Thesis Report

Submitted at the end of the Spring 2026 Semester

Student Name: Smayan Agarwal

Student ID: 1020221117

Supervisor: Prof. Aalok Thakkar

Submission Date: 19 April 2026

Ashoka University

Certificate by the Supervisor

This is to certify that Smayan Agarwal's Thesis, has been prepared under my supervision and is approved for submission and defense.

Signature of Supervisor
Prof. Aalok Thakkar
Assistant Professor, Department of Computer Science
Ashoka University

Contents

1 Introduction **3**

2 Literature Survey **4**

3 Work Done **6**

3.1 Preliminaries: Notation 6

3.2 Decidability of Distribution Property 6

3.3 Normalisation and Local Stochasticity 8

3.3.1 Normalising a Strongly Connected Component 9

3.3.2 Normalising the Acyclic Structure 10

3.4 Stochastic Regular Expressions 11

3.4.1 Equivalence with Weighted Automata 12

3.5 Identity Testing for Stochastic Languages 14

3.5.1 Identity Testing via Truncation 14

3.5.2 Towards Instance-Optimal Testing 15

3.5.3 Computing the Maximum-Probability Word 16

4 Extensions and Future Work **16**

5 Conclusion **17**

6 Acknowledgment **18**

1 Introduction

Weighted automata are a class of well-studied models in theoretical computer science. The easiest way to think of a weighted automaton is as a scoring machine. Feed it a word (a finite sequence of symbols from some alphabet, say $\{a, b\}$) and it returns a number. Internally, the machine reads the word one symbol at a time, moving between a finite set of states and accumulating a numerical score as it goes, with each transition contributing a multiplicative factor to the running total. The final score depends on the entire path the word traces through the machine. In this sense a weighted automaton is a generalisation of the classical finite automaton: instead of simply accepting or rejecting a word, it gives it a score. This shift from Boolean to quantitative output is what makes weighted automata so broadly applicable. They arise naturally across a wide range of domains: in natural language processing as models of syntactic structure, in bioinformatics as probabilistic models of DNA and protein sequences, in formal verification as specifications of quantitative system properties, and in information theory as models for optimal compression. Their appeal comes not only from their expressive power but from the remarkable range of properties that can be determined efficiently. Equivalence of two weighted automata, computation of total mass, minimisation to a canonical form, and determination of spectral properties are all tractable. This is a situation that contrasts sharply with more expressive models such as cost register automata, where even basic semantic questions become undecidable.

When the scores assigned by a weighted automaton to all words sum to a finite positive value, the automaton implicitly defines a probability distribution over strings. This connects weighted automata to a broad class of contemporary models: labelled Markov chains, hidden Markov models, probabilistic grammars, and large language models are all, in one form or another, machines that assign non-negative weights to sequences of symbols. Weighted automata are a natural formal model for studying this class of objects. In this thesis we study the class of weighted automata that define probability distributions, and develop a new characterisation of it. Our characterisation is more compositional and syntactically transparent than the automaton representation alone. We then show that this characterisation is not merely of theoretical interest, but has concrete consequences for the problem of identity testing for distributions over strings.

We introduce *stochastic regular expressions* (SREs), a grammar for probability distributions over strings built from four operations — Dirac distributions, convex combination, Cauchy product, and a discounted Kleene star. Each of these operations preserves the property of being a valid probability distribution. We prove that this grammar characterises exactly the class of distributions definable by weighted automata. The SRE representation turns out to have direct computational consequences

for the problem of *identity testing*. Given sample access to an unknown distribution P over strings and a known reference distribution Q, identity testing asks whether P and Q are close in total variation distance. For distributions over finite domains this problem is well understood, with optimal algorithms known since the work of Valiant and Valiant **valiant2017**. For distributions over the infinite domain Σ^*, however, the problem is harder: there is no bounded domain to test over, and one must first identify a finite region of Σ^* that captures enough of the mass of Q to make truncation valid.

