For any always alive $\alpha \in \Sigma^\omega$,*

1. *if α is ultimately periodic and $\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = x$ then the set $\{\mathsf{val}(\{\mathsf{q_{acc}}\}, x, \alpha[0:i]) \mid i \in \mathbb{N}, i \geq 0\}$ is a finite set,*
2. *if $\lim_{i\to\infty} \delta_{\alpha[0:i]}(q_0, q_0) = 0$ and $x \in (0,1)$ then $\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = x \Leftrightarrow \forall i \geq 0, \mathsf{val}(\{\mathsf{q_{acc}}\}, x, \alpha[0:i]) \in [0,1]$.*

Proof. We prove each of the parts of the lemma.

1. Fix an always alive $\alpha \in \Sigma^\omega$ such that $\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = x$. Consider a prefix $u \in \Sigma^*$ and $\gamma \in \Sigma^\omega$ such that $\alpha = u\gamma$. Now, it is easy to see that

$$\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = \delta_u(q_0, \mathsf{q_{acc}}) + \delta_u(q_0, q_0)\mu_{\mathcal{A},\gamma}(\text{GoodRuns}(\gamma)).$$

   Hence

$$\mu_{\mathcal{A},\gamma}(\text{GoodRuns}(\gamma)) = \frac{\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) - \delta_u(q_0, \mathsf{q_{acc}})}{\delta_u(q_0, q_0)}.$$

   As $\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = x$, we get that

$$\mu_{\mathcal{A},\gamma}(\text{GoodRuns}(\gamma)) = \mathsf{val}(\{\mathsf{q_{acc}}\}, u, x).$$

   Now, if α is ultimately periodic then the number of its suffixes is finite; (if it is periodic with period p starting from $n > 0$, then for all $i \geq n$, the suffix of α, starting from the i^{th} symbol is same as it's suffix starting from the $(i+p)^{th}$ symbol). The result follows.

2. ($\Rightarrow$) For each $i > 0$ let us write $\alpha[0 : i]$ as u_i and $\alpha[i+1 : \infty]$ as γ_i. If $\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = x$ then as in the first part of the lemma we have that for each i,

$$\mu_{\mathcal{A},\gamma_i}(\text{GoodRuns}(\gamma_i)) = \mathsf{val}(\{\mathsf{q_{acc}}\}, x, u_i).$$

Thus, we get $\mathsf{val}(\{\mathsf{q_{acc}}\}, x, u_i) \in [0,1]$ for each i.

($\Leftarrow$) Now assume that $\mathsf{val}(\{\mathsf{q_{acc}}\}, x, \alpha[0 : i])) \in [0,1]$ for all i. Note that since $\lim_{i\to\infty} \delta_{u_i}(q_0, q_0) = 0$, we get that $\mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) = \lim_{i\to\infty} \delta_{u_i}(q_0, \mathsf{q_{acc}})$. Let $y = \mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha))$. Now, if $y \neq x$ then we will get that $\lim_{i\to\infty}(x - \delta_{u_i}(q_0, \mathsf{q_{acc}})) = x - y \neq 0$. Now, observe that

$$\lim_{i\to\infty} \mathsf{val}(\{\mathsf{q_{acc}}\}, x, u_i) = \lim_{i\to\infty} \frac{(x - \delta_{u_i}(q_0, \mathsf{q_{acc}}))}{\delta_{u_i}(q_0, q_0)}.$$

Since $\lim_{i\to\infty}(x - \delta_{u_i}(q_0, \mathsf{q_{acc}})) \neq 0$ and $\lim_{i\to\infty} \delta_{u_i}(q_0, q_0) = 0$ we must get that there is an i such that $\mathsf{val}(\{\mathsf{q_{acc}}\}, x, u_i) > 1$ or $\mathsf{val}(\{\mathsf{q_{acc}}\}, x, u_i) < 0$, contradicting our assumption that $\mathsf{val}(\{\mathsf{q_{acc}}\}, x, \alpha[0 : i])) \in [0,1]$ for all i.

□

Now, we show that SAHPA can recognize non-regular languages. We start by recalling a result originally proved in [1]. Let $\Sigma = \{\mathbf{0}, \mathbf{1}\}$. Any word $\kappa \in \Sigma^* \cup \Sigma^\omega$ can be thought of as the binary representation of a number in the unit interval $[0,1]$ by placing a decimal in front of it. Formally,

Definition 3.4. Let $\Sigma = \{\mathbf{0}, \mathbf{1}\}$. The map $\mathsf{bin} : \Sigma^* \cup \Sigma^\omega \to [0,1]$ is the unique map such that $\mathsf{bin}(\epsilon) = 0$ and $\mathsf{bin}(a\kappa_1) = \frac{\bar{a}}{2} + \frac{1}{2}\mathsf{bin}(\kappa_1)$, where $\bar{a} = 0$ if $a = \mathbf{0}$ and 1 otherwise.

Note that $\mathsf{bin}(\alpha)$ is irrational iff α is an infinite word which is not ultimately periodic. The following is shown in [1].

Theorem 3.5. *$\Sigma = \{\mathbf{0}, \mathbf{1}\}$ and $\alpha \in \Sigma^\omega$ be a word which is not ultimately periodic. Given $\rhd \in \{>, \geq\}$,*

- *$\{u \in \Sigma^* \mid \mathsf{bin}(u) \rhd \mathsf{bin}(\alpha)\}$ is not regular.*
- *$\{\gamma \in \Sigma^\omega \mid \mathsf{bin}(\gamma) \rhd \mathsf{bin}(\alpha)\}$ is not ω-regular.*

We make some observations about the automaton $\mathcal{A}_{\frac{1}{3}}$ shown in Fig. 2 in Lemma 3.6.

Lemma 3.6. *Let $\mathcal{A}_{\frac{1}{3}}$ be the* SAHPA *over the alphabet $\Sigma = \{\mathbf{0}, \mathbf{1}\}$ defined in Example 2.3. Let $\alpha \in \Sigma^\omega$ be such that α is not an ultimately periodic word. We have that for each $\kappa \in \Sigma^* \cup \Sigma^\omega$,*

$$\mathsf{bin}(\kappa) < \mathsf{bin}(\alpha) \Leftrightarrow \mu_{\mathcal{A}_{\frac{1}{3}},\kappa}(\text{GoodRuns}(\kappa)) < \mu_{\mathcal{A}_{\frac{1}{3}},\alpha}(\text{GoodRuns}(\alpha))$$

and

$$\mathsf{bin}(\kappa) > \mathsf{bin}(\alpha) \Leftrightarrow \mu_{\mathcal{A}_{\frac{1}{3}},\kappa}(\text{GoodRuns}(\kappa)) > \mu_{\mathcal{A}_{\frac{1}{3}},\alpha}(\text{GoodRuns}(\alpha)).$$

Proof. Fix κ such that $\kappa \neq \alpha$. Note that if κ is a prefix of α, the lemma follows immediately from construction of $\mathcal{A}_{\frac{1}{3}}$. Assume that κ is not a prefix of α. In this case there must be a word $u \in \Sigma^*$, letters $a, b \in \Sigma, \kappa_1 \in \Sigma^* \cup \Sigma^\omega, \alpha_1 \in \Sigma^\omega$ such that $a \neq b, \kappa = ua\kappa_1$ and $\alpha = ub\alpha_1$.

Note that α_1 contains infinite occurrences of $\mathbf{0}$ and $\mathbf{1}$ (as α is not ultimately periodic). Using this fact, it is easy to see that $0 < \mu_{\mathcal{A},\alpha_1}(\text{GoodRuns}(\alpha_1)) < 1$. Also, note that we have by construction of $\mathcal{A}_{\frac{1}{3}}$, $\delta_u(q_0, q_0) > 0$.

Assume first $a = \mathbf{0}$ and $b = \mathbf{1}$. Then we will have $\mathsf{bin}(\kappa) < \mathsf{bin}(\alpha)$. It is easy to see that

$$\begin{aligned} \mu_{\mathcal{A},\kappa}(\text{GoodRuns}(\kappa)) &= \delta_u(q_0, \mathsf{q_{acc}}) + \tfrac{2}{3}\delta_u(q_0, q_0)\mu_{\mathcal{A},\kappa_1}(\text{GoodRuns}(\kappa_1)) \\ &\leq \delta_u(q_0, \mathsf{q_{acc}}) + \tfrac{2}{3}\delta_u(q_0, q_0) \end{aligned}$$

and

$$\begin{aligned} \mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha)) &= \delta_u(q_0, \mathsf{q_{acc}}) + \tfrac{2}{3}\delta_u(q_0, q_0) + \\ &\qquad \tfrac{1}{3}\delta_u(q_0, q_0)\mu_{\mathcal{A},\alpha_1}(\text{GoodRuns}(\alpha_1)) \\ &> \delta_u(q_0, \mathsf{q_{acc}}) + \tfrac{2}{3}\delta_u(q_0, q_0) \\ &> \mu_{\mathcal{A},\kappa}(\text{GoodRuns}(\kappa)). \end{aligned}$$

Similarly, we can show that if $a = \mathbf{1}$ and $b = \mathbf{0}$ we have that $\mathsf{bin}(\kappa) > \mathsf{bin}(\alpha)$ and $\mu_{\mathcal{A},\kappa}(\text{GoodRuns}(\kappa)) > \mu_{\mathcal{A},\alpha}(\text{GoodRuns}(\alpha))$. The result follows. □

We have:

Theorem 3.7. *Consider the* SAHPA $\mathcal{A}_{\frac{1}{3}}$ *over the alphabet* $\Sigma = \{\mathbf{0}, \mathbf{1}\}$ *defined in Example 2.3. Consider the finite acceptance condition and the Büchi acceptance condition defined by setting* $\mathsf{Acc} = \{\mathsf{q_{acc}}\}$. *Given* $\triangleright \in \{>, \geq\}$*, we have that the language of finite words* $\mathsf{L}^{\mathsf{f}}_{\triangleright \frac{1}{2}}(\mathcal{A}_{\frac{1}{3}})$ *is not regular and the language of infinite words* $\mathsf{L}^{\mathsf{b}}_{\triangleright \frac{1}{2}}(\mathcal{A}_{\frac{1}{3}})$ *is not* ω*-regular.*

Proof. Given $u \in \Sigma^*$, we shall denote $\mathsf{val}(\{\mathsf{q_{acc}}\}, \frac{1}{2}, u)$ by val_u. We observe some properties of the value val_u.

Claim 1. *For any* $u \in \Sigma^*$,

- $\mathsf{val}_{u\mathbf{0}} = \frac{3}{2}\mathsf{val}_u$ *and* $\mathsf{val}_{u\mathbf{1}} = 3\mathsf{val}_u - 2$.
- *If* $\mathsf{val}_u \in [0, 1]$ *then it is of the form* $\frac{p}{2^i}$ *where* p *is an odd number and* $i - 1$ *is the number of occurrences of* $\mathbf{0}$ *in* u.
- $\mathsf{val}_u \notin \{0, 1, \frac{2}{3}\}$.

Proof. The first part of the claim follows from observing that $\delta_{u\mathbf{0}}(q_0, q_0) = \frac{2}{3}\delta_u(q_0, q_0)$, $\delta_{u\mathbf{0}}(q_0, \mathsf{q_{acc}}) = \delta_u(q_0, \mathsf{q_{acc}})$, $\delta_{u\mathbf{1}}(q_0, q_0) = \frac{1}{3}\delta_u(q_0, q_0)$ and that $\delta_{u\mathbf{1}}(q_0, \mathsf{q_{acc}}) = \delta_u(q_0, \mathsf{q_{acc}}) + \frac{2}{3}\delta_u(q_0, q_0)$. The second part can be shown easily by an induction on the length of u using the first part of the claim. (Observe that the base case is $\mathsf{bin}(\epsilon) = \frac{1}{2}$). The third part of the claim is an easy consequence of the second part. (End: Proof of Claim 1) □

We now show that there is exactly one word $\beta \in \Sigma^\omega$ such that $\mu_{\mathcal{A},\beta}(\text{GoodRuns}(\beta)) = \frac{1}{2}$. As each $\alpha \in \Sigma^\omega$ is always alive and $\lim_{i\to\infty} \delta_{\alpha[0:i]}(q_0, q_0) = 0$, it follows from Lemma 3.3 and Claim 1 that it suffices to show that there is exactly one word $\beta \in \Sigma^\omega$ such that $\forall i \geq 0,\ \mathsf{val}_{\beta[0:i]} \in (0, 1)$.

We prove this by constructing β, starting from the empty word and showing that it can be extended one letter at a time in exactly one way. Clearly, thanks to Claim 1, since $\mathsf{val}_{\mathbf{0}} = \frac{3}{4}$ and $\mathsf{val}_{\mathbf{1}} = -\frac{1}{2}$, $\beta[0]$ should be $\mathbf{0}$. Suppose we have constructed $\beta[0 : i]$. Now, thanks to Claim 1 if $0 < \mathsf{val}_{\beta[0:i]} < \frac{2}{3}$ then $0 < \mathsf{val}_{\beta[0:i]\mathbf{0}} < \frac{3}{2}(\frac{2}{3}) = 1$ and $\mathsf{val}_{\beta[0:i]\mathbf{1}} < 3(\frac{2}{3}) - 2 < 0$. If $\frac{2}{3} < \mathsf{val}_{\beta[0:i]} < 1$ then $\mathsf{val}_{\beta[0:i]\mathbf{0}} > \frac{3}{2}(\frac{2}{3}) = 1$ and $0 = 3(\frac{2}{3}) - 2 < \mathsf{val}_{\beta[0:i]\mathbf{1}} < 3(1) - 2 = 1$. Thus if $\mathsf{val}_{\beta[0:i]} < \frac{2}{3}$ then $\beta[i+1]$ has to be $\mathbf{0}$, otherwise $\beta[i+1]$ has to be $\mathbf{1}$. Thus, we see that there is exactly one word $\beta \in \Sigma^\omega$ such that $\mu_{\mathcal{A},\beta}(\text{GoodRuns}(\beta)) = \frac{1}{2}$. We shall now show that the values $\mathsf{val}_{\beta[0:i]}$ are all distinct.

Claim 2. *For each i, j such that $i \neq j$, $\mathsf{val}_{\beta[0:i]} \neq \mathsf{val}_{\beta[0:j]}$.*

Proof. Fix i, j. Without loss of generality, we can assume that $j > i$. Note that thanks to Claim 1 that if there is an occurrence of $\mathbf{0}$ in $\beta[i + 1 : j]$ then $\mathsf{val}_{\beta[0:i]} \neq \mathsf{val}_{\beta[0:j]}$. If there is no occurrence of $\mathbf{0}$ in $\beta[i + 1 : j]$ then every letter of $\beta[i + 1 : j]$ must be a $\mathbf{1}$. Thus, the result will follow if we can show that for each $i + 1 \leq k < j$, we have that $\mathsf{val}_{\beta[1:k]\mathbf{1}} < \mathsf{val}_{\beta[1:k]}$. Using Claim 1, we have that

$$\mathsf{val}_{\beta[1:k]\mathbf{1}} < \mathsf{val}_{\beta[1:k]} \Leftrightarrow 3\mathsf{val}_{\beta[1:k]} - 2 < \mathsf{val}_{\beta[1:k]} \Leftrightarrow \mathsf{val}_{\beta[1:k]} < 1.$$

Now $\mathsf{val}_{\beta[1:k]} < 1$ by construction of β. The claim follows. (End: Proof of Claim 2) □

Now, thanks to Lemma 3.3 and Claim 2, we have that β is not ultimately periodic. The result follows from Lemma 3.6 and Theorem 3.5. □

Remark: Note that since any Büchi acceptance condition can be converted into an equivalent Muller acceptance condition, HPA also recognize non-regular languages under Muller acceptance conditions.

3.2. Decision problems for 1-HPA

We now show that the problems of checking emptiness and universality for HPA are decidable; more specifically, they are in **EXPTIME**.

Notation: For a HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ and $q \in Q$, let $\mathcal{A}_q = (Q, q, \delta, \mathsf{Acc})$ be the HPA that is the same as $\mathcal{A}$ except that it starts at q.

Recall that a word κ is *definitely accepted* from witness set W iff for every $q \in W$ with $q \in Q_i$ (for $i \in \{0, 1\}$) there is an accepting run ρ on κ starting from q such that for every $j < |\kappa|$, $\rho[j] \in Q_i$ and $\delta_{\kappa[j]}(\rho[j], \rho[j + 1]) = 1$. We need the following technical lemma in the proofs of our upper bounds. This lemma is a version of the pumping lemma for HPAs.

Lemma 3.8. *Let $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ be an* HPA *over an alphabet Σ, $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$ and $s \in \Sigma^* \cup \Sigma^\omega$. For $k < |s|$, let $X_k \subseteq \mathsf{post}(q_0, s[0:k])$ be the set of all states in Q_1 from which $s[k+1:\infty]$ is definitely accepted. Further, assume that there are integers $i < j$ such that $\mathsf{post}(q_0, s[0:i]) \cap Q_0 = \mathsf{post}(q_0, s[0:j]) \cap Q_0 = q$, $X_i \neq \emptyset$ and $X_i = X_j$. Then, for $\rhd \in \{<, =, >\}$, for every $\ell > 0$, we have*

$$\mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_\ell} \rhd \mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_{\ell-1}} \text{ iff } z_2 \rhd y_2 \cdot z_3$$

where $s_\ell = s[0:i](s[i+1:j])^\ell s[j+1:\infty]$, $y_2 = \delta_{s[i+1:j]}(q, Q_1)$, $z_2 = \delta_{s[i+1:j]}(q, X_i)$ and $z_3 = \mu^{acc,\mathsf{a}}_{\mathcal{A}_q,\, s[j+1:\infty]}$.