For a detailed account of any individual component we refer the reader to the associated papers **icalp2026**, **fossacs2026**, **datamod2025**, which contain full proofs, worked examples, and extended discussion. What follows is an integrated account of the contributions and the conceptual thread connecting them.

2 Literature Survey

Stochastic languages define a probability distribution over words. A fragment of weighted automata and probabilistic grammar can be thought of as defining these probability distributions **droste2007weighted**. While a lot of work on stochasticity has been done in this field, our work differs in its definitions and aims.

In the literature there are broadly two different definitions of probabilistic automata. The definition given by Paz **paz1971probabilistic** takes that for each state the sum of all outgoing transitions must sum upto 1. This creates a probability distribution over strings. The alternate defintion given by Rabin **rabin1963probabilistic** says that for each state the sum of outgoing transitions for each letter must sum upto 1. This means that their probability distributions are over transitions from states and, by extension, over runs rather than over words. Our work shall focus on the definition given by Paz. In fact, Rabin probabilistic automata are a fragment of Paz probabilistic automata and every Rabin automata can be converted into a Paz automata by adding a dead state and transitions to that from each state. Work on Rabin automata has been conductd by Bollig et al who have established a Kleene style equivalence between probabilistic automata and probabilistic regular expressions **Bollig2012ProbabilisticKleene**. Chatterjee, Doyen, and Henzinger work on probabilistic buchi automata where words are recognised by cutoff conditions. However, their focus is on the transitions themselves being probabilistic rather than creating probability distributions over strings **chatterjee2009probabilistic**.

Work by Denis and Esposito formalized rational stochastic languages through multiplicity automata. They demonstrated that this class is closed under convex combinations and summations and designed the DEES algorithm to approximate target distributions using concise machines. However, their primary focus was on generalizing the underlying semirings. Consequently, they did not address the verification prob-

lem of determining if an arbitrary weighted automaton defines a valid distribution **Denis2009LearningWA**, **Denis2004**.

To fully unify these frameworks, one must also consider the descriptive and algorithmic complexity of these models. Droste and Gastin **droste2007weighted** extended Büchi's fundamental theorem to the quantitative setting, proving that formal power series recognizable by Weighted Automata are exactly those definable in Weighted Monadic Second-Order Logic (WMSO). However, the verification of such quantitative properties is computationally fraught. Gimbert and Oualhadj **gimbert2010probabilistic** proved that checking if a Probabilistic Automaton accepts a word with probability 1 (the Value 1 Problem) is undecidable. More recently, Chistikov et al. **chistikov2020big** extended this hardness to relative comparisons, proving that the problem of determining if the weights of one automaton are bounded by a constant multiple of another is undecidable for general Weighted Automata and Labelled Markov Chains. In doing so, they identify deep connections to Spectral Theory to find correspondences between automata and matrices, results which we intend to leverage. Beyond this, we rely on work by Rajeev Alur's group **Alur2013** on Cost Register Automata to delineate the boundaries of decidability, and on classical results in weighted automata as illustrated by Bollig and Zeitoun **bollig2015weighted** in their lecture notes.

The statistical side of our work connects to a long line of research on distribution testing. The problem of determining whether an unknown distribution matches a known reference has been studied since Pearson's chi-squared test **pearson1900**, which remains one of the most widely used tools in statistics. Pearson's test compares observed frequencies against expected frequencies under the reference distribution, and while it is effective for distributions over small domains, its sample complexity scales poorly with domain size. The modern theoretical treatment of identity testing was initiated by Batu et al. **batu2000**, who gave the first sublinear algorithms for testing closeness of distributions, and the problem has since attracted significant attention in the property testing literature **canonne2020survey**. A landmark result of Chan et al. **chan2014** established that identity testing over a domain of size n requires $\Theta(\sqrt{n}/\varepsilon^2)$ samples, and this bound is tight. Valiant and Valiant **valiant2017** sharpened this picture considerably by introducing the notion of *instance-optimal* testing: rather than bounding sample complexity purely in terms of domain size, their framework gives a bound that depends on the specific structure of the reference distribution Q via the quantity $p_{\max} = \max_x Q(x)$. This instance-optimal perspective is particularly relevant to our setting, where Q is a rational stochastic language whose tail structure is encoded in the SRE syntax. Understanding how $p_{\max}$ interacts with the truncation length θ is a key question we take up in our new work.