In the statement of the above lemma, when $y_2 > 0$, it can easily be shown that $\frac{z_2}{y_2}$ denotes the probability transferred to the states in X_i from state q when the input is the infinite sequence $(s[i+1:j])^\omega$ and z_3 is the probability transferred from state q by the rest of the sequence coming after j, i.e., $s[j+1:\infty]$. The lemma states that the probability of acceptance of s_ℓ is monotonically increasing or decreasing with increasing values of ℓ depending on whether $\frac{z_2}{y_2} > z_3$ or $\frac{z_2}{y_2} < z_3$.

Proof. Observe that $z_2 \leq y_2$. Suppose $y_2 = 0$. In this case, $z_2 = 0$ and hence $z_2 = y_2 \cdot z_3$. It is also easy to see that in this case, for every $\ell > 0$, $\mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_\ell} = \mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_{\ell-1}}$. Hence the lemma holds. Now consider the case when $y_2 > 0$. For $\ell \geq 0$, let $u_\ell = s[0:i](s[i+1:j])^\ell$. Observe that $s_\ell = u_\ell s[j+1:\infty]$. For $\ell > 0$, we have

$$\mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_\ell} \;=\; \delta_{u_{(\ell-1)}}(q_0, X_i) + (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot z_2 + (1 - \delta_{u_\ell}(q_0, Q_1)) \cdot z_3.$$

For $\ell > 0$, since $s_{\ell-1} = u_{\ell-1} s[j+1:\infty]$, we have

$$\mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_{\ell-1}} \;=\; \delta_{u_{(\ell-1)}}(q_0, X_i) + (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot z_3. \tag{1}$$

Therefore,

$$\begin{aligned}\mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_\ell} - \mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_{\ell-1}} \;=\;& (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot z_2 - \\ & (\delta_{u_\ell}(q_0, Q_1) - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot z_3.\end{aligned}$$

In addition,

$$\delta_{u_\ell}(q_0, Q_1) \;=\; \delta_{u_{(\ell-1)}}(q_0, Q_1) + (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot y_2$$

and hence

$$\delta_{u_\ell}(q_0, Q_1) - \delta_{u_{(\ell-1)}}(q_0, Q_1) \;=\; (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot y_2.$$

Putting all the above together, we get for all $\ell > 0$,

$$\mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_\ell} - \mu^{acc,\mathsf{a}}_{\mathcal{A},\, s_{\ell-1}} \;=\; (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) \cdot (z_2 \;-\; y_2 \cdot z_3).$$

Since $(1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) = \delta_{u_{(\ell-1)}}(q_0, Q_0) > 0$, the lemma follows trivially from the above equation. □

Example 3.9. Consider the PFA $\mathcal{A}_{\frac{1}{3}}$ given in Figure 2. Let s be the word **0101**. Observe that $\mathsf{post}(q_0, s[0:i]) \cap Q_1 = \{q_0\}$ for each i. It is easy to see that $X_0 = \emptyset$ and $X_1 = X_2 = X_3 = \{\mathsf{q_{acc}}\}$ where X_i is as defined in Lemma 3.8. Now, $i = 1$ and $j = 2$ satisfy the conditions of the Lemma 3.8. For $i = 1$, $j = 2$, s_ℓ as defined in Lemma 3.8 is the word $\mathbf{010^\ell 1}$ and the probability that $\mathcal{A}_{\frac{1}{3}}$ accepts s_ℓ is $\frac{2}{3} \times \frac{2}{3} + \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3}^\ell \times \frac{2}{3}$ which strictly monotonically decreases with ℓ. The quantities y_2, z_2, z_3 as defined in Lemma 3.8 are $\frac{1}{3}$, 0 and $\frac{2}{3}$ respectively. Thus, we have $z_2 < y_2 z_3$.

Upper bound for Emptiness Problem and strict thresholds. We start by considering emptiness for the language $\mathsf{L}^{a}_{>x}(\mathcal{A})$ for an HPA $\mathcal{A}$. In order to construct the decision procedure for this language, we need to consider special kinds of witness sets. Recall that a witness set W is good if the language L_W is non-empty (Definition 2.5). The decision procedure for emptiness relies on the following observation.

Lemma 3.10. *Let $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ be an* HPA *with n states (i.e., $|Q| = n$) such that all the transition probabilities of $\mathcal{A}$ have size at most r* [1]*. Let $x \in [0,1]$ be a rational threshold of size at most r. For any $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$, $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) \neq \emptyset$ iff there is a finite word u and a good non-empty set H, such that $|u| \leq 4rn8^n$ and $\delta_u(q_0, H) > x$.*

Proof. Observe that if there is a finite word u and a good non-empty set H such that $\delta_u(q_0, H) > x$ then there is a $\kappa \in \mathsf{L}_H$ (because H is good) and therefore $u\kappa \in \mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$. Thus, we only need to prove that nonemptiness of $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$ guarantees the existence of u and H as in the lemma.

Let $\mathsf{gwords} = \{(s, G) \mid G \neq \emptyset, G \text{ is good and } \delta_s(q_0, G) > x\}$. We first observe that gwords is non-empty. Since $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) \neq \emptyset$, there is $\kappa \in \mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$. If $\mathsf{a} = \mathsf{f}$ then observe that $(\kappa, (\mathsf{post}(q_0, \kappa) \cap \mathsf{Acc})) \in \mathsf{gwords}$. Otherwise, let us define $G_i = \{q \in \mathsf{post}(q_0, \kappa[0 : i] \mid \kappa[i+1 : \infty] \text{ is definitely accepted from } q\}$, and $x_i = \delta_{\kappa[0:i]}(q_0, G_i)$. Observe that x_i is a non-decreasing sequence, and there must be j, such that $x_j > x$; otherwise, κ cannot be accepted with probability $> x$. Then $(\kappa[0 : j], G_j) \in \mathsf{gwords}$. Fix $(s, G) \in \mathsf{gwords}$ such that for every $(s_1, G_1) \in \mathsf{gwords}$, $|s| \leq |s_1|$, i.e., s is the shortest word appearing in a pair in gwords. Note if $|s| \leq 2^n$ then the lemma follows.

Let us consider the case when $|s| > 2^n$. Let $k_1 = |s| - 1$. Observe that by our notation, $s = s[0 : k_1]$. Now, for any $0 \leq i \leq k_1$, let $Y_i = \mathsf{post}(q_0, s[0 : i]) \cap Q_1$ and $X_i = \{q \in Y_i : \mathsf{post}(q, s[i+1 : k_1]) \subseteq G\}$. Note that $X_i \subseteq Y_i$ and is good. Since $|s| > 2^n$ and $\mathcal{A}$ has n states, there must be i, j with $i < j \leq k_1$ such $X_i = X_j$ and $\mathsf{post}(q_0, s[0 : i]) \cap Q_0 = \mathsf{post}(q_0, s[0 : j]) \cap Q_0$. If $\mathsf{post}(q_0, s[0 : i]) \cap Q_0 = \emptyset$ then it is easy to see that $(s[0 : i]s[j+1 : k_1], G) \in \mathsf{gwords}$ contradicting the fact that s is the shortest such word. Hence, fix j to be the smallest integer such that for some $i < j$, $X_i = X_j$ and $\mathsf{post}(q_0, s[0 : i]) \cap Q_0 = \mathsf{post}(q_0, s[0 : j]) \cap Q_0 \neq \emptyset$. Note that such integers i, j exist, are unique, $i < 2^n$ and $j - i \leq 2^n$. Let q be the unique state in $\mathsf{post}(q_0, s[0 : i]) \cap Q_0$. Let $s[0 : i] = v$, $s[i+1 : j] = w$, $s[j+1 : k_1] = t$; thus, $s = vwt$. Figure 4 illustrates $v, w, t, X_i, X_j, Y_i, Y_j$.

Now, let $z_1 = \delta_v(q_0, X_i)$ and $y_1 = \delta_v(q_0, Q_1)$. Similarly, let $z_2 = \delta_w(q, X_j)$, $y_2 = \delta_w(q, Q_1)$ and $z_3 = \delta_t(q, G)$. Since $X_i, X_j \subseteq Q_1$, $z_1 \leq y_1$ and $z_2 \leq y_2$. Also note that $|w| > 0$ by construction of j and that $y_2 = \delta_w(q, Q_1) > 0$ (by the minimality of length of s).

For any integer $\ell \geq 0$, let $u_\ell = vw^\ell$ and $s_\ell = u_\ell t$. Note that $u_0 = v$ and $s_1 = s$.

Since $s = s_1$ is the shortest word in gwords and $s_0 = vt$ is a strictly smaller word than s_1, we must have that $\delta_{s_0}(q_0, G) \leq x$ and hence $\delta_{s_1}(q_0, G) > \delta_{s_0}(q_0, G)$. Let $t' \in \Sigma^* \cup \Sigma^\omega$ be a sequence that is definitely accepted from all the states in G. For each $\ell \geq 0$, let $s'_\ell = s_\ell t'$. Clearly, $\mu^{acc,\mathsf{a}}_{\mathcal{A}, s'_1} - \mu^{acc,\mathsf{a}}_{\mathcal{A}, s'_0} > 0$. From Lemma 3.8, we have for all $\ell > 0$,

[1] We say a rational number s has size r iff there are integers m, n such that $s = \frac{m}{n}$ and the binary representation of m and n has at most r-bits.

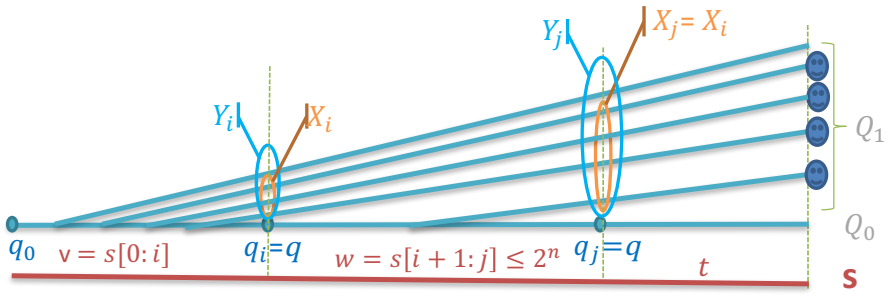

Figure 4: $v, w, t, X_i, X_j, Y_i, Y_j$ in the proof of Lemma 3.10.

$\mu^{acc,\mathsf{a}}_{\mathcal{A},s'_\ell} - \mu^{acc,\mathsf{a}}_{\mathcal{A},s'_{\ell-1}} > 0$. From this, it follows that

$$\text{for all } \ell > 0,\ \delta_{s_\ell}(q_0, G) > \delta_{s_{(\ell-1)}}(q_0, G). \tag{2}$$

Hence, $\lim_{\ell\to\infty} \delta_{s_\ell}(q_0, G)$ exists and is $\geq \delta_{s_1}(q_0, G)$. Hence, we get that $\lim_{\ell\to\infty} \delta_{s_\ell}(q_0, G) > x$ as $s_1 = s$.

Observe that $\delta_w(q, Q_1) > 0$. Hence, one can show that $\lim_{\ell\to\infty} (1 - \delta_{u_{(\ell-1)}}(q_0, Q_1)) = 0$. From this and the equation

$$\delta_{s_\ell}(q_0, G) = \delta_{u_\ell}(q_0, X_i) + (1 - \delta_{u_\ell}(q_0, Q_1)) \cdot z_3$$

we see that $\lim_{\ell\to\infty} \delta_{s_\ell}(q_0, G) = \lim_{\ell\to\infty} \delta_{u_\ell}(q_0, X_i)$. Observe that

$$\begin{aligned}
\delta_{u_\ell}(q_0, X_i) &= \delta_v(q_0, X_i) + \delta_v(q_0, q)(\textstyle\sum_{p=0}^{\ell-1} \delta_w(q, q)^p \delta_w(q, X_i)) \\
&= z_1 + (1 - y_1)(\textstyle\sum_{p=0}^{\ell-1} (1 - y_2)^\ell z_2) \\
&= z_1 + (1 - y_1) z_2 \textstyle\sum_{p=0}^{\ell-1} (1 - y_2)^\ell.
\end{aligned}$$

Thus, $\lim_{\ell\to\infty} \delta_{u_\ell}(q_0, X_i)$ is seen to be $z_1 + (1 - y_1) \cdot \frac{z_2}{y_2}$ and since $\lim_{\ell\to\infty} \delta_{s_\ell}(q_0, G) > x$, we get that $z_1 + (1 - y_1) \cdot \frac{z_2}{y_2} > x$.

Observe that X_i is a good set. Let m be the minimum ℓ such that $\delta_{u_\ell}(q_0, X_i) > x$. Such an m exists since $\lim_{\ell\to\infty} \delta_{u_\ell}(q_0, X_i) > x$. Now, we show that the length of u_m is bounded by $4rn8^n$ and hence the lemma is satisfied by taking u to be u_m and H to be X_i. Observe that

$$\delta_{u_\ell}(q_0, X_i) = z_1 + (1 - y_1) \cdot (1 - (1 - y_2)^\ell) \cdot \frac{z_2}{y_2}.$$

From this, we see that m is the minimum ℓ such that

$$(1 - y_2)^\ell < 1 - \frac{(x - z_1) y_2}{(1 - y_1) z_2}.$$

That is, m is the minimum ℓ such that $\ell > \frac{\log(n_1)}{\log(n_2)}$, where

$$n_1 = \frac{(1-y_1)z_2}{(1-y_1)z_2 - (x-z_1)y_2} \quad \text{and} \quad n_2 = \frac{1}{(1-y_2)}.$$

Now, observe that the probability of a run ρ of $\mathcal{A}$ starting from any state, on an input string of length at most 2^n is a product of 2^n fractions of the form $\frac{m_1}{m_2}$ where m_i, for $i = 1, 2$, is an integer bounded by 2^r. Hence the probability of such a run is itself a fraction whose numerator and denominator are bounded by 2^{r2^n}. Second, in an HPA with n states, on any input of length k, there are at most kn different runs; this is because once the run reaches a state in Q_1 the future is deterministic, and for any prefix, there is at most one run in a state in Q_0. Hence, $\delta_v(q_0, Q_1)$ is the sum of at most $n2^n$ such fractions. Therefore, y_1 is a fraction whose numerator and denominator are integers bounded by 2^{rn4^n}. By a similar argument, we see that z_1, y_2, z_2 are also fractions whose numerators and denominators are similarly bounded. Now, it should be easy to see that n_1 is bounded by 2^{4rn4^n} and hence $m \leq 4rn4^n$. Now, the length of $u_m = |vw| + (m-1)|w|$ which is easily seen to be bounded $m2^n$ since $|vw|$ and $|w|$ are bounded by 2^n. Hence $u_m \leq 4rn8^n$. □

Lemma 3.10 immediately yields a **NEXP** algorithm to check non-emptiness of the language of a HPA with strict thresholds: given a HPA $\mathcal{A}$, guess u and H satisfying the conditions in Lemma 3.10 and check that u and H do satisfy those conditions. We will now show that we can actually check non-emptiness (and hence emptiness) in **EXPTIME**.

Theorem 3.11. *Given an* HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$*, a rational threshold* $x \in [0, 1]$ *and* $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$*, the problem of determining if* $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) = \emptyset$ *is in* **EXPTIME**.

Proof. It suffices to show that the problem of determining if $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) \neq \emptyset$ is in **EXPTIME**. Let $\mathcal{X}$ be the collection of all witness sets U such that $U \cap Q_0 \neq \emptyset$ and $U \cap Q_1$ is a good set; for a witness set $U \in \mathcal{X}$, we will denote by q_U the unique state in $U \cap Q_0$. Let $\mathcal{Y}$ be the collection of good witness sets. For $U \in \mathcal{X}$ and natural number $i > 0$, let

$$\mathsf{Prob}(U, i) = \max\{\delta_u(q_U, W) \mid u \in \Sigma^*,\ W \in \mathcal{Y},\ \mathsf{post}(U \cap Q_1, u) \subseteq W,\ |u| \leq i\}.$$

In the above definition, we take the maximum of the empty set to be 0. Let k be the bound given by Lemma 3.10 for the length of the word u. Lemma 3.10 implies that $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) \neq \emptyset$ iff $\mathsf{Prob}(\{q_0\}, k) > x$. This observation yields a simple algorithm to check non-emptiness: compute $\mathsf{Prob}(\{q_0\}, k)$ and check if it is greater than x.

$\mathsf{Prob}(\cdot, \cdot)$ can be computed by an iterative dynamic programming algorithm as follows.

$$\begin{aligned}
\mathsf{Prob}(U, 1) &= \max\{\delta_a(q_U, W) \mid a \in \Sigma,\ W \in \mathcal{Y},\ \mathsf{post}(U \cap Q_1, a) \subseteq W\} \\
\mathsf{Prob}(U, i+1) &= \max\big(\{\mathsf{Prob}(U, i)\} \bigcup \\
&\qquad \{\delta_a(q_U, q_V)\mathsf{Prob}(V, i) + \delta_a(q_U, V \cap Q_1) \mid a \in \Sigma,\ V \in \mathcal{X}, \\
&\qquad\qquad \mathsf{post}(U \cap Q_1, a) \subseteq V\}\big).
\end{aligned}$$

Let us analyze the algorithm computing $\mathsf{Prob}(\cdot, \cdot)$. Let us assume that $\mathcal{A}$ has n states, and that $\delta_a(p, q)$ is of size at most r for any $a \in \Sigma$ and $p, q \in Q$. Thus, $\mathcal{X}$ and $\mathcal{Y}$ have cardinality at most 2^n, and by Proposition 2.4, the sets $\mathcal{X}$ and $\mathcal{Y}$ can be computed in **EXPTIME** (in fact, even in **PSPACE**). In addition, because $|\mathcal{X}|, |\mathcal{Y}| \leq 2^n$, the maximum in the above equations for computing Prob is over at most $O(2^n)$ terms. Thus, we would get an exponential time bound provided the arithmetic operations needed to compute Prob can also be carried out in exponential time. This requires us to bound

the size of the numbers involved in computing $\mathsf{Prob}(U, i)$. Observe that for any witness set W and $q \in Q$, $\delta_a(q, W)$ is the sum of at most n rational numbers and so has size at most $r + n$. Hence, we can inductively show that the size of $\mathsf{Prob}(U, i)$ (for any U) is a rational number of size at most $2i(r + n)$. Since $i \leq k$ and k is at most exponential in n (by Lemma 3.10), the dynamic programming algorithm is in **EXPTIME**. □

Upper bound for Emptiness problem and non-strict thresholds. Given a HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$, a threshold $x \in [0, 1]$ and $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$, we consider the problem of checking if the language $\mathsf{L}^{\mathsf{a}}{}_{\geq x}(\mathcal{A}) \neq \emptyset$. We have the Lemmas 3.13 and 3.14 that gives a necessary and sufficient conditions for checking the above problem. We will need Lemma 3.12 given below in order to establish Lemmas 3.13 and 3.14.

Lemma 3.12. *Consider an* HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ *over an alphabet* Σ*, a threshold* $x \in [0, 1]$ *and* $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$ *such that* $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) = \emptyset$ *and* $\kappa \in \mathsf{L}^{\mathsf{a}}{}_{\geq x}(\mathcal{A})$*. For* $k \leq |\kappa|$*, let* $X_k \subseteq \mathsf{post}(q_0, \kappa[0 : k])$ *be the set of all states from which* $\kappa[k + 1 : \infty]$ *is definitely accepted. If, in addition,* κ *is such that there are integers* $i < j$ *such that* $\mathsf{post}(q_0, \kappa[0 : i]) \cap Q_0 = \mathsf{post}(q_0, \kappa[0 : j]) \cap Q_0$*,* $X_i \neq \emptyset$ *and* $X_i = X_j$ *then for every* $\ell \geq 0$*,* $\mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell)} = x$*, where* $\kappa(\ell) = \kappa[0 : i](\kappa[i + 1 : j])^{\ell}\kappa[j + 1 : \infty]$.

Proof. Assume κ, i, j are as given in the statement of the lemma. Observe that $\kappa(1) = \kappa$. From Lemma 3.8, we see that (a) $\forall \ell > 0, \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell)} > \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell-1)}$, or (b) $\forall \ell > 0, \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell)} < \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell-1)}$ or (c) $\forall \ell > 0, \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell)} = \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell-1)}$. Case (a) implies that $\kappa(2) > \kappa(1) = x$ and case (b) implies that $\kappa(0) > \kappa(1) = x$. Both these cases contradict the assumption that $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) = \emptyset$. Hence case (c) holds which implies that $\forall \ell \geq 0\, \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa(\ell)} = x$. □

The following lemma gives necessary and sufficient conditions for checking whether the language $\mathsf{L}^{\mathsf{f}}{}_{\geq x}(\mathcal{A}) \neq \emptyset$ where $\mathcal{A}$ is a HPA over finite words.

Lemma 3.13. *Let* $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ *be a* HPA *over alphabet* Σ *such that* $|Q| = n$*. For a threshold* $x \in [0, 1]$ *and the language* $\mathsf{L}^{\mathsf{f}}{}_{\geq x}(\mathcal{A}) \neq \emptyset$ *iff one of the following conditions hold:*

- *either* $\mathsf{L}^{\mathsf{f}}{}_{>x}(\mathcal{A}) \neq \emptyset$
- *or there is an* $u \in \Sigma^*$ *such that* $|u| \leq 2^n$ *and a good set* G *such that* $\delta_u(q_0, G) = x$*.*

Proof. Without loss of generality, we can assume that $x > 0$ (otherwise, the lemma is trivially true). We prove the lemma in only one direction ($\Rightarrow$), as the other direction is trivial. Assume that $\mathsf{L}^{\mathsf{f}}{}_{\geq x}(\mathcal{A}) \neq \emptyset$ and $\mathsf{L}^{\mathsf{f}}{}_{>x}(\mathcal{A}) = \emptyset$.

Clearly $\exists u \in \Sigma^*$ and a good non-empty set G such that $\delta_u(q_0, G) = x$. Let u be the shortest such sequence. Now, if $|u| > 2^n$, then this would imply that there exist integers i, j such that $i < j$, $\mathsf{post}(q_0, u[0 : i]) \cap Q_0 = \mathsf{post}(q_0, u[0 : j]) \cap Q_0$, $X_i \neq \emptyset$ and $X_i = X_j$ where X_i, X_j are as defined in the statement of Lemma 3.12. Using Lemma 3.12 with $\ell = 0$, we get a shorter sequence satisfying the above property, a contradiction. □

The following lemma gives necessary and sufficient conditions for checking whether the language $\mathsf{L}^{\mathsf{a}}{}_{\geq x}(\mathcal{A}) \neq \emptyset$ where $\mathcal{A}$ is a HPA with Büchi or Muller acceptance conditions.

Lemma 3.14. *Let $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ be a* HPA *over alphabet Σ such that $|Q| = n$. For a threshold $x \in [0, 1]$ and $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$, the language $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) \neq \emptyset$ iff either $\mathsf{L}^{\mathsf{a}}_{>x}(\mathcal{A}) \neq \emptyset$ or at least one of the following two conditions is satisfied:*

(1) $\exists u \in \Sigma^*$ such that $|u| \leq 2^n$ and there is a good set G such that $\delta_u(q_0, G) = x$.

(2) $\exists u, v \in \Sigma^*$ such that $|u| \leq 2^n$, $|v| \leq 2^{(n+2)\log n}$ and $\alpha = uv^\omega$ is accepted by $\mathcal{A}$ with probability exactly x, i.e., $\mu^{acc,\mathsf{a}}_{\mathcal{A},\alpha} = x$.

Proof. Without loss of generality, we can assume that $x > 0$ (otherwise, the lemma is trivially true). We prove the lemma in only one direction ($\Rightarrow$), as the other direction is trivial. Assume that $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) \neq \emptyset$ and $\mathsf{L}^{\mathsf{a}}_{>x}(\mathcal{A}) = \emptyset$. We show that either (1) or (2) is satisfied. Now, if there is a finite sequence u and good set G such that $\delta_u(q_0, G) = x$, then the argument in Lemma 3.13 also applies here and condition (1) is satisfied.

Now, consider the case when Condition (1) is not satisfied. Now, there is an $\alpha \in \Sigma^\omega$ such that $\mu^{acc,\mathsf{a}}_{\mathcal{A},\alpha} = x$. Let L_1 be the set of all such α. Now, consider any $\alpha \in L_1$. Then there exist integers i, j such that $i < j$, and the following conditions hold; below we take $u = \alpha[0 : i]$ and $v = \alpha[i + 1 : j]$.

(a) $\mathsf{post}(q_0, u) \cap Q_0 = \mathsf{post}(q_0, uv) \cap Q_0$.

(b) For $k \in \mathbb{N}$, let us take $X_k = \{q \in \mathsf{post}(q_0, \alpha[0 : k]) | \alpha[k{+}1 : \infty]$ is definitely accepted from $q\}$ as in the statement of Lemma 3.12. Then $X_i \neq \emptyset$ and $X_i = X_j$.

(c) For every $q \in X_i$, $\mathsf{post}(q, v) = \{q\}$.

(d) For every $q \in X_i$, the set Y_q of all states that appear on the finite run from q on the string v satisfies the acceptance condition given by Acc, i.e., $Y_q \cap \mathsf{Acc} \neq \emptyset$ if $\mathsf{a} = \mathsf{b}$ and $Y_q \in \mathsf{Acc}$ if $\mathsf{a} = \mathsf{m}$.

(e) $\mathsf{post}(q_0, u) \cap Q_0 \neq \emptyset$. Let $\{q_1\} = \mathsf{post}(q_0, u) \cap Q_0 = \mathsf{post}(q_0, v) \cap Q_0$.

(f) $\delta_v(q_1, X_i) > 0, \delta_v(q_1, q_1) < 1$ and $q_1 \notin X_i$.

Using Lemma 3.12, we see that for each $\ell \geq 0$, $\alpha_\ell = uv^\ell\alpha[j + 1 : \infty]$ is accepted with probability x.

Claim 3. *$\mu^{acc,\mathsf{a}}_{\mathcal{A},\beta} = x$ where $\beta = uv^\omega$.*

Proof. Recall that q_1 is the unique state in $\mathsf{post}(q_0, u) \cap Q_0$. Since $\delta_v(q_1, q_1) < 1$, it is easy to see that $\mu^{acc,\mathsf{a}}_{\mathcal{A},\beta} = \lim_{\ell\to\infty} \mu^{acc,\mathsf{a}}_{\mathcal{A},\alpha_\ell}$. Since, for each $\ell \geq 0$, $\mu^{acc,\mathsf{a}}_{\mathcal{A},\alpha_\ell} = x$, it follows that $\mu^{acc,\mathsf{a}}_{\mathcal{A},\beta} = x$.

□

Now, let M be the set of all quadruples $(u, v, X_i, post(q_0, u) \cap Q_0)$ where u, v, X_i are as defined above for some $\alpha \in L_1$. Now let (u, v, U, q_1) be a quadruple in M such that $|u| + |v|$ is the minimum for all such triples in M and $|u|$ be the minimum among all triples having this minimum value. Using the argument of Lemma 3.12, it is not difficult to show that $|u| \leq 2^n$. Now, we prove that $|v| \leq n^{n+2} = 2^{(n+2)\log n}$ through contradiction. Assume that $|v| > n^{n+2}$. Let $U = \{q_2, ..., q_m\}$ for some $m < n$. Let ρ_1 be

the run $q_1, q_1^1, q_1^2, \ldots$ of $\mathcal{A}$ on the input v starting from the state q_1 such that $q_i^i \in Q_0$ for each i. For $2 \leq p \leq m$, let ρ_p be the (unique) run of $\mathcal{A}$ on the input v starting from the state q_p. Observe that length of ρ_p is $|v| + 1$ and $\rho_p[0] = q_p$ for each $1 \leq p \leq m$.

For any k, $0 \leq k \leq |v|$, let w_k be the vector $(\rho_1[k], ..., \rho_m[k])$. Now, for any $r = (r_1, ..., r_m) \in Q^m$, let $I_r = \{k \mid w_k = r, 0 \leq k < |v|\}$. Since $|v| > n^{n+2}$ and the cardinality of Q^m is bounded by n^n (since $m \leq n$), it follows that there exists an $r \in Q^m$, such that I_r has more than n^2 elements in it. Fix such an $r \in Q^m$. Let $i_1 < i_2 < ... < i_k$ be all the elements in I_r. Since $k > n^2$, it is easy to see that there exists j', $1 \leq j' < k$, such that for every p, $1 \leq p \leq m$, every state that appears in the sequence of states $\rho_p[i_{j'}], \rho_p[i_{j'} + 1], ..., \rho_p[i_{j'+1} - 1]$ also appears somewhere else, outside this subsequence, in the same run ρ_p. Now let $k' = i_{j'}$, $k'' = i_{j'+1}$ and $v' = (v_0, v_1, ..., v_{k'-1}, v_{k''}, v_{k''+1}, ..., v_{|v|-1})$, i.e., v' is obtained by removing the subsequence $(v_{k'}, ..., v_{k''-1})$ from v. Observe that, for each $p = 1, ..., m$, the run of $\mathcal{A}$ starting from state q_p on the input v' has the same set of states appearing in it as the run ρ_p. Using this and Lemma 3.12 repeatedly, it can be easily seen that the triple $(u, v', U, q_1) \in M$ which contradicts the condition that $|u| + |v|$ is the minimum among all triples in M. □

Using Lemma 3.14, it is easy to give an algorithm, that is in **NEXP**, which checks if $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) \neq \emptyset$. We now give a non-trivial **EXPTIME** algorithm for this problem.

Theorem 3.15. *Given an* HPA $\mathcal{A}$*, a rational threshold* $x \in [0, 1]$ *and* $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$*, the problem of determining if* $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) = \emptyset$ *is in* **EXPTIME***.*

Proof. First observe that if $\mathsf{a} = \mathsf{f}$ then the proof of Lemma 3.10 and Lemma 3.14 shows that $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) \neq \emptyset$ iff there exists a good set G and finite word u such that $|u| \leq 2^n$ and $\delta_u(q_0, G) \geq x$. Theorem 3.15 is easily proved on lines similar to that of Theorem 3.11 for this case.

Now if $\mathsf{a} = \mathsf{b}$ or $\mathsf{a} = \mathsf{m}$ then we first check if either of the following two conditions hold:

(i) $\mathsf{L}^{\mathsf{a}}_{>x}(\mathcal{A}) \neq \emptyset$.

(ii) There exists a good non-empty set G and finite word u such that $|u| \leq 2^n$ and $\delta_u(q_0, G) \geq x$.

If either of the above two conditions hold then we can conclude that $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) \neq \emptyset$, and these conditions can be checked in **EXPTIME** using the above observations and Theorem 3.11. Otherwise, the proof of Lemma 3.14 shows that $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) \neq \emptyset$ iff there exists a set $\emptyset \neq C \subseteq Q_1$, a state $q_1 \in Q_0$ and words $u, v \in \Sigma^*$ such that $|u| \leq 2^n$, $|v| \leq 2^{(n+2)\log n}$ such that following hold:

1. $\mathsf{post}(q_0, u) \cap Q_0 = \{q_1\}$ and $\mathsf{post}(q_1, v) \cap Q_0 = \{q_1\}$.
2. For every $q \in C \cup \{q_1\}$, $\mathsf{post}(q, v) = \{q\}$.
3. For every $q \in C$, the set Y_q of all states that appear on the (unique) finite run from q on the string v satisfies the acceptance condition given by Acc, i.e., $Y_q \cap \mathsf{Acc} \neq \emptyset$ if $\mathsf{a} = \mathsf{b}$ and $Y_q \in \mathsf{Acc}$ if $\mathsf{a} = \mathsf{m}$.

4. $\mu^{acc,\mathsf{a}}_{\mathcal{A},\beta} \geq x$, where $\beta = uv^\omega$. This condition can be restated in a form that is more useful for checking. Let $\delta_u(q_0, q_1) = x_1, \delta_u(q_0, C) = z_1, \delta_v(q_1, C) = z_2, \delta_v(q_1, Q_1) = y_2$. Then as in the proof of Claim 3, we have that

$$\begin{aligned}\mu^{acc,\mathsf{a}}_{\mathcal{A},\beta} &= \delta_u(q_0, C) + \delta_u(q_0, q_1)\delta_v(q_1, C)(1 + \delta_v(q_1, q_1) + (\delta_v(q_1, q_1))^2 + \cdots) \\ &= \delta_u(q_0, C) + \delta_u(q_0, q_1)\tfrac{\delta_v(q_1,C)}{1-\delta_v(q_1,q_1)} \\ &= z_1 + x_1\tfrac{z_2}{y_2}.\end{aligned}$$

Therefore $\mu^{acc,\mathsf{a}}_{\mathcal{A},\beta} \geq x$ iff $z_1 + x_1\frac{z_2}{y_2} \geq x$. Observe that $\mathsf{val}(C, x, u) = \frac{x-z_1}{x_1}$. Hence this condition can be restated as

$$\frac{z_2}{y_2} \geq \mathsf{val}(C, x, u).$$

We have the following claim.