3 Work Done

3.1 Preliminaries: Notation

A weighted automaton is a directed graph whose edges carry numerical weights, and whose purpose is to assign a score to each word formed from letters in some finite alphabet Σ. To fix intuition, take $\Sigma = \{a, b\}$, so that words are finite strings such as *ab*, *abab*, and *bba*. A weighted automaton over Σ assigns each such word a non-negative real score: the word *abab* might receive a score of 4.32, while *ba* receives 0.71. This score is computed by tracing the word through the graph, one edge per letter, and multiplying together the weights encountered along the path, then summing over all paths that spell out the word.

Definition 1 (Weighted Automaton)**.** *A* weighted automaton *(WA) over semiring* $\mathbb{R}_{\geq 0}$ *is a tuple* $\mathcal{A} = (Q, \Sigma, \lambda, \{M_\sigma\}_{\sigma\in\Sigma}, \mu)$ *where:*

- Q *is a finite set of states,*
- $\lambda : Q \to \mathbb{R}_{\geq 0}$ *is the initial weight vector,*
- $\mu : Q \to \mathbb{R}_{\geq 0}$ *is the final weight vector,*
- $M_\sigma \in \mathbb{R}_{\geq 0}^{Q\times Q}$ *is the transition weight matrix for each symbol* $\sigma \in \Sigma$*.*

The score assigned to a word $w = w_1 w_2 \cdots w_n \in \Sigma^*$ *is*

$$[\![\mathcal{A}]\!](w) \;=\; \lambda^\top M_{w_1} M_{w_2} \cdots M_{w_n}\, \mu.$$

The function $[\![\mathcal{A}]\!] : \Sigma^* \to \mathbb{R}_{\geq 0}$ is called the *behaviour* of $\mathcal{A}$. The class of functions realisable in this way are called *rational series* over $\mathbb{R}_{\geq 0}$.

Now suppose the scores assigned to all words sum to exactly 1, that is,

$$\sum_{w\in\Sigma^*} [\![\mathcal{A}]\!](w) \;=\; 1.$$

Then $\mathcal{A}$ also defines a *probability distribution* over Σ^*. Each word receives a probability, all probabilities are non-negative, and they sum to 1. We call such a function a *stochastic language*, and the class of stochastic languages realisable by weighted automata over $\mathbb{R}_{\geq 0}$ the class of *rational stochastic languages*, denoted $\mathcal{S}^{\mathrm{rat}}_{\mathbb{R}_{\geq 0}}(\Sigma^*)$.

3.2 Decidability of Distribution Property

Given a weighted automaton $\mathcal{A}$, one might hope that whether it defines a probability distribution is somehow readable off its local structure: from the weights on individual

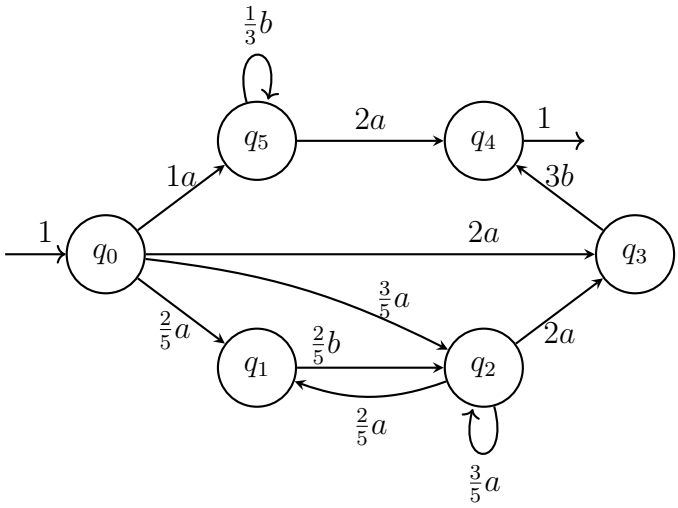

Figure 1: A weighted automaton A with alphabet $\Sigma = \{a, b\}$ and total mass $\|f_A\|_1 = 28$. The automaton is *globally* stochastic after normalisation by $1/28$, but not *locally* stochastic: state q_0 has outgoing weight $\frac{2}{5} + \frac{3}{5} + 2 + 1 > 1$.

edges or states. This hope turns out to be false. Whether $[\![\mathcal{A}]\!]$ sums to a finite value, let alone to 1, is a *global* property of the machine. No inspection of individual transitions can settle the question. We call this the gap between *local* and *global* stochasticity. We showed that the right tool to determine whether a machine defines a probability distribution turns out to be the *spectral radius* of the joint transition matrix.

Definition 2 (Spectral Radius)**.** *Let $M \in \mathbb{R}_{\geq 0}^{n \times n}$ be a non-negative square matrix with eigenvalues $\xi_1, \xi_2, \ldots, \xi_n \in \mathbb{C}$. The* spectral radius *of M is*

$$\rho(M) \;=\; \max_{1 \leq i \leq n} |\xi_i|.$$

Intuitively, $\rho(M)$ captures the long-run growth rate of the matrix powers M^n: if $\rho(M) < 1$ then $M^n \to 0$ as $n \to \infty$, while if $\rho(M) \geq 1$ the powers grow without bound. For a weighted automaton, define the *joint transition matrix* $M := \sum_{\sigma \in \Sigma} M_\sigma$, which aggregates all transitions regardless of the letter that triggers them. The total score of the automaton over all words of length n is then $\lambda^\top M^n \mu$, so the total score over all words is

$$\sum_{w \in \Sigma^*} [\![\mathcal{A}]\!](w) \;=\; \sum_{n=0}^{\infty} \lambda^\top M^n \mu \;=\; \lambda^\top \left(\sum_{n=0}^{\infty} M^n \right) \mu.$$

By the Neumann series theorem, the matrix series $\sum_{n=0}^{\infty} M^n$ converges if and only if $\rho(M) < 1$, in which case it converges to $(I - M)^{-1}$. This gives us the following decidability result.

Theorem 1. *Let $\mathcal{A} = (Q, \Sigma, \lambda, \{M_\sigma\}, \mu)$ be a weighted automaton over $\mathbb{R}_{\geq 0}$ with*

joint transition matrix $M = \sum_{\sigma \in \Sigma} M_\sigma$. *Then* $[\![\mathcal{A}]\!]$ *has finite total mass if and only if* $\rho(M) < 1$. *In this case,*

$$\sum_{w \in \Sigma^*} [\![\mathcal{A}]\!](w) \;=\; \lambda^\top (I - M)^{-1} \mu,$$

and this quantity is computable in polynomial time.[1]

3.3 Normalisation and Local Stochasticity

Knowing that a weighted automaton defines a probability distribution is useful, but it leaves the probabilistic structure global and implicit. Our aim is to go further: to rewrite any such automaton into an equivalent one where the probability structure is *local*. We want that for each state q the sum of all transitions and finalisation weights equals 1. Formally, for all q.

$$\mu(q) + \sum_{\sigma \in \Sigma} \sum_{q' \in Q} M_\sigma(q, q') \;=\; 1.$$

The normalisation procedure operates on the structure of the joint transition graph of the automaton. The states in a *strongly connected components* (SCCs) of this graph are its maximal subsets of states that mutually reach one another. Every state in an SCC can reach every other state in the same SCC via some sequence of transitions.