Claim 4.

$$\frac{z_2}{y_2} \geq \mathsf{val}(C, x, u) \Leftrightarrow \mathsf{val}(C, x, uv) \leq \mathsf{val}(C, x, u).$$

Proof. Recall that $\delta_u(q_0, q_1) = x_1, \delta_u(q_0, C) = z_1, \delta_v(q_1, C) = z_2, \delta_v(q_1, Q_1) = y_2$.

$$\begin{aligned}\mathsf{val}(C, x, uv) - \mathsf{val}(C, x, u) &= \tfrac{x-\delta_{uv}(q_0,C)}{\delta_{uv}(q_0,q_1)} - \tfrac{x-\delta_u(q_0,C)}{\delta_u(q_0,q_1)} \\ &= \tfrac{x-(\delta_u(q_0,C)+\delta_u(q_0,q_1)\delta_v(q_1,C))}{\delta_u(q_0,q_1)\delta_v(q_1,q_1)} - \tfrac{x-\delta_u(q_0,C)}{\delta_u(q_0,q_1)} \\ &= \tfrac{x(1-\delta_v(q_1,q_1))-\delta_u(q_0,C)(1-\delta_v(q_1,q_1))-\delta_u(q_0,q_1)\delta_v(q_1,C)}{\delta_u(q_0,q_1)\delta_v(q_1,q_1)} \\ &= \tfrac{xy_2-z_1y_2-x_1z_2}{x_1(1-y_2)}\end{aligned}$$

Also

$$\begin{aligned}\tfrac{z_2}{y_2} - \mathsf{val}(C, x, u) &= \tfrac{z_2}{y_2} - \tfrac{x-\delta_u(q_0,C)}{\delta_u(q_0,q_1)} \\ &= \tfrac{z_2}{y_2} - \tfrac{x-z_1}{x_1} \\ &= \tfrac{z_2x_1-xy_2+z_1y_2}{x_1y_2} \\ &= -\tfrac{(xy_2-z_1y_2-x_1z_2)}{x_1y_2}\end{aligned}$$

Hence $\frac{z_2}{y_2} - \mathsf{val}(C, x, u) \geq 0$ iff $\mathsf{val}(C, x, uv) - \mathsf{val}(C, x, u) \leq 0$. The claim follows. □

Suppose that there are $\emptyset \neq C \subseteq Q_1, q_1 \in Q_0, u, v \in \Sigma^*$, which satisfy the above conditions. Now, we make two more observations which will be useful in the algorithm.

(a) Having fixed C and q_1 above, we can always replace u by u_1 such that $|u_1| \leq 2^n$, $\mathsf{post}(q_0, u_1) \cap Q_0 = \{q_1\}$ and $\mathsf{val}(C, x, u_1) \leq \mathsf{val}(C, x, u)$. This is because the values of z_2, y_2 do not depend on u, but only on C and q_1. Hence, once C and q_1 has been fixed, we can take u to be the word that realizes the minimum of the set $\{\mathsf{val}(C, x, u_1) \mid |u_1| \leq 2^n, \mathsf{post}(q_0, u_1) \cap Q_0 = \{q_1\}\}$.

(b) Thanks to Claim 4 above, once we have picked u as described above, we can choose v to be the word such that $|v| \leq 2^{(n+2)\log n}$, v satisfies Conditions 2 and 3 above and minimizes $\mathsf{val}(C, x, uv)$. Then $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A})$ is non-empty iff

$$\mathsf{val}(C, x, uv) \leq \mathsf{val}(C, x, u).$$

Hence, it suffices to show that for any given C, q_1 we can check that u, v can be chosen according to (a) and (b) above.

In order to achieve this, we first compute using a dynamic algorithm the value $\mathsf{val}(C, x, u)$ for u chosen according to (a) above. This is carried out as follows. For each $D \subseteq Q_1$ ($D \neq \emptyset$), $q \in Q_0$, and integer $0 \leq i \leq 2^n$, we define the function

$$U(D, q, i) = \min\{\mathsf{val}(D, x, u) \mid |u| \leq i \text{ and } \mathsf{post}(q_0, u) \cap Q_0 = \{q\}\},$$

Here we take the minimum of an empty set to be ∞. We will compute the function $U(\cdot, \cdot, \cdot)$ iteratively through dynamic programming as follows.

$$U(D, q, 0) = \begin{cases} x & \text{if } q = q_0 \\ \infty & \text{otherwise} \end{cases}$$
$$U(D, q, i+1) = \min\left(\left\{\frac{U(\mathsf{pre}(D,a), q', i) - \delta_a(q', D)}{\delta_a(q', q)} \mid a \in \Sigma,\ q' \in Q_0,\ \delta_a(q', q) > 0\right\} \cup \{U(D, q, i)\}\right)$$

where $\mathsf{pre}(D, a) = \{r \in Q_1 | \delta_a(r, D) = 1\}$. Let us assume that the size of $\mathcal{A}$ is bounded by n; thus, n bounds the size of Q, the number of transitions, and the number of bits needed to represent the transition probabilities. Observe that the computation of $U(\cdot, \cdot, i+1)$ involves taking a minimum over $O(n)$ terms. Second, as observed in Theorem 3.11, $\delta_a(q', D)$ is a rational number whose size is bounded by $2n$, and we can inductively prove that $U(\cdot, \cdot, i)$ is a rational number requiring at most $4in$ bits to represent. Thus, all the arithmetic operations are bounded by exponential time. Further, since we have an exponential number of sub-problems in this dynamic programming formulation, we can conclude that $U(\cdot, \cdot, \cdot)$ for all possible arguments, can be computed in exponential time. For a fixed C_1 and q_1, all we need is $U(C, q_1, 2^n)$. In the above formulation we don't compute the actual string u itself. This is not needed to decide emptiness. But if we desire to produce a witness when the language is non-empty, we can easily produce the string u itself by maintaining an auxiliary parallel table $string(D, q, i)$ which is updated as follows. The value $string(D, q, i)$ is set to the empty string symbol when $i = 0, q = q_0$. For $i \geq 0$, $string(D, q, i+1)$ is set as follows: if $U(D, q, i+1)$ is same as $U(D, q, i)$ then $string(D, q, i+1)$ is set to be same as $string(D, q, i)$, otherwise the $string(D, q, i+1)$ is set to be the sequence $string(\mathsf{pre}(D, a), q', i)$ followed by a, where a, q' are the values that give the minimum value in the equation defining $U(D, q, i+1)$.

Now having computed the value $\mathsf{val}(C, x, u)$ for u chosen according to (a) above, we can similarly compute the value $\mathsf{val}(C, x, uv)$ for v chosen according to (b) above, using dynamic programming. To check Conditions 2 and 3, we need to know the states at the beginning and end of the run on input v, and the set of states visited in between. Therefore, we need to consider *partial functions* of type $f : Q_1 \to (Q_1 \times 2^{Q_1})$, where $f(q)$ is a pair (q', S) where q' will be the state the computation ends in and S is the set of states visited in the computation. For such a partial function f, $\mathsf{dom}(f)$ denotes the domain of f, and $\mathsf{rng}(f) = \{q' \in Q_1 \mid \exists q, S.\ f(q) = (q', S')\}$ denotes the set of states

that are the first component of elements in the range of f. In addition, if for a state $q \in \mathsf{dom}(f)$, $f(q) = (q', S)$ then $\mathsf{dest}_f(q)$ will be used to denote q', and $\mathsf{visit}_f(q)$ the set S. For $a \in \Sigma$, we will say that a partial function $g : Q_1 \to (Q_1 \times 2^{Q_1}) \in \mathsf{pre}(f, a)$ if $\mathsf{dom}(f) = \mathsf{dom}(g)$ and for every $q \in \mathsf{dom}(f)$, $\delta_a(\mathsf{dest}_g(q), \mathsf{dest}_f(q)) = 1$ and $\mathsf{visit}_f(q) = \mathsf{visit}_g(q) \cup \{\mathsf{dest}_f(q)\}$. Finally, given partial function $f : Q_1 \to (Q_1 \times 2^{Q_1})$, we say that $v \in \Sigma^*$ has *runs consistent with* f, if for every $q \in \mathsf{dom}(f)$ $\mathsf{post}(q, v) = \mathsf{dest}_f(q)$ and the set of states in the run starting from q on input v is $\mathsf{visit}_f(q)$. Having introduced all this notation, we are ready to present the algorithm. Given partial function $f : Q_1 \to (Q_1 \times 2^{Q_1})$, $q_1, q_2 \in Q_0$, and $0 \leq i \leq 2^{(n+2)\log n}$, we will compute the function

$$V(f, (q_1, q_2), i) = \min\{\mathsf{val}(\mathsf{rng}(f), x, uv) \mid |v| \leq i,\ \mathsf{post}(q_0, u) \cap Q_0 = \{q_1\},\ \mathsf{post}(q_1, v) \cap Q_0 = \{q_2\} \text{ and } v \text{ has runs consistent with } f\}.$$

As before, the minimum of the empty set is ∞. V will be computed by dynamic programming in a manner similar to U above. Iteratively,

$$V(f, (q_1, q_2), 0) = \begin{cases} U(\mathsf{dom}(f), q_1, 2^n) & \text{if } q_1 = q_2 \text{ and } \forall q \in \mathsf{dom}(f).\ f(q) = (q, \{q\}) \\ \infty & \text{otherwise} \end{cases}$$

$$V(f, (q_1, q_2), i+1) = \min(\left\{ \frac{V(g, (q_1, q_2'), i) - \delta_a(q_2', \mathsf{rng}(f))}{\delta_a(q_2', q_2)} \mid a \in \Sigma,\ q_2' \in Q_0,\ \delta_a(q_2', q_2) > 0 \text{ and } g \in \mathsf{pre}(f, a) \right\} \cup \{V(f, (q_1, q_2), i)\})$$

Like before one can argue that the number of bits for any $V(\cdot, \cdot, \cdot)$ have at most exponential bits, and the number of sub-problems is exponential. Therefore $V(\cdot, \cdot, \cdot)$ can be computed for all possible arguments in exponential time. Having computed these values, we need to check that there is a non-empty $C \subseteq Q_1$, $q_1 \in Q_0$ and a partial function $f : Q_1 \to (Q_1 \times 2^{Q_1})$ such that (i) $\mathsf{dom}(f) = C$, (ii) $\forall q \in \mathsf{dom}(f).\ \mathsf{dest}_f(q) = q$, (iii) $\forall q \in \mathsf{dom}(f).\ \mathsf{visit}_f(q)$ satisfies the acceptance condition Acc, and (iv) the following inequality holds

$$V(f, (q_1, q_1), 2^{(n+2)\log n}) \leq U(C, q_1, 2^n).$$

Since U and V are exponential sized tables, this check can also be done in **EXPTIME**. □

Lower Bound for the Emptiness Problem. We show that the problem of checking the emptiness of the language (of finite or infinite length words) of a HPA is **PSPACE**-hard. This is the content of the next theorem.

Theorem 3.16. *Given an* HPA $\mathcal{A}$, $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$, $\rhd \in \{>, \geq\}$, *the problem of determining if* $\mathsf{L}^{\mathsf{a}}_{\rhd \frac{1}{2}}(\mathcal{A}) \neq \emptyset$ *is* **PSPACE**-*hard.*

Proof. The following problem is well known to be **PSPACE**-complete [18]— Given a finite set of deterministic finite automata on finite words, determine if the intersection of their languages is non-empty. We will reduce this problem to ours.

Let $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_m$, where $m \geq 2$, be the given set of deterministic automata on finite strings over an input alphabet Σ'. For $i = 1, ..., m$, let $\mathcal{A}_i = (Q_i, q_i, \delta_i, F_i)$ where

q_i is the initial state, $\delta_i : Q_i \times \Sigma' \to Q_i$ is the transition function, and $F_i \subseteq Q_i$ is the set accepting states. We assume that the sets Q_i, $i = 1, ..., m$ are mutually disjoint.

Consider the HPA $\mathcal{A} = (Q, q_0, \delta, F)$ over $\Sigma = \Sigma' \cup \{a\}$ where a is not in Σ', defined as follows.

- $Q = \cup_{1 \leq i \leq m} Q_i \cup \{q_0, r\}$, where $q_0, r \notin \cup_{1 \leq i \leq m} Q_i$. q_0 is the only level 0 state; all states in $Q \setminus \{q_0\}$ are in level 1.
- $F = \cup_{1 \leq i \leq m} F_i$.
- Let $y = \frac{1}{2} - \frac{1}{4(m-1)}$. Now, we have the following transitions: $\delta(q_0, a, r) = y$ and $\delta(q_0, a, q_i) = \frac{1-y}{m}$ for $i = 1, ..., m$; $\delta(q_0, b, r) = 1$ for $\forall b \in \Sigma'$; for each $q \in Q_i, b \in \Sigma'$, $\delta(q, b, q') = 1$ iff $\delta_i(q, b) = q'$ and is 0 otherwise, and $\delta(q, a, r) = 1$. Finally, r is an absorbing state on all inputs.

Let $\rhd \in \{>, \geq\}$. It is fairly easy to see that for any $u \in (\Sigma')^*$, $\delta_{au}(q_0, F) \rhd \frac{1}{2}$ iff u is accepted by each $\mathcal{A}_i$ for $i = 1, \ldots m$; furthermore, for any $v \in \Sigma^*$ that does not start with a or that contains a after the first symbol, $\delta_v(q_0, F) = 0$. Hence, $\mathsf{L}^{\mathsf{f}}_{\rhd \frac{1}{2}}(\mathcal{A}) \neq \emptyset$ iff there is an input string u that is accepted by each $\mathcal{A}_i$, for $i = 1, \ldots m$.

In order to establish the lower bound for Büchi automata, we consider the alphabet $\Sigma_\omega = \Sigma \cup \{\tau\}$ where $\tau \notin \Sigma$. We construct an automaton $\mathcal{B} = (Q_\omega, q_0, \delta_\omega, \mathsf{Acc})$ on Σ_ω which is obtained from $\mathcal{A}$ as follows. The set of states Q_ω for $\mathcal{A}$ is $Q \cup \{\mathsf{good}\}$ where $\mathsf{good} \notin Q$; again, the initial state q_0 is the only level 0 state in Q_ω. Now, we have the following transitions: $\delta_\omega(q_0, \tau, r) = 1$, $\delta_\omega(q, b, q') = \delta(q, b, q')$ for each $q, q' \in Q$ and $b \in \Sigma$, $\delta_\omega(q, \tau, \mathsf{good}) = 1$ for each $q \in F$, $\delta_\omega(q, \tau, r) = 1$ for each $q \in Q \setminus F$, and good is an absorbing state. Let $\mathsf{Acc} = \{\mathsf{good}\}$. It is fairly easy to see that for any infinite word α over Σ_ω, $\alpha \in \mathsf{L}^{\mathsf{a}}{}_{\rhd x}(\mathcal{B})$ iff $\alpha = au\tau\beta$ for some u, β such that u is accepted by each $\mathcal{A}_i$ for $i = 1, \ldots m$.

For Muller automaton we just replace the acceptance condition Acc in $\mathcal{B}$ above with $\{\{\mathsf{good}\}\}$. □

Remark: The exact complexity of checking emptiness of a HPA $\mathcal{A}$ remains an open question. The difficulty in characterizing the exact complexity lies in the fact that a word of shortest length witnessing nonemptiness is exponential in the size of $\mathcal{A}$. This has two consequences: 1) the probability distribution on the states after a shortest word is input may not be representible in polynomial space, and 2) a shortest word may not be represented in polynomial space. Thus, a **PSPACE** algorithm that checks for emptiness seems difficult to obtain.

Observe that the lower bound proof of checking emptiness uses at least two input symbols. For unary automata, the emptiness problem turns out to be decidable in the fourth level of counting hierarchy, which is contained in **PSPACE**, and believed to contained strictly. In order to establish this result we recall some definitions and results. Recall that the complexity class **RP** consists of problems which can be solved using a randomized polynomial time algorithm that returns "yes" on yes-instances with probability at least $\frac{1}{2}$, and always returns "no" on no-instances. We know that **RP** is contained in **NP**. The complexity class **PP** is the class of decision problems for which

there are polynomial time randomized algorithms which answers "yes" with probability $> \frac{1}{2}$ on yes-instances, and answers "no" with probability $\geq \frac{1}{2}$ on no-instances. The counting hierarchy is a class of decision problems introduced by Wagner [19]. The 0-th level, $\mathbf{C}_0\mathbf{P}$, is defined as $\mathbf{P}$. The k-th level of the hierarchy is denoted by $\mathbf{C}_k\mathbf{P}$, and is and defined recursively as $\mathbf{C}_{k+1}\mathbf{P} = \mathbf{P}\mathbf{P}^{\mathbf{C}_k\mathbf{P}}$. The whole counting hierarchy is contained in **PSPACE**. The following is proved in [20]:

Lemma 3.17. *Given unary PFA A, a non-negative integer m in binary and a rational number x, the problem of checking:*

1. *if a^m is accepted with probability equal to x is in* **coRP**.
2. *if a^m is accepted with probability strictly greater than x lies in $\mathbf{P}^{\mathbf{C}_3\mathbf{P}}$.*

The following proposition shows that the problem of checking emptiness of HPAs lies in the $\mathbf{C}_4\mathbf{P}$ when the acceptance condition is finite and can be decided in polynomial time otherwise.

Proposition 3.18. *Given an* HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ *on a unary alphabet* $\Sigma = \{a\}$*, a rational threshold* $x \in [0, 1]$*,* $\rhd \in \{>, \geq\}$*, the problem of determining if* $\mathsf{L}^{\mathsf{f}}_{\rhd x}(\mathcal{A}) = \emptyset$ *is in* $\mathbf{coNP}^{\mathbf{C}_3\mathbf{P}}$*. Given if* $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$*, the problem of determining if* $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A}) = \emptyset$ *is in* $\mathbf{P}$.