Intuitively, SCCs are the *cyclic cores* of the automaton. The acyclic transitions simply scale up the mass assigned to each branch. The SCCs however are special. They assign non zero weights to an infinite numbers of words in Σ^*. They can add mass in an unbounded manner. When the weight of a self loop is more than 1 then the total mass added to the system is infinite. When the self loop mass is less than 1, however, the mass added is finite. However, in general it is impossible to talk about the self loop on a state as we might have a complex interconnected system. The spectral radius being less than 1 is analogous to the self loop weight being less than 1. This decomposition is important because the two parts of the automaton require different normalisation strategies.

By the Perron-Frobenius theorem for reducible matrices, the joint transition matrix

[1] The situation is markedly different for the more expressive class of cost register automata (CRAs), where determining whether the total mass converges is undecidable in general, by reduction from the Post Correspondence Problem. Weighted automata, corresponding to CRAs with linear register updates, sit at precisely the boundary where decidability is recovered.

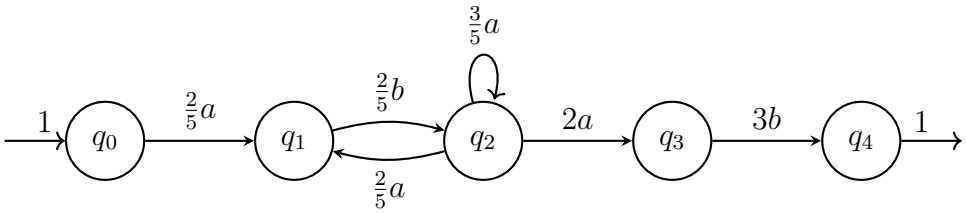

Figure 2: The weighted automaton from Figure 1 can be written down as acyclic components according to its topological sorting. The machine in this figure corresponds to the partial order $\{q_0\} \prec \{q_1, q_2\} \prec \{q_3\} \prec \{q_4\}$.

M can be arranged into block upper-triangular form

$$M \;=\; \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1k} \\ 0 & M_{22} & \cdots & M_{2k} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & M_{kk} \end{pmatrix}$$

where each diagonal block M_{ii} is the restriction of M to the i-th SCC, and the off-diagonal blocks M_{ij} for $i < j$ encode transitions between SCCs. Crucially, since $\rho(M) < 1$, each diagonal block satisfies $\rho(M_{ii}) \leq \rho(M) < 1$ as well, so every SCC individually has finite mass and can be normalised independently.

3.3.1 Normalising a Strongly Connected Component

Fix a single SCC with transition matrix M_i, initial weights λ_i, and final weights μ_i. The key quantity is the *expected future mass vector* $d \in \mathbb{R}_{\geq 0}^{|Q_i|}$, defined as

$$d \;=\; (I - M_i)^{-1}\mu_i.$$

The entry $d(q)$ represents the total probability mass that will eventually be accepted by the automaton if it starts in state q within this SCC. Since $\rho(M_i) < 1$, the matrix $(I - M_i)$ is invertible and d is well-defined and non-negative.

We use d to construct a diagonal scaling matrix $D = \mathrm{diag}(d)$ and define a normalised version of the SCC:

$$\lambda_i' = \lambda_i D, \qquad M_\sigma' = D^{-1} M_\sigma D \text{ for each } \sigma \in \Sigma, \qquad \mu_i' = D^{-1}\mu_i.$$

This is a similarity transformation: it rescales the states by their expected future mass, which redistributes the weights without changing what the automaton computes. One can verify that $[\![\mathcal{A}']\!](w) = [\![\mathcal{A}]\!](w)$ for every word w. What changes is that the

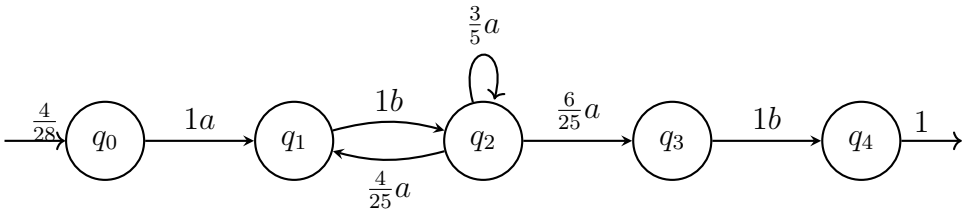

Figure 3: This is the machine we obtain after applying our acyclic normalisation algorithm to the machine in Figure 2

normalised SCC now satisfies local stochasticity at every state $q \in Q_i$:

$$\sum_{\sigma \in \Sigma} \sum_{q' \in Q_i} M'_\sigma(q, q') + \mu'_i(q) \;=\; 1.$$

3.3.2 Normalising the Acyclic Structure

Once each SCC has been normalised independently, we must handle the transitions *between* SCCs. After collapsing each SCC to a single super-state, the resulting graph is acyclic. We can therefore process the remaining states in reverse topological order, handling each state exactly once.

The key observation is that any self-loop at a state q acts as a geometric multiplier. A path that enters q may traverse the self-loop any number of times before eventually leaving, so its total contribution is multiplied by the geometric factor $\sum_{n=0}^{\infty} S(q)^n = \frac{1}{1-S(q)}$, where $S(q) = \sum_{\sigma \in \Sigma} M_\sigma(q, q)$ is the total self-loop weight at q. To normalise, we simply fold this factor into the outgoing transitions, removing the self-loop entirely and rescaling everything leaving q so that the semantics is preserved.

Concretely, for each state q processed in reverse topological order, let $V(q) = \sum_{\sigma \in \Sigma} \sum_{q' \neq q} M_\sigma(q, q') + \mu(q)$ denote the total non-self-loop outgoing mass. We update:

$$M'_\sigma(q, q') \;=\; \frac{M_\sigma(q, q')(1 - S(q))}{V(q)} \quad \text{for } q' \neq q$$

$$\mu'(q) \;=\; \frac{\mu(q)(1 - S(q))}{V(q)}, \qquad M'_\sigma(q, q) \;=\; 0.$$

Processing states in reverse topological order ensures that when we normalise q, all states reachable from q have already been handled, so the argument applies inductively. The result is that after processing every state, each one satisfies

$$\sum_{\sigma \in \Sigma} \sum_{q' \in Q} M'_\sigma(q, q') + \mu'(q) \;=\; 1.$$

Combining SCC normalisation with acyclic normalisation yields our main structural result: every weighted automaton over $\mathbb{R}_{\geq 0}$ with finite total mass admits an

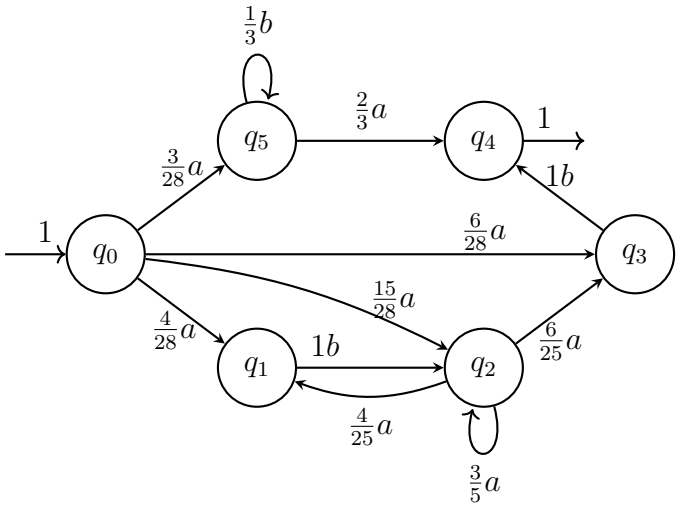

Figure 4: This automaton represents the normalised form of the automaton from Figure 1

equivalent locally stochastic representation, constructible in $O(n^4 + n^2|\Sigma|)$ time without increasing the number of states.