Proof. Observe that Σ^ω contains exactly one word a^ω. If $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$, then the probability of accepting the word a^ω can be computed exactly in polynomial time [21] and hence the problem of determining if $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A}) = \emptyset$ is in $\mathbf{P}$.

We now consider the case when the acceptance condition of $\mathcal{A}$ is finite. It suffices to show that the problem of checking nonemptiness of $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A})$ is in $\mathbf{NP}^{\mathbf{C}_3\mathbf{P}}$. Thanks to Lemma 3.10 and Lemma 3.13, $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A}) \neq \emptyset$ if and only if there exists an m such that size of m in binary is polynomial in the size of $\mathcal{A}$ and the finite word a^m is accepted with probability $\rhd x$. The decision procedure for checking nonemptiness of $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A})$ first guesses m and then checks if a^m is accepted with probability $\rhd x$ using Lemma 3.17. The result follows. □

The Universality problem. Theorems 3.11 and 3.15 also give algorithms to check non-universality. Our next result establishes these connections and proves a **PSPACE** lower bound as well.

Theorem 3.19. *Given an* HPA $\mathcal{A}$*,* $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$*,* $\rhd \in \{>, \geq\}$*, the problem of checking the universality of language* $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A})$ *is in* **EXPTIME** *and is* **PSPACE***-hard.*

Proof. Let us begin by considering $\mathsf{a} = \mathsf{f}$. If $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ then in this case Acc is given by $Q_f \subseteq Q$. Consider $\mathcal{B} = (Q, q_0, \delta, Q \setminus Q_f)$. Observe that $\mathsf{L}^{\mathsf{f}}_{>x}(\mathcal{A}) = \Sigma^*$ iff $\mathsf{L}^{\mathsf{f}}_{\geq 1-x}(\mathcal{B}) = \emptyset$ and that $\mathsf{L}^{\mathsf{f}}_{\geq x}(\mathcal{A}) = \Sigma^*$ iff $\mathsf{L}^{\mathsf{f}}_{>1-x}(\mathcal{B}) = \emptyset$. The upper bound follows from Theorems 3.11 and 3.15, while the lower bound follows from Theorem 3.16.

We now establish the **EXPTIME** upper bound when $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$. Consider $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$. If $\mathsf{a} = \mathsf{b}$ then Acc is given by $Q_f \subseteq Q$ and take $F = 2^{Q \setminus Q_f}$. On the other hand, if $\mathsf{a} = \mathsf{m}$ then $\mathsf{Acc} \subseteq 2^Q$ and take $F = 2^Q \setminus \mathsf{Acc}$. In either case, consider the Muller automaton $\mathcal{B} = (Q, q_0, \delta, F)$. Observe that, for $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$, $\mathsf{L}^{\mathsf{a}}_{>x}(\mathcal{A}) = \Sigma^\omega$ iff $\mathsf{L}^{\mathsf{m}}_{\geq 1-x}(\mathcal{B}) = \emptyset$ and that $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) = \Sigma^\omega$ iff $\mathsf{L}^{\mathsf{m}}_{>1-x}(\mathcal{B}) = \emptyset$. The upper bound will

be established using Theorems 3.11 and 3.15, to check the emptiness of the Muller automaton $\mathcal{B}$. Now, $\mathcal{B}$ as defined above, maybe exponentially larger than $\mathcal{A}$ because of the size of its acceptance condition F. However, this does not change the complexity as F does not need to be explicitly constructed when checking the emptiness of $\mathcal{B}$. Recall that the algorithms in Theorems 3.11 and 3.15 use dynamic programming to compute 3 functions: $\mathsf{Prob}(\cdot,\cdot)$, $U(\cdot,\cdot,\cdot)$, and $V(\cdot,\cdot,\cdot)$. The algorithms to compute U and V (Theorem 3.15) do not depend on the acceptance condition of the automaton. After computing U and V, checking Condition 3 in the proof of Theorem 3.15 depends on the acceptance condition. However, to carry out this check, all we need to determine is which sets of states belong to F, which can be determined using Acc without constructing F explicitly. Next, the only step during the computation of Prob that depends on the acceptance condition F, is determining the set of all good witness sets. For a state $q \in Q$, and set $S \subseteq Q$, let $\mathcal{M}_{q,S}$ be the (non-probabilistic) Muller automaton $\mathcal{M}_{q,S} = (Q, q, \Delta, \{S\})$, where $(q_1, a, q_2) \in \Delta$ iff $\delta_a(q_1, q_2) = 1$. Observe that a witness set W is good for $\mathcal{B}$ iff for every $q \in W$, there is $S_q \in F$ such that

$$\cap_{q \in W} \mathsf{L}(\mathcal{M}_{q,S_q}) \neq \emptyset. \tag{3}$$

Thus, to check if W is good, we can go over all possible choices of S_q and check Equation (3) above in exponential time without explicitly constructing F.

We now prove the **PSPACE**-hardness in the case of $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$. The construction is similar to Theorem 3.16 with small differences. Once again we reduce the problem of checking the non-emptiness of the intersection of DFAs. Let $\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_m$ be DFA over alphabet Σ' with $\mathcal{A}_i = (Q_i, q_1, \delta_i, F_i)$. Again, we assume that the sets Q_i are mutually disjoint. We take the alphabet for the HPA to be $\Sigma = \Sigma' \cup \{a, \tau\}$, where $a, \tau \notin \Sigma'$. The HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ is defined as follows

- $Q = (\cup_{i=1}^{m} Q_i) \cup \{q_0, r, f_1, f_2\}$, where $(\cup_{i=1}^{m} Q_i) \cap \{q_0, r, f_1, f_2\} = \emptyset$. Again q_0 is the only level 0 state, with the remaining states at level 1.
- Let $x = \frac{1}{2} - \frac{1}{2(m+1)}$, and $y = \frac{1-x}{m+1}$. The states r and f_1 are absorbing, i.e., $\delta_c(q, q) = 1$ for $c \in \Sigma$ and $q \in \{r, f_1\}$. The remaining transitions are as follows.
  - $\delta_c(q_0, f_1) = 1$ if $c \neq a$. On the other hand, the transitions on a from q_0 are as follows: $\delta_a(q_0, f_1) = x$, $\delta_a(q_0, q_i) = \delta_a(q_0, f_2) = y$
  - For $q \in Q_i$, we have $\delta_a(q, f_1) = 1$. For $c \in \Sigma'$, $\delta_c(q, q') = 1$ iff $\delta_i(q, c) = q'$. Finally, $\delta_\tau(q, r) = 1$ if $q \in F_i$, and $\delta_\tau(q, f_1) = 1$ if $q \notin F_i$.
  - Finally, $\delta_c(f_2, f_2) = 1$ if $c \neq \tau$ and $\delta_\tau(f_2, r) = 1$.
- The acceptance condition Acc is defined as follows. If $\mathsf{a} = \mathsf{b}$ then $\mathsf{Acc} = \{f_1, f_2\}$, and if $\mathsf{a} = \mathsf{m}$ then $\mathsf{Acc} = \{\{f_1\}, \{f_2\}\}$.

The definition of Acc and the transition structure ensures that $\mathsf{L}^{\mathsf{b}}{}_{\rhd x}(\mathcal{A}) = \mathsf{L}^{\mathsf{m}}{}_{\rhd x}(\mathcal{A})$, for any x and $\rhd \in \{>, \geq\}$. We will now argue that $\cap_{i=1}^{m} \mathsf{L}(\mathcal{A}_i) = \emptyset$ iff $\mathsf{L}^{\mathsf{a}}{}_{\rhd \frac{1}{2}}(\mathcal{A})$ for $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$ and $\rhd \in \{>, \geq\}$. First observe that if $\alpha \notin (a\Sigma'^{*}\tau\Sigma^{\omega} \cup a\Sigma'^{\omega})$ then α is accepted with probability 1 by $\mathcal{A}$. If $\alpha \in a\Sigma'^{\omega}$ then α is accepted with probability $x + y > \frac{1}{2}$ because of the computations that go to f_1 and f_2 after reading a. Consider $\alpha = au\tau\beta$ for some

$u \in \Sigma'^*$ and $\beta \in \Sigma^\omega$. If $u \in \cap_i \mathsf{L}(\mathcal{A}_i)$ then α is accepted with probability x which $< \frac{1}{2}$. If $u \notin \cap_i \mathsf{L}(\mathcal{A}_i)$ then α is accepted with probability at least $x + y > \frac{1}{2}$. This establishes the lower bound. □

Once again, the lower bound proof of universality requires alphabet with two letters. For unary alphabet, the universality problem turns to decidable in the fourth level of counting hierarchy. This can be shown in a fashion similar to the proof of Proposition 3.18.

Proposition 3.20. *Given an* HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ *on a unary alphabet* $\Sigma = \{a\}$*, a rational threshold* $x \in [0, 1]$*,* $\rhd \in \{>, \geq\}$*, the problem of determining if* $\mathsf{L}^{\mathsf{f}}{}_{\rhd x}(\mathcal{A}) = \Sigma^*$ *is in* $\mathbf{coNP}^{\mathbf{C_3P}}$*. Given if* $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$*, the problem of determining* $\mathsf{L}^{\mathsf{a}}{}_{\rhd x}(\mathcal{A}) = \Sigma^\omega$ *is in* **P**.

4. Integer HPA

Previously we saw that even though 1-HPA have a very simple transition structure, their ability to toss coins allows them to recognize non-regular languages. In this section, we will show that if we restrict the numbers that appear as transition probabilities in the automaton, then the HPA can only recognize regular languages (see Theorem 4.7). We will also show that the problems of checking emptiness and universality of this class of HPA are **PSPACE**-complete (see Theorem 4.8). We will call this restricted class of HPA, integer HPA.

Definition 4.1. An *integer* HPA is a 1-HPA $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ over alphabet Σ with Q_0 and Q_1 being the level 0 and level 1 states, respectively, such that for every $q \in Q_0$ and $a \in \Sigma$, if $\mathsf{post}(q, a) \cap Q_0$ is non-empty and equal to $\{q'\}$, then for every $q'' \in Q_1$, $\delta_a(q, q'')$ is an integer multiple of $\delta_a(q, q')$.

Example 4.2. Consider automata $\mathcal{A}_{\mathsf{int}}$, $\mathcal{A}_{\frac{1}{3}}$, and $\mathcal{A}_{\mathsf{Rabin}}$ from Example 2.3 that are shown in Figs. 1, 2, and 3. Observe that $\mathcal{A}_{\mathsf{int}}$ and $\mathcal{A}_{\mathsf{Rabin}}$ are integer automata. On the other hand, $\mathcal{A}_{\frac{1}{3}}$, which was shown to accept non-regular languages in Section 3.1, is not an integer automaton. The reason is because of the transition from q_0 on symbol $\mathbf{0}$; $\delta_{\mathbf{0}}(q_0, \mathsf{q_{rej}}) = \frac{1}{3}$ is not an integer multiple of $\delta_{\mathbf{0}}(q_0, q_0) = \frac{2}{3}$.

The main result of this section is that for any integer HPA $\mathcal{A}$, and rational x, the language $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$ is regular (for $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$). The proof of this result will rest on observations made in the Proposition 4.3 below that states that a word κ is accepted exactly when a prefix of κ reaches a witness set with sufficient probability, and the rest of the word κ is definitely accepted from the witness set.

Proposition 4.3. *For an* HPA $\mathcal{A}$*, threshold* $x \in [0, 1]$*, and word* κ*,* $\kappa \in \mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$ *(where* $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$*) if and only if there is a non-empty witness set* $W \subseteq \mathsf{post}(q_0, u)$*,* $u \in \Sigma^*$ *and* $\kappa' \in \Sigma^* \cup \Sigma^\omega$ *such that* $\kappa = u\kappa'$*,* κ' *is definitely accepted by* $\mathcal{A}$ *from* W*, and one of the following holds.*

- *Either* $W \subseteq Q_1$ *and* $\mathsf{val}(W, x, u) < 0$*, or*
- $W \cap Q_0 \neq \emptyset$ *and* $0 \leq \mathsf{val}(W \cap Q_1, x, u) < 1$.

Proof. ($\Rightarrow$) Fix κ such that $\kappa \in \mathsf{L}^{\mathsf{a}}_{>x}(\mathcal{A})$.

If $a = \mathsf{f}$ then we let $u = \kappa$ and κ' be the empty word. Let the acceptance condition acc be given by the set Q_f. Let $W_0 = \mathsf{post}(q_0, u) \cap Q_0 \cap Q_f$ and $W_1 = \mathsf{post}(q_0, u) \cap Q_1 \cap Q_f$. Observe that

$$\mu^{acc,\mathsf{f}}_{\mathcal{A},\kappa} = \delta_u(q_0, W_0) + \delta_u(q_0, W_1).$$

If $\delta_u(q_0, W_1) > x$ then we can take $W = W_1$ and the proposition follows. Otherwise, $\delta_u(q_0, W_1) \leq x$. Thus, $\delta_u(q_0, W_0) \neq 0$ and hence W_0 must be non-empty. Let $W = W_0 \cup W_1$. As $\delta_u(q_0, W_1) \leq x$, $\mathsf{val}(W \cap Q_1, x, u) \geq 0$. Also observe that

$$\begin{array}{rcl} \mathsf{val}(W \cap Q_1, x, u) & = & \frac{x - \delta_u(q_0, W_1)}{\delta_u(q_0, Q_0)} \\ & < & \frac{\mu^{acc,\mathsf{f}}_{\mathcal{A},\kappa} - \delta_u(q_0, W_1)}{\delta_u(q_0, Q_0)} \\ & = & \frac{\delta_u(q_0, W_0)}{\delta_u(q_0, Q_0)} \\ & = & 1. \end{array}$$

The proposition follows.

Now assume that $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$. For each $j \in \mathbb{N}$, let $u_j = \kappa[0 : j]$ and $\kappa_j = \kappa[j + 1 : \infty]$. For each $j \in \mathbb{N}$, let $W_j = \{q \in \mathsf{post}(q_0, u_j) \cap Q_1 \mid \kappa_j \in \mathsf{L}_{\{q\}}\}$.

Observe that for each $j \geq 0$,

$$\delta_{u_j}(q_0, W_j) \leq \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa} \leq \delta_{u_j}(q_0, W_j) + \delta_{u_j}(q_0, Q_0).$$

Observe also that $\delta_{u_j}(q_0, W_j)$ is an increasing sequence and that $\delta_{u_j}(q_0, Q_0)$ is a decreasing sequence. Hence, $\lim_{j\to\infty} \delta_{u_j}(q_0, W_j)$ and $\lim_{j\to\infty} \delta_{u_j}(q_0, Q_0)$ exist. Furthermore,

$$\lim_{j\to\infty} \delta_{u_j}(q_0, W_j) \leq \mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa} \leq \lim_{j\to\infty} \delta_{u_j}(q_0, W_j) + \lim_{j\to\infty} \delta_{u_j}(q_0, Q_0).$$

There are two possibilities.

- There is a j_0 such that $\delta_{u_j}(q_0, W_{j_0}) > x$. In this case, fix one such j_0. The proposition follows by taking $u = u_{j_0}$, $\kappa' = \kappa_{j_0}$ and $W = W_{j_0}$.

- $\delta_{u_j}(q_0, W_j) \leq x$ for each j. From the fact that $\mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa} > x$, it is easy to see that in this case, we must have that $\lim_{j\to\infty} \delta_{u_j}(q_0, Q_0) > 0$.

  For each j, let q_j be the unique state in Q_0 such that $\delta_{u_j}(q_0, q_j) > 0$. Now, as $\lim_{j\to\infty} \delta_{u_j}(q_0, Q_0) > 0$, there must be a j_0 such that $\delta_{\kappa[j+1]}(q_j, q_{j+1}) = 1$ for each $j \geq j_0$. Fix j_0. Furthermore, it must be the case that the run $q_{j_0}, q_{j_0+1}, q_{j_0+2} \ldots$ must be an accepting run (otherwise, $\mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa}$ will be $\delta_{u_j}(q_0, W_j)$ which is $\leq x$). Observe that

  $$\mu^{acc,\mathsf{a}}_{\mathcal{A},\kappa} = \delta_{u_{j_0}}(q_0, q_{j_0}) + \delta_{u_{j_0}}(q_0, W_{j_0}).$$

  Let $u = u_{j_0}, \kappa' = \kappa_{j_0}$ and $W = \{q_{j_0}\} \cup W_{j_0}$. As in the case of finite acceptance condition, it can now be shown that $0 \leq \mathsf{val}(W \cap Q_1, x, u) < 1$. The proposition follows in this case as well.

($\Leftarrow$) Let u, κ', W be as in the statement of the proposition.

If $W \subseteq Q_1$ then as $\mathsf{val}(W, x, u) < 0$ it follows that $\delta_u(q_0, W) > x$. The result follows from observing that $\mu^{acc}_{\mathcal{A},\kappa} \geq \delta_u(q_0, W)$.

On the other hand, if $W \cap Q_0 \neq \emptyset$, then let q be the unique state in $W \cap Q_0$. Then, it is also the case that $\mathsf{post}(q_0, u) \cap Q_0 = \{q\}$. It is easy to see that

$$\mu^{acc}_{\mathcal{A},\kappa} \geq \delta_u(q_0, q) + \delta_u(q_0, W \cap Q_1)$$

in this case. Now as $\mathsf{val}(W \cap Q_1, x, u) < 1$, it follows that

$$x - \delta_u(q_0, (W \cap Q_1)) < \delta_u(q_0, q).$$

Hence,

$$\begin{array}{rcl} \mu^{acc}_{\mathcal{A},\kappa} & \geq & \delta(q_0, q) + \delta(q_0, W \cap Q_1) \\ & > & x. \end{array}$$

The result follows. □

Proposition 2.4 states that the words definitely accepted from any witness set is regular. Thus, the crux of the proof of the fact that integer HPAs recognize regular languages will be to show that there is a way to maintain the $\mathsf{val}(\cdot, x, \cdot)$ function for each witness set using only finite memory. In order to prove this, we first observe that integer HPA have the special property that transitions between states of level 0 have transition probability of the form $\frac{1}{m}$ where m is a natural number.

Proposition 4.4. *For an integer* HPA $\mathcal{A}$*, any pair of states* $q, q' \in Q_0$ *and* $a \in \Sigma$*, if* $\delta_a(q, q') \neq 0$ *then* $\delta_a(q, q') = \frac{1}{m}$ *for some* $m \in \mathbb{N}$*.*

Proof. Let $m = 1 + \sum_{q'' \in Q_1} \frac{\delta_a(q,q'')}{\delta_a(q,q')}$. Observe that $m \in \mathbb{N}$ by the properties of an integer HPA. Moreover $m \times \delta_a(q, q') = \sum_{q'' \in Q} \delta_a(q, q'') = 1$. The proposition follows. □

Remark: The proof of the Proposition 4.4 shows that the transition probabilities of an integer HPA is always going to be a rational number.

Proposition 4.5 below makes a very important observation — the set of relevant values that the function val can take are finite. Proposition 2.7 in Section 2.1 essentially says that when the function val takes on values either below 0 or above 1, either all extensions of the current input will have sufficient probability among witness sets in Q_1 or no extension will have sufficient probability. Thus, when measuring the quantity val what matters is only whether it is strictly less than 0, strictly greater than 1 or its exact value when it is in $[0, 1]$. Proposition 4.5 below guarantees that val takes values from a finite set when it lies within $[0, 1]$.

Proposition 4.5. *Let* $\mathcal{A}$ *be an integer* HPA *over alphabet* Σ *with level 0 and level 1 sets* Q_0 *and* Q_1*,* $C \subseteq Q_1$*, and* x *be a rational number* $\frac{c}{d}$*. For any* $u \in \Sigma^*$*, if* $\mathsf{val}(C, x, u) \in [0, 1]$ *then there is* $e \in \{0, 1, 2, \ldots d\}$ *such that* $\mathsf{val}(C, x, u) = \frac{e}{d}$*.*

Proof. The proposition is proved by induction on the length of u. Observe that for any $C \subseteq Q_1$, $\mathsf{val}(C, x, \epsilon) = x = \frac{c}{d}$, and so the proposition holds in the base case. Let the proposition hold for string u. Consider $a \in \Sigma$ and $C \subseteq Q_1$. We will show that the proposition holds for $\mathsf{val}(C, x, ua)$. Let $D = \{q \in Q_1 \mid \mathsf{post}(q, a) \in C\}$. Observe that if $\mathsf{val}(C, x, ua) \in [0, 1]$ then $\mathsf{val}(D, x, u) \in [0, 1]$. Hence, by the induction hypothesis, we know that there is e such that $\mathsf{val}(D, x, u) = \frac{e}{d}$. If $\delta_{ua}(q_0, Q_0) = 0$ and $\mathsf{val}(C, x, ua) \in [0, 1]$ then $\mathsf{val}(C, x, ua) = 0$ and the proposition follows. Let us consider the case when $\delta_{ua}(q_0, Q_0) \neq 0$. Let $\mathsf{post}(q_0, u) \cap Q_0 = \{q\}$ and $\mathsf{post}(q_0, ua) \cap Q_0 = \{q'\}$. We can make the following sequence of observations.

$$\begin{aligned}\mathsf{val}(C, x, ua) &= \tfrac{x - \delta_{ua}(q_0, C)}{\delta_{ua}(q_0, Q_0)} = \tfrac{x - (\delta_u(q_0, D) + \delta_u(q_0, q)\delta_a(q, C))}{\delta_u(q_0, q)\delta_a(q, q')} \\ &= \tfrac{\mathsf{val}(D, x, u)}{\delta_a(q, q')} - \tfrac{\delta_a(q, C)}{\delta_a(q, q')}\end{aligned}$$

Proposition 4.4 implies that $\frac{1}{\delta_a(q,q')}$ is a natural number. Further, since $\mathcal{A}$ is an integer HPA, $\frac{\delta_a(q,C)}{\delta_a(q,q')}$ is also a natural number. Putting it all together the proposition follows. □

Proposition 4.5 allows us to keep track of val using finite memory. This is captured in the following Lemma.

Lemma 4.6. *Consider an integer* HPA $\mathcal{A}$ *over alphabet* Σ *with* Q_0 *and* Q_1 *as level 0 and level 1 states. Let* $x = \frac{c}{d}$ *be a rational threshold. For an arbitrary* $C \subseteq Q_1$*,* $q \in Q_0$*, and* $e \in \{0, 1, \ldots d\}$*, the following six languages*

$$\begin{aligned}
\mathsf{L}_{(q,C,e)} &= \{u \in \Sigma^* \mid \mathsf{post}(q_0, u) \cap Q_0 = \{q\} \textit{ and } \mathsf{val}(C, x, u) \leq \tfrac{e}{d}\} \\
\mathsf{L}_{(q,C,-)} &= \{u \in \Sigma^* \mid \mathsf{post}(q_0, u) \cap Q_0 = \{q\} \textit{ and } \mathsf{val}(C, x, u) < 0\} \\
\mathsf{L}_{(q,C,+)} &= \{u \in \Sigma^* \mid \mathsf{post}(q_0, u) \cap Q_0 = \{q\} \textit{ and } \mathsf{val}(C, x, u) > 1\} \\
\mathsf{L}_{(*,C,e)} &= \{u \in \Sigma^* \mid \mathsf{post}(q_0, u) \cap Q_0 = \emptyset \textit{ and } \mathsf{val}(C, x, u) \leq \tfrac{e}{d}\} \\
\mathsf{L}_{(*,C,-)} &= \{u \in \Sigma^* \mid \mathsf{post}(q_0, u) \cap Q_0 = \emptyset \textit{ and } \mathsf{val}(C, x, u) < 0\} \\
\mathsf{L}_{(*,C,+)} &= \{u \in \Sigma^* \mid \mathsf{post}(q_0, u) \cap Q_0 = \emptyset \textit{ and } \mathsf{val}(C, x, u) > 1\}
\end{aligned}$$

are all regular.

Proof. The result is proved by constructing a single NFA that essentially recognizes each of the languages in the proposition. The NFA that we will construct will maintain the level 0 state that $\mathcal{A}$ could be in, guess the set C whose val we care to compute, and maintain a upper bound on val. Proposition 2.7 and 4.5 together ensure that there are only finitely many relevant values that val can take which can be maintained by a finite automaton.

We now give the precise construction of the NFA. Consider NFA $\mathcal{M} = (U, r_0, \delta', F)$ on the alphabet Σ, where the set of states is $U = (Q_0 \cup \{*\}) \times 2^{Q_1} \times (\{0, 1, 2, \ldots d\} \cup \{+, -\})$. Thus, a state is of the form (q, C, e), where $q \in Q_0 \cup \{*\}$, $C \subseteq Q_1$ and e is a natural number between 0 and d or $+$ or $-$. The invariant that the automaton will maintain is the following. For any finite word u, Σ^*,

$$(q, C, e) \in \delta'(r_0, u) \text{ iff} \quad \text{either } (\mathsf{post}(q_0, u) \cap Q_0 = \emptyset \text{ and } q = *) \text{ or } \mathsf{post}(q_0, u) \cap Q_0 = \{q\} \text{ and } \begin{cases} \mathsf{val}(C, x, u) \leq \frac{e}{d} & \text{if } e \notin \{+, -\} \\ \mathsf{val}(C, x, u) < 0 & \text{if } e = - \end{cases}.$$

To be consistent with the above invariant, the initial state $r_0 = (q_0, \emptyset, c)$. The transition function δ' will maintain this invariant and is defined as follows. $(q', C', e') \in \delta'((q, C, e), a)$ if (q', C', e') satisfies the following constraints.

- If $q = *$ or $Q_0 \cap \mathsf{post}(q, a) = \emptyset$ then $q' = *$. Otherwise, $q' \in Q_0 \cap \mathsf{post}(q, a)$.
- $\mathsf{post}(C, a) \subseteq C'$, and
- If $e \in \{+, -\}$ or $q = *$ then $e' = e$. When $q' = *$, $e' = +$ if $\delta_a(q, C') < \frac{e}{d}$, $e' = -$ if $\delta_a(q, C') > \frac{e}{d}$, and $e' = 0$ if $\delta(q, C') = \frac{e}{d}$. Otherwise, consider

$$f = \frac{e}{\delta_a(q, q')} - d \cdot \frac{\delta_a(q, D)}{\delta_a(q, q')}$$

  where $D = \{q \in Q_1 \mid \mathsf{post}(q, a) \in C\}$. Observe that f is an integer. Then, $e' = f$ if $0 \leq f \leq d$, $e' = +$ if $f > d$, and $e' = -$ if $f < 0$.

The transition function defined above maintains the required invariant.

The languages defined in the proposition can then be defined by making an appropriate choice of final states. To recognize the language $\mathsf{L}_{(q,C,e)}$ where $q \in Q_0 \cup \{*\}$, and $e \in \{0, 1, \ldots d\} \cup \{-\}$, we take $F = \{(q, C, e)\}$. Next, $\overline{\mathsf{L}_{(q,C,+)}}$, when $q \in Q_0 \cup \{*\}$ can be recognized by taking $F = \{(q, C, e) \mid e \neq +\}$. Hence $\mathsf{L}_{(q,C,+)}$ is also regular. This completes the proof of the proposition. □

We are ready to present the main result of this section.

Theorem 4.7. *For any integer* HPA $\mathcal{A}$*, rational threshold* $x \in [0, 1]$*, the languages* $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$ *and* $\mathsf{L}^{\mathsf{a}}{}_{\geq x}(\mathcal{A})$ *are regular (where* $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$*).*

Proof. From Proposition 4.3, we can conclude that

$$\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A}) = \left(\bigcup_{C \subseteq Q_1,\ q \in Q_0 \cup \{*\}} \mathsf{L}_{(q,C,-)} \mathsf{L}_C \right) \cup \left(\bigcup_{C \subseteq Q_1,\ q \in Q_0, e \in [0,1)} \mathsf{L}_{(q,C,e)} \mathsf{L}_{C \cup \{q\}} \right)$$

where L_C is the set of words definitely accepted from the witness set C. From Proposition 2.4 and Lemma 4.6, we can conclude that each of the languages on the right hand side is regular, and therefore, $\mathsf{L}^{\mathsf{a}}{}_{>x}(\mathcal{A})$ is regular.

From Proposition 2.4 and Lemma 4.6, we can conclude the regularity of the language

$$L_1 = \left(\bigcup_{C \subseteq Q_1,\ q \in Q_0 \cup \{*\},\ e \in [0,-} \mathsf{L}_{(q,C,e)} \mathsf{L}_C \right) \cup \left(\bigcup_{C \subseteq Q_1,\ q \in Q_0, e \in [0,1)} \mathsf{L}_{(q,C,e)} \mathsf{L}_{C \cup \{q\}} \right).$$

Now, it is easy to see that $\mathsf{L}^{\mathsf{f}}{}_{\geq x}(\mathcal{A}) = L_1$ and so that completes the proof of this proposition for the case of finite words.

For the case of Büchi acceptance (or Muller acceptance), $\mathsf{L}^{\mathsf{b}}{}_{\geq x}(\mathcal{A})$ contains strings that are accepted with probability x "in the limit", in addition to the strings in L_1. We will now define a language L_2 such that $\mathsf{L}^{\mathsf{a}}{}_{\geq x}(\mathcal{A}) = L_1 \cup L_2$ (when $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$)

and prove that L_2 is regular to complete the proof. To define the language L_2, we need some auxiliary definitions, which we present first. Recall that the level 0 and 1 states of $\mathcal{A}$ are Q_0 and Q_1. Let the threshold be $x = \frac{c}{d}$, and the input alphabet of $\mathcal{A}$ be Σ. For an input $\alpha \in \Sigma^\omega$, a *witnessing run* of $\mathcal{A}$ on α is an infinite sequence $\lambda = (q_0, C_0, e_0), (q_1, C_1, e_1), \ldots$ such that

- $q_0 = q_0$, $C_0 = \emptyset$, $e_0 = c$,
- For each i, $q_i \in Q_0$, $C_i \subseteq Q_1$, and $e_i \in \{0, 1, \ldots d\}$,
- For each i, $q_{i+1} \in \mathsf{post}(q_i, \alpha[i])$, and $\mathsf{post}(C_i, \alpha[i]) \cap Q_1 \subseteq C_{i+1}$,
- For each i, $\mathsf{val}(C_i, x, \alpha[0 : i-1]) \leq \frac{e_i}{d}$, and
- For infinitely many i, $\delta_{\alpha[i]}(q_i, q_{i+1}) < 1$.

In other words, a witnessing run is a run of the nondeterministic machine constructed in the proof of Lemma 4.6, with the added restriction that for infinitely many steps, the probability of transitions at level 0 is < 1. Given a witnessing run $\lambda = (q_0, C_0, e_0)$, $(q_1, C_1, e_1), \ldots$ on input α, the set of runs associated with λ, denoted $R(\lambda, \alpha)$, is given as

$$R(\lambda, \alpha) = \{\rho \in Q^\omega \mid \forall i.\ \rho[i] \in \{q_i\} \cup C_i,\ \delta_{\alpha[i]}(\rho[i], \rho[i+1]) > 0 \text{ and } \exists j.\ \rho[j] \in Q_1\}.$$

In other words, $R(\lambda, \alpha)$ is the set of all runs of $\mathcal{A}$ on α that "conform" to the witnessing run λ, with the added restriction that at some point the run reaches (and stays) at level 1. Notice, that since in a witnessing run λ infinitely many transitions between level 0 states is < 1, ensures that the run that stays at level 0 and conforms with λ has measure 0. Thus, the measure of runs in $R(\lambda, \alpha)$ is the same as the measure of *all* runs that conform to λ. Finally, we say that a witnessing run λ on input α is *accepting* iff every run $\rho \in R(\lambda, \alpha)$ is an accepting run of $\mathcal{A}$.

The language L_2 is defined as follows.

$$L_2 = \{\alpha \in \Sigma^\omega \mid \text{there is an accepting witnessing run } \lambda \text{ on } \alpha\}.$$

Claim 5. *For any* HPA $\mathcal{A}$ *with threshold* $x = \frac{c}{d}$, *and* $\mathsf{a} \in \{\mathsf{b}, \mathsf{m}\}$, $\mathsf{L}^{\mathsf{a}}_{\geq x}(\mathcal{A}) = L_1 \cup L_2$.

The crux of the proof is arguing that the language L_2 is regular. We do this by constructing a nondeterministic Büchi automata accepting L_2. The nondeterministic automaton described in the proof of Lemma 4.6 almost has all the ingredients to recognize the language L_2 — it keeps track of the states visited in level 0, and the val for some subset of states at level 1. However, it does not keep track of whether a final accepting state has been visited, and whether transitions between level 0 states have probability < 1. We will fix this problem carrying out something like a "marked subset" construction and using the ideas of the automaton in Lemma 4.6. Details are given below.

Claim 6. *The language* L_2 *is regular.*

Proof. We prove this observation for only the case when $\mathcal{A}$ is a PBA. We will sketch how the construction can be adapted for the Muller case at the end of this proof sketch.

Let $\mathcal{A} = (Q, q_0, \delta, Q_f)$, where Q_f is the set of final states, and we take the threshold $x = \frac{c}{d}$. Consider the following nondeterministic Büchi automaton $\mathcal{M} = (U, r_0, \delta', F)$. A state in U is a tuple of the form (q, C, m, e), where $q \in Q_0$, $C \subseteq Q_1$, $\mathsf{m} : C \cup \{q\} \to \{0, 1\}$, and $e \in \{0, 1, 2, \ldots d\}$. Here m is a marking on $C \cup \{q\}$ that keeps track of whether a final state has been visited and whether a transition between level 0 states has probability < 1. The initial state $r_0 = (q_0, \emptyset, \mathsf{m}_*, c)$, where $\mathsf{m}_*(q_0) = 1$. The set of final states F will be all states (q, C, m, e) such that for every q' in the domain of m, $\mathsf{m}(q') = 1$.

Before defining the transition function δ' it is useful to spell out the invariant maintained by the automaton, to explain how the construction works. Suppose $\rho = (q_0, C_0, \mathsf{m}_0, e_0), (q_1, C_1, \mathsf{m}_1, e_1), \ldots$ is an infinite run of $\mathcal{M}$ on input word α. The constraints this run will satisfy are the following.

- For each i, $q_i \in \mathsf{post}(q_0, \alpha[0 : i]) \cap Q_0$, and $\mathsf{val}(C_i, x, \alpha[0 : i]) \leq \frac{e_i}{d}$
- For each i, let j_i be the maximum position $< i$ such that for all q in the domain of m_{j_i}, $\mathsf{m}_{j_i}(q) = 1$. Note, such a j_i always exists because $r_0 = (q_0, C_0, \mathsf{m}_0, e_0)$ is such a state. For every state $q \in C_i$ such that $\mathsf{m}_i(q) = 1$, every run of $\mathcal{A}$ from some state $q' \in \{q_{j_i}\} \cup C_{j_i}$ to q on the word $\alpha[j_i : i]$ visits an accepting state in Q_f, and $\mathsf{m}_i(q_i) = 1$, if the (unique) run from q_{j_i} to q_i on $\alpha[j_i : i]$ has some transition with probability < 1.

Observe that if this invariant is maintained then $\mathcal{M}$ will indeed accept the language L_2.

The transition function δ' that maintains this invariant is defined as follows. $(q', C', \mathsf{m}', e') \in \delta'((q, C, \mathsf{m}, e), a)$ if $(q', C', \mathsf{m}', e')$ satisfies the following constraints.

- $q' \in \mathsf{post}(q, a) \cap Q_0$
- $\mathsf{post}(C, a) \subseteq C'$
- $e' = \frac{e}{\delta_a(q,q')} - d \cdot \frac{\delta_a(q,D)}{\delta_a(q,q')}$ and $e' \in \{0, 1, \ldots d\}$, where $D = \{q \in Q_1 \mid \mathsf{post}(q, a) \in C\}$.
- m' is given as follows. We will say the marking m is *full* if $\mathsf{m}(p) = 1$ for all $p \in C \cup \{q\}$. For q',

$$\mathsf{m}'(q') = \begin{cases} 1 & \text{if } \delta_a(q, q') < 1 \\ \mathsf{m}(q) & \text{if } \mathsf{m} \text{ is not full and } \delta_a(q, q') = 1 \\ 0 & \text{otherwise} \end{cases}$$

  For the states in C', m' is defined as follows. For every $p' \in C' \setminus \mathsf{post}(C, a)$, $\mathsf{m}'(p') = 0$. For every $p' \in C' \cap Q_f$, $\mathsf{m}'(p') = 1$. For $p' \in C' \setminus Q_f$, m' is defined as follows. If m is full, then $\mathsf{m}'(p') = 0$. On the other hand, if m is not full then $\mathsf{m}'(p') = 1$ if and only if for every $p_1 \in C \cup \{q\}$ such that $p' \in \mathsf{post}(p_1, a)$, $\mathsf{m}(p_1) = 1$.

The automaton $\mathcal{M}$ constructed above recognizes L_2. If $\mathcal{A}$ has a Muller acceptance condition then $\mathcal{M}$ will initially guess an accepting set of states F, and the marking function on level 1 states will turn 1 only if the sub-run visits each of the states in F. □

□

The following theorem shows that the problems of checking emptiness and universality are **PSPACE**-complete for integer HPA, thus giving a tight upper bound.

Theorem 4.8. *Given an integer* HPA $\mathcal{A}$, $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$, $\rhd \in \{>, \geq\}$, *the problem of determining if* $\mathsf{L}^{\mathsf{a}}_{\rhd x}(\mathcal{A}) = \emptyset$ *is* **PSPACE**-*complete. Similarly, the problem of checking universality is also* **PSPACE**-*complete.*

Proof. The proof of Theorem 4.7 shows that the languages accepted by an integer HPA are recognized by an an exponential sized (non-probabilistic) automaton. Thus the emptiness problem for integer HPA can be reduced to the emptiness problem for exponential sized automaton, giving us the **PSPACE** upper bound (we don't need to explicitly construct the automaton, just *run* the standard emptiness checking algorithm for (non-probabilistic) automaton). Next, recall that in the proof of Theorem 3.19, we showed that checking the universality of an HPA $\mathcal{A}$ can be reduced to checking the emptiness of another closely related HPA $\mathcal{B}$ simply by changing the accepting condition of $\mathcal{A}$. Thus, if $\mathcal{A}$ is an integer HPA then so is $\mathcal{B}$. Finally, since the emptiness problem of integer HPA can be decided in **PSPACE**, so can the universality problem.

The **PSPACE**-hardness follows from Theorems 3.16 and 3.19; the HPAs constructed in those proofs are (vacuously) integer HPA because they have only one state at level 0 and no transitions between level 0 states. □

5. Undecidability for 2-HPA

We now formally define general k-HPA in which the set of states is stratified into $k + 1$ levels.

Definition 5.1. For an integer $k > 0$, a *k-hierarchical probabilistic automaton* (HPA) is a probabilistic automaton $\mathcal{A} = (Q, q_0, \delta, \mathsf{Acc})$ over alphabet Σ such that Q can be partitioned into $k + 1$ sets $Q_0, Q_1, \ldots, Q_k$ satisfying the following properties:

- $q_0 \in Q_0$;
- for every i, $0 \leq i \leq k$ and every $q \in Q_i$ and $a \in \Sigma$, $|\mathsf{post}(q, a) \cap Q_i| \leq 1$; and,
- for every i, $0 < i \leq k$, $q \in Q_i$ and $a \in \Sigma$, $\mathsf{post}(q, a) \cap Q_j = \emptyset$ for every $j < i$.

For any k-HPA $\mathcal{A}$, as given above, for $0 \leq i \leq k$, we call elements of Q_i as *level i* states of $\mathcal{A}$.

In Section 3, we showed that even though 1-HPA with non-extremal cut-points can recognize non-regular languages, decision problems like emptiness and universality are decidable in **EXPTIME**. We will now demonstrate that 1-HPAs are the most "expressive" class for which these problems are decidable. More precisely, our main result in this section shows that the emptiness problem for 2-HPAs is undecidable.

Theorem 5.2. *Given a* 2-HPA $\mathcal{A}$, $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$, *and* $\rhd \in \{>, \geq\}$, *the problem of determining if* $\mathsf{L}^{\mathsf{a}}_{\rhd \frac{1}{2}}(\mathcal{A}) = \emptyset$ *is undecidable.*

Proof. We use the approach given in [10], with major modifications, where the non-emptiness problem for k-HPAs was shown to be undecidable for $k \geq 6$. The undecidability is established by reducing the halting problem of deterministic 2-counter machines to the non-emptiness problem of 2-HPAs.

Let T be a deterministic 2-counter machine with control states Q and a special halting state q_h. Without loss of generality, we make the following assumptions about T: the operation T does on the counters (including zero and non-zero tests) is uniquely determined by it's control state; each transition of T changes at most one counter as determined by the control state; and the initial counter values are 0. Recall that a configuration of such a machine is of the form (q, a^i, b^j), where $q \in Q$ is the current control state, and a^i (b^j) is the unary representation of the value stored in the first counter (second counter, respectively). We will refer to the first counter as the "a-counter" (since it is encoded using 'a' symbols) and the second counter as the "b-counter" (since it is encoded using 'b' symbols).

We will construct a 2-HPA $\mathcal{A}_T$ with the property that its language is non-empty iff T halts. The input alphabet Σ of $\mathcal{A}_T$ will have 6 symbols — ',', '(', ')', 'a', 'b', and 'τ'. $\mathcal{A}_T$ will have the following properties.

- The only words, ρ, accepted with *non-zero* probability are those that contain the symbol τ, i.e., ρ is of the form $\alpha\tau\beta$ where $\alpha \in (\Sigma \setminus \{\tau\})^*$ and $\beta \in \Sigma^* \cup \Sigma^\omega$.
- Consider $\rho = \alpha\tau\beta$, where $\alpha \in (\Sigma \setminus \{\tau\})^*$. If α is of the form $\alpha_1\alpha_2$, where α_1 is the (unique) halting computation of T then ρ will be accepted with probability $> \frac{1}{2}$. On the other hand, if α is either a *proper* prefix of a computation of T, or is invalid (i.e., either α is not of the right format or has an invalid transition), then ρ is accepted with probability $< \frac{1}{2}$.

If we successfully ensure the above properties then we would have, T halts iff $\mathsf{L}^{\mathsf{a}}_{\rhd \frac{1}{2}}(\mathcal{A}_T) \neq \emptyset$, for $\mathsf{a} \in \{\mathsf{f}, \mathsf{b}, \mathsf{m}\}$, and $\rhd \in \{>, \geq\}$.

To implement the above ideas, the HPA $\mathcal{A}_T$, on an input ρ, needs to check the following conditions.

(1) ρ is of the form $\alpha\tau\beta$, where $\alpha \in (\Sigma \setminus \{\tau\})^*$.

(2) α is of the right format, i.e., it is a sequence of tuples of the form (q, a^i, b^j).

(3) The first configuration of α is the initial configuration of T.

(4) The control state in successive configurations in α is in accordance with the transitions of T.

(5) Successive counter values in the sequence follow because of a valid transition of T.

(6) In the last configuration of α, T is in the halting state q_h.

All the above checks, *except (5)*, can be accomplished using a deterministic finite automaton (which is a 0-HPA). Thus, there is a deterministic finite automaton $\mathcal{B}_T$ with a special, absorbing accept state such that the input words ρ that cause $\mathcal{B}_T$ to reach this

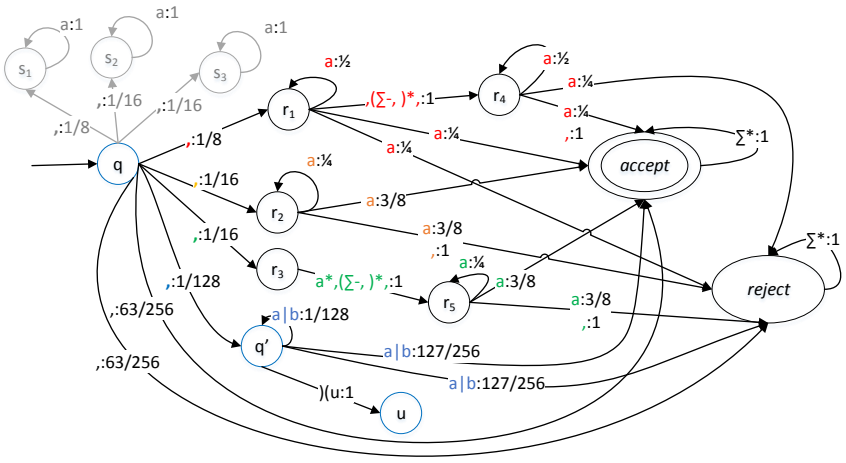

Figure 5: Transitions from state $q \neq q_h$ in C_T. Inputs and associated probabilities are shown on each transition separated by ":" symbol. Missing transitions are as follows: from every state (except accept) there is a τ transitions to reject with probability 1; transitions on an input symbol (not τ) not shown from a state (e.g., on a from q) are assumed to be self loops with probability 1. Transition structure from states s_1, s_2, s_3 is not shown and is "similar" to that from r_1, r_2, r_3. u is the control state of T after taking the transition enabled from control state q. Some of the transitions are labeled by regular expressions; these are implemented by having appropriate intermediate states and transitions. In the picture q, q', u are level 0 states, the states $r_1, \ldots, r_5$ and $s_1, \ldots$ are level 1 states, and accept and reject are level 2 states. The level 0 states are drawn as blue circles, level 1 states are drawn as black circles and level 2 states are drawn as ellipses.

unique accepting state are exactly those that satisfy all the above conditions, except possibly (5). Checking condition (5) will be accomplished by a 2-HPA C_T; this is the main challenge, and its description will take the rest of this section. Before presenting the details of C_T, we point out that the desired automaton $\mathcal{A}_T$ is one that synchronously executes $\mathcal{B}_T$ and C_T on the input, and accepts when both accept. Notice, that $\mathcal{A}_T$ will be a 2-HPA, since $\mathcal{B}_T$ is a deterministic finite automaton, and C_T is a 2-HPA.

We now describe the automaton C_T. We will not give a formal, tuple-based definition of automaton C_T because it will obfuscate the ideas behind the construction. The set of control states of C_T will include Q (the control states of T) and a unique, absorbing accepting state accept, and a unique, absorbing rejecting state reject. To outline the ideas behind the construction of C_T, we will only consider its behavior on inputs ρ that satisfy conditions (1), (2), (3), (4), and (6), i.e., ρ is a sequence of configurations $(q_0, a^{i_0}, b^{j_0})(q_1, a^{i_1}, b^{j_1}) \cdots$, where each control state q_{i+1} follows correctly from q_i according to the transition *determined* by q_i; this is because $\mathcal{B}_T$ ensures that any input that violates (1), (2), (3), (4) or (6) will be rejected with probability 1 by $\mathcal{A}_T$ (independent of C_T's behavior). On such a well formed input ρ, we will describe how C_T processes each configuration (q, a^j, b^j) in ρ as it is encountered. After reading '$(q$' of the configuration (q, a^i, b^j), the "main" thread of C's computation will be in state q. If q is the unique halting state q_h of T, C_T will transition to (accepting state) accept with probability 1 on reading ',', and stay there for the rest of the computation.

On the other hand, if q is not the halting state, C_T's behavior is shown in Figure 5; it is complicated and we will explain how it works below. To reduce clutter, not all transitions are shown in Figure 5. In particular, from any state, except accept, on symbol τ, C_T transitions to reject with probability 1. And, if from a state a transition on an input symbol other than τ is missing, then it is assumed that the automaton stays in the same state with probability 1. The level 0 states are drawn as blue circles, level 1 states are drawn as black circles and level 2 states are drawn as ellipses. We now explain the rationale behind the transitions. Intuitively, on reading ',', C_T will probabilistically choose to do one of the following.

Accept Accept the input without reading any further, by transitioning to state accept. This happens with probability $\frac{63}{256}$.

Reject Reject the input without reading any further, by transitioning to state reject. This also happens with probability $\frac{63}{256}$.

Skip "Ignore" the configuration (q, a^i, b^j) and potentially check the correctness of counter values in the future. This is depicted by the computation(s) going through state q'. In this computation, as it reads the counters a and b, there is a chance that C_T will abandon reading more of the input, and transition to accept or reject states. Let u be the control state of T after taking the transition enabled at q. If the input is neither accepted nor rejected while reading (q, a^i, b^j) in this mode, then C_T moves to state u. The computation proceeds again by making a choice between Accept/Reject/Skip/Check phases from u.

Check Check that the a-counter (or b-counter, choice made probabilistically) of the next configuration is consistent with the transition taken from (q, a^i, b^j), the current configuration being processed. Checking the value of the a-counters is accomplished by the paths going through the r_i states, while checking the b-counters is depicted by the computations through the s_i states, which is not shown completely in Figure 5. We will explain these computations in greater detail below.

Before explaining how the "Check" phase of C_T proceeds, it is important to make a couple of observations about the transitions shown in Figure 5. First, suppose C_T chooses one among the Accept, Reject or Skip options. Then the probability that C_T is in state accept after reading the configuration (q, a^i, b^j) is the same as the probability that it is in state reject. Thus, neither acceptance nor rejection probabilities are biased, *unless* C_T chooses the Check phase. Second, it is useful to clarify the structure of the states shown in Figure 5. The states q', r_i, and s_i (and intermediate states needed to implement transition labeled by regular expressions in Figure 5) are different for each state $q \in Q$. The states $q, u \in Q$ and q' are level 0 states; r_i and s_i (and other missing intermediate states) are level 1 states; and accept and reject are level 2 states. Thus, C_T is a 2-HPA.

Now let us explain how the "Check" phase proceeds. First the ideas behind checking the correctness of b-counter values are the same as those to check the a-counter values. Thus, we will only explain how a-counter values are checked (the transition structure involving the states r_i). Next, recall that, based on the state q, T might choose

either increment, decrement, or leave unchanged the a-counter. If the configuration immediately after (q, a^i, b^j) in input ρ is (u, a^k, b^ℓ), this requires checking $i + 1 = k$ (increment), $i = k + 1$ (decrement), or $i = k$ (unchanged). Hence, the challenge is primarily to check the *equality* of two numbers probabilistically; if we can check the "unchanged" case, then checking the increment or decrement cases only involves checking the unchanged case when 1 is added to either i or k. So in what follows we will only explain how C_T checks the unchanged case.

Let us fix the configurations (q, a^i, b^j) and (u, a^k, b^ℓ) and let us assume that the goal in the Check phase is to check that $i = k$. Checking if $i = k$ requires "counting" the number of as. But since C_T has only finite memory this cannot be carried out explicitly. Instead C_T will use coin tosses to *implicitly* count these numbers in terms of the probability of reaching various states; this general principle has been exploited previously [12, 10]. Intuitively, C_T probabilistically chooses to do one of the following.

Count i and k If C_T goes through state r_1 then it implicitly counts both i and k. In state r_1, on every a symbol read, it probabilistically chooses to either stay in r_1 (with probability $\frac{1}{2}$), or go to accept or reject (with probability $\frac{1}{4}$ each). After all the as are read, it skips all the symbols until the start of the as in the next configuration, and moves to state r_4 with probability 1. In state r_4, with each a read, it chooses to either stay in r_4, go to accept or go to reject with the same probabilities as from state r_1. Once all the as have been read, it goes to state accept with probability 1 when it reads ','.

Count i If C_T goes through state r_2 then it implicitly counts only i. With every a read, C_T stays in r_2 with probability $\frac{1}{4}$, moves to accept or reject with probability $\frac{3}{8}$ each. When all the as in the first configuration are read, C_T moves to reject with probability 1 when ',' is read, skipping the rest of the input.

Count k C_T implicitly counts k when it moves to state r_3. It skips the encoding of the a and b-counters in the first configuration, and the control state of the next configuration, and goes to state r_5 with probability 1, when the beginning of the encoding of the a-counter in the next configuration is reached. At this point, it counts k in r_5 in the same way as i was counted in r_2; with every a symbol read, it either stays in r_5 with probability $\frac{1}{4}$, or moves to accept or reject with probability $\frac{3}{8}$ each. After all the as have been read, it moves to reject with probability 1 when ',' is encountered.

To understand how i and k are implicitly being counted through these probabilistic transitions, we need to analyze the probabilities of acceptance and rejection. This analysis will also help establish the correctness of our construction.

Let p_{accept} and p_{reject} be the probabilities of reaching accept and reject states, respectively, when choosing each of the 3 options — Count i and k (state r_1), Count i (state r_2), and Count k (state r_3). These values are shown in Table 1. These values are derived as follows. Observe that C_T is going to be in the states r_1 and r_4, respectively, when receiving input a symbols of the first counter in the current and next configurations, after taking the r_1 branch from state q. The probabilities of taking the r_1 branch and then be in the states r_1, accept and reject, respectively, at the end of the first counter

Table 1: Probability Analysis for Undecidability Proof.

	P_{accept}	P_{reject}	sum
Count i and k (state r_1)	$\frac{1}{16}(1+2^{-(i+k)})$	$\frac{1}{16}(1-2^{-(i+k)})$	$\frac{1}{8}$
Count i (state r_2)	$\frac{1}{32}(1-2^{-2i})$	$\frac{1}{32}(1+2^{-2i})$	$\frac{1}{16}$
Count k (state r_3)	$\frac{1}{32}(1-2^{-2k})$	$\frac{1}{32}(1+2^{-2k})$	$\frac{1}{16}$

of the current configuration, are easily seen to be $\frac{1}{8}\cdot 2^{-i}$, $\frac{1}{16}(1-2^{-i})$ and $\frac{1}{16}(1-2^{-i})$. The probabilities of being in the states r_4, accept and reject, respectively, after passing through both r_1 and r_4 , at the end of the first counter of the next configuration, are easily seen to be $\frac{1}{8}\cdot 2^{-(i+k)}$, $\frac{1}{16}2^{-i}(1-2^{-k})$ and $\frac{1}{16}2^{-i}(1-2^{-k})$. Putting the above observations together and observing that the remaining probability of being in r_4 is transferred to the accept state after ',' is read the first counter of the next configuration, we get the probabilities shown in the first row of Table 1.

Observe that C_T is also in state r_2 with non-zero probability when the symbols of the first counter of the current configuration are received. The probabilities of taking the r_2 branch and then be in state r_2, accept and reject, respectively, at the end of the first counter of the current configuration are $\frac{1}{16}4^{-i}$, $\frac{1}{32}(1-4^{-i})$ and $\frac{1}{32}(1-4^{-i})$. Using these probability values and observing that the probability of being in r_2 is transferred to the reject state immediately after ',' is read after the first counter of the current configuration, we get the probability values shown in the second row of Table 1. By symmetric argument and by observing that, when in state r_3, C_T is receiving the symbols of the first counter of the next configuration, we get the probability values given in the third row of Table 1.

From this, we see that

$$p_{\mathsf{reject}} - p_{\mathsf{accept}} = \frac{1}{16}(2^{-2i} + 2^{-2k} - 2\cdot 2^{-(i+k)}) = \frac{1}{16}(2^{-i} - 2^{-k})^2 \geq 0.$$

If $i = k$, then $p_{\mathsf{reject}} - p_{\mathsf{accept}} = 0$. Now consider the case when $i \neq k$. Assume without loss of generality that $i > k$. In this case,

$$p_{\mathsf{reject}} - p_{\mathsf{accept}} \geq \frac{1}{16}(2^{-k} - 2^{-k-1})^2 = \frac{1}{16}2^{-2(k+1)} = \frac{1}{64}\cdot 2^{-2k}.$$

Symmetrically, when $k > i$, we have $p_{\mathsf{reject}} - p_{\mathsf{accept}} \geq \frac{1}{64}\cdot 2^{-2i}$. As observed before, the probabilities of going to the accept and reject states from q, without going through r_1, r_2, r_3 or s_1, s_2, s_3, are equal. Now the probability that C_T is at a level 0 state (i.e., such as q, q') after the current configuration is $\frac{1}{128}\cdot(\frac{1}{128})^{(i+j)}$ where i, j are the values of the first and second counters in the current configuration, which is $= \frac{1}{128}\cdot 2^{-(7\cdot i+7j)}$. This value is less than $\frac{1}{64}\cdot 2^{-2k}$ when $i > k$, and is less than $\frac{1}{64}\cdot 2^{-2i}$ when $k > i$. (The same reasoning is used to establish this property when we consider the second counter values using the values j, ℓ). This shows that $p_{\mathsf{reject}} - p_{\mathsf{accept}}$ is greater the probability that C_T is at a level 0 state plus the probabilities that C_T is in any of the intermediate level 1 states which go to the accepting state if the following configuration has a halt state in an illegal computation. From this, we see that even if we get an illegal computation that

ends with a halting state, that computation will be accepted with probability $< \frac{1}{2}$. On the other hand a legal halting computation will be accepted with probability $> \frac{1}{2}$. □

6. Related work

Probabilistic automata on finite words (PFAs) were first introduced by Rabin in [1] who showed that PFAs recognize non-regular languages even with finite memory. Rabin also showed that the language recognized by a PFA $\mathcal{A}$ is regular if the threshold x is *isolated* for $\mathcal{A}$. A threshold x is isolated if the acceptance probabilities of all words are bounded away from the x. Surprisingly, it was shown in [2, 12] that the emptiness and universality problems are undecidable for PFAs provided the threshold is not 0 or 1. When the threshold is 0 or 1, PFAs recognize regular languages which can be effectively constructed.

Probabilistic automata on infinite words were introduced by Baier and Größer in [3, 11]. The undecidability result on PFAs implies that problems of checking emptiness and universality for such automata is undecidable for any threshold in $(0, 1)$. Surprisingly, it is shown in [3, 11] that unlike the finite case, the language accepted with threshold 0 (and for any acceptance conditions) may not be ω-regular and that the emptiness problem of such automata is undecidable even for threshold 0. The problem of checking emptiness and universality for Probabilistic Büchi Automata when the threshold is 1 is shown to be decidable in **PSPACE** in [11, 22, 23]. The emptiness and universality problems are undecidable for Probabilistic Muller Automata for any threshold [3, 11]. We studied the expressiveness and the level of undecidability for probabilistic automata over infinite words when all the states of the automata are accepting states except for one unique reject state [17] and for general probabilistic automata in [22]. When the threshold is 0 and 1, we also showed that the language recognized by k-HPAs over infinite words is regular in [22]. Furthermore, it is shown that the emptiness/universality problems of these regular languages can be decided. A semantic class of automata, called *stochastically simple automata*, which include k-HPAs is identified in [24]. The emptiness and universality problems for such automata is shown to be decidable when the threshold is 0 and 1, for any parity acceptance condition.

As far as we know, 1-HPAs considered in this paper are the only class of probabilistic automata that can recognize non-regular languages and whose emptiness and universality is decidable. A semantic class of automata on infinite words, called uniform PBAs is identified in [3, 11] for which the emptiness and universality problems are decidable. Uniform PBAs are incomparable to HPAs and recognize only ω-regular language. Recently, it is shown in [25] that the problems of checking emptiness and universality of PFAs of bounded ambiguity is decidable. A PFA has bounded ambiguity if there is a number b such that the number of accepting runs on any word is bounded by b. However, such automata only recognize regular languages [25].

The value of an automaton (over both finite and infinite words) is said to be the supremum over acceptance probabilities over all words. The value problem asks given an automaton $\mathcal{A}$ and a rational number x if the value of the automaton $\mathcal{A}$ is x. Observe that the value problem for $x = 1$ is exactly the problem of checking whether 1 is not an

isolated cutpoint. The value problem is undecidable for PFAs [26], even for extremal thresholds [9]; which implies that it is undecidable for automata on infinite words. This problem is shown to be $\mathbf{\Pi_2^0}$-complete for PFAs [27]. The problem of checking if the value is 1 was shown to be **PSPACE**-complete for leak-tight automata [7, 8] which is sub-class of probabilistic automata that includes k-HPAs considered here. We have subsequently shown in [28] that the value problem of 1-HPAs is decidable and the value can be computed in **EXPTIME**. We showed that the value problem is **co-R.E.**-complete for 2-HPAs (and higher) in [28].

The problem of checking whether x is a isolated was shown to be undecidable in [26] and known to be $\mathbf{\Sigma_2^0}$-complete [27]. The result continues to hold even when x is 0 or 1 [9, 27]. As mentioned above, [8] that the problem of checking whether 1 or 0 is an isolated cutpoint for leaktight automata (including HPAs) is decidable in **PSPACE**. The problem of checking isolation was shown to be **R.E.**-complete for 2-HPAs (and higher) [28]. The isolation problem for 1-HPAs remains an open problem.

When the cut-point is isolated, the decidability of the emptiness problem for general probabilistic automata is not known. However, for eventually weakly ergodic PFAs [27] and k-HPAs (over both finite and infinite words), the emptiness problem is decidable when the cut-point is isolated [28].

The problem of checking emptiness and universality of PFAs when the input alphabet is unary, ie, consists of a single letter is a longstanding open problem. Recently it has been observed in [29, 30] that the emptiness problem on unary automata is equivalent to another longstanding open problem, the Skolem problem. We had shown in [20] that the emptiness problem is decidable under the assumption that the automata with unary alphabet is limit-isolated. We also established in [20] that the isolation problem is decidable in $\mathsf{coNP}^{\mathsf{RP}}$ for automata over unary alphabet.

7. Conclusions

We investigated the expressiveness of 1-HPA with non-extremal thresholds and showed, in spite of their very simple transition structure, they can recognize non-regular languages. Nevertheless, the canonical decision problems of emptiness and universality for 1-HPA turn out to be decidable in **EXPTIME** and are **PSPACE**-hard, when the transition probabilities are rational numbers. Bridging the gap between the upper and lower bounds for the above decision problems for 1-HPAs will be an interesting problem to explore as part of future work.

We also showed that integer 1-HPAs recognize regular languages and that the emptiness and universality problems are undecidable for k-HPA when $k \geq 2$. The expressiveness problem for integer k-HPAs, for $k \geq 2$, (i.e., whether they can express non-regular languages), and decidability of the emptiness/universality problems for these classes of automata need further investigation as part of future research.

8. Acknowledgments

Rohit Chadha was partially supported by NSF CNS 1314338 and CNS 1553548. Mahesh Viswanathan was partially supported by NSF CNS 1314485. A. Prasad Sistla

and Yue Ben were partially supported by CNS 1035914, CCF 1319754 and CNS 1314485.

[1] M. O. Rabin, Probabilistic automata, Inf. and Control 6 (3) (1963) 230–245.

[2] A. Paz, Introduction to Probabilistic Automata, Academic Press, 1971.

[3] C. Baier, M. Größer, Recognizing ω-regular languages with probabilistic automata, in: 20th IEEE Symp. on Logic in Computer Science, 2005, pp. 137–146.

[4] M. Größer, Reduction methods for probabilistic model checking, Ph.D. thesis, TU Dresden (2008).

[5] K. Chatterjee, T. A. Henzinger, Probabilistic automata on infinite words: Decidability and undecidability results, in: Intl. Symp. on Automated Technology for Verification and Analysis, 2010, pp. 1–16.

[6] R. Chadha, A. P. Sistla, M. Viswanathan, Power of randomization in automata on infinite strings, Logical Methods in Computer Science 7 (3) (2011) 1–22.

[7] N. Fijalkow, H. Gimbert, Y. Oualhadj, Deciding the value 1 problem for probabilistic leaktight automata, in: IEEE Symp. on Logic in Computer Science, 2012, pp. 295–304.

[8] N. Fijalkow, H. Gimbert, E. Kelmendi, Y. Oualhadj, Deciding the value 1 problem for probabilistic leaktight automata, Logical Methods in Computer Science 11 (2).

[9] H. Gimbert, Y. Oualhadj, Probabilistic automata on finite words: Decidable and undecidable problems, in: Intl. Colloquium on Automata, Languages and Programming, 2010, pp. 527–538.

[10] R. Chadha, A. P. Sistla, M. Viswanathan, Probabilistic Büchi automata with non-extremal acceptance thresholds, in: Intl. Conference on Verification, Model checking and Abstract Interpretation, 2010, pp. 103–117.

[11] C. Baier, M. Größer, N. Bertrand, Probabilistic ω-automata, Journal of the ACM 59 (1) (2012) 1–52.

[12] A. Condon, R. J. Lipton, On the complexity of space bounded interactive proofs (extended abstract), in: Symp. on Foundations of Computer Science, 1989, pp. 462–467.

[13] R. Chadha, A. P. Sistla, M. Viswanathan, Y. Ben, Decidable and expressive classes of probabilistic automata, in: 16th International Conference on Foundations of Software Science and Computation Structures, Vol. 9034 of Lecture Notes in Computer Science, Springer, 2015, pp. 200–214.

[14] Y. Ben, A. P. Sistla, Model checking failure-prone open systems using probabilistic automata, in: 13th International Symposium on Automated Technology for Verification and Analysis, Vol. 9364 of Lecture Notes in Computer Science, Springer, 2015, pp. 148–165.

[15] M. Vardi, Automatic verification of probabilistic concurrent finite-state programs, in: Symp. on Foundations of Computer Science, 1985, pp. 327–338.

[16] J. Kemeny, J. Snell, Denumerable Markov Chains, Springer-Verlag, 1976.

[17] R. Chadha, A. P. Sistla, M. Viswanathan, On the expressiveness and complexity of randomization in finite state monitors, Journal of the ACM 56 (5).

[18] D. Kozen, Lower bounds for natural proof systems, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 254–266.

[19] K. W. Wagner, Some observations on the connection between counting and recursion, Theoretical Computer Science 47 (3) (1986) 131–147.

[20] R. Chadha, D. Kini, M. Viswanathan, Decidable problems for unary pfas, in: 11th International Conference on Quantitative Evaluation of Systems, Vol. 8657 of Lecture Notes in Computer Science, Springer, 2014, pp. 329–344.

[21] C. Courcoubetis, M. Yannakakis, The complexity of probabilistic verification, J. ACM 42 (4) (1995) 857–907. doi:10.1145/210332.210339.
URL http://doi.acm.org/10.1145/210332.210339

[22] R. Chadha, A. Sistla, M. Viswanathan, Power of randomization in automata on infinite strings, Logical Methods in Computer Science 7 (3).

[23] C. Baier, N. Bertrand, M. Größer, On decision problems for probabilistic Büchi automata, in: FoSSaCS, 2008, pp. 287–301.

[24] K. Chatterjee, M. Tracol, Decidable problems for probabilistic automata on infinite words, in: 2012 27th Annual IEEE Symposium on Logic in Computer Science, 2012, pp. 185–194.

[25] N. Fijalkow, C. Riveros, J. Worrell, Probabilistic automata of bounded ambiguity, in: 28th International Conference on Concurrency Theory, CONCUR 2017, Vol. 85 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 19:1–19:14.

[26] A. Bertoni, The solution of problems relative to probabilistic automata in the frame of the formal languages theory, in: GI Jahrestagung, 1974, pp. 107–112.

[27] R. Chadha, A. Sistla, M. Viswanathan, Probabilistic automata with isolated cutpoints, in: Proceedings of the International Symposium on Mathematical Foundation of Computer Science, 2013, pp. 254–265.

[28] R. Chadha, A. P. Sistla, M. Viswanathan, Emptiness under isolation and the value problem for hierarchical probabilistic automata, in: Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Vol. 10203 of Lecture Notes in Computer Science, 2017, pp. 231–247.

[29] M. Biscaia, D. Henriques, P. Mateus, Decidability of approximate skolem problem and applications to logical verification of dynamical properties of markov chains, ACM Trans. Comput. Log. 16 (1) (2014) 4:1–4:17.

[30] S. Akshay, T. Antonopoulos, J. Ouaknine, J. Worrell, Reachability problems for Markov chains, Information Processing Letters 115 (2) (2015) 155–158.