3.4 Stochastic Regular Expressions

A locally stochastic weighted automaton has clean probabilistic structure, but it is still a graph. We now show that every such automaton can be expressed as a closed-form algebraic expression, in the same way that every finite automaton can be expressed as a regular expression. We call these *stochastic regular expressions* (SREs).

SREs are built from four operations, each with a natural probabilistic interpretation.

Dirac distribution. For a symbol $\sigma \in \Sigma$, the expression δ_σ places all probability mass on the single-letter word σ. It is the atomic unit of the grammar, analogous to a single letter in a regular expression.

Convex combination. For two SREs r_1 and r_2 and a parameter $\alpha \in (0, 1)$, the expression $\alpha r_1 + (1 - \alpha) r_2$ defines a distribution that generates a word from r_1 with probability α and from r_2 with probability $1 - \alpha$. This is the probabilistic analogue of union.

Cauchy product. The expression $r_1 \cdot r_2$ generates a word by independently sampling u from r_1 and v from r_2, then concatenating them to produce uv. This is the probabilistic analogue of concatenation.

Discounted Kleene star. For an SRE r and a parameter $\alpha \in (0, 1)$, the expression r^*_α generates a word by repeatedly sampling from r, where the number of repetitions follows a geometric distribution with success probability α. Concretely,

$$[\![r^*_\alpha]\!](w) \;=\; \sum_{k=1}^{\infty} \sum_{\substack{w = w_1 \cdots w_k \\ w_i \in \Sigma^+}} \alpha(1-\alpha)^{k-1} \prod_{i=1}^{k} [\![r]\!](w_i).$$

This is the probabilistic analogue of the Kleene star, where the classical star iterates a language zero or more times, the discounted star iterates a distribution a geometrically random number of times.

The machine in Figure 3 has the corresponding SRE:

$$\frac{4}{28}\Big(ab\Big(\frac{4}{19}ab + \frac{15}{19}a\Big)^*_{\frac{19}{25}}\Big)ab$$

The stochastic regular expression for the entire machine in Figure 1 reduces to:

$$\frac{19}{28}\Big(\Big(\frac{4}{19}ab + \frac{15}{19}a\Big)\Big(\frac{4}{19}ab + \frac{15}{19}a\Big)^*_{\frac{19}{25}}\Big)ab \;+ \frac{6}{28}ab \;+ \frac{3}{28}a\Big(b\Big)^*_{\frac{1}{3}}a$$

The parallel with regular expressions is precise. Classical regular expressions are built from letters, union, concatenation, and Kleene star and they characterise exactly the regular languages. SREs are built from Dirac distributions, convex combination, Cauchy product, and discounted Kleene star.

Notice that every operator in the SRE grammar preserves the property of being a valid probability distribution: a convex combination of two distributions is a distribution, a Cauchy product of two distributions is a distribution, and a discounted Kleene star of a distribution is a distribution. We shall show that every weighted automaton that describes a probability distribution over strings can be written as an SRE, and every SRE describes a probability distribution over strings. The two representations are interchangeable.

3.4.1 Equivalence with Weighted Automata

The connection between SREs and locally stochastic weighted automata runs in both directions.

Theorem 2 (Kleene–Schützenberger for Stochastic Languages)**.** *A function $f : \Sigma^* \to \mathbb{R}_{\geq 0}$ is a rational stochastic language if and only if it is expressible as a stochastic regular expression. That is, $\mathcal{S}^{\mathrm{rat}}_{\mathbb{R}_{\geq 0}}(\Sigma^*)$ is precisely the class of functions definable by SREs.*

The forward direction (WA to SRE) proceeds by a state elimination argument

adapted to the probabilistic setting. Given a locally stochastic weighted automaton $\mathcal{A}$, we augment it with a fresh initial state q_s and final state q_f, connected to the original states via ε-transitions weighted by λ and μ respectively. We then eliminate states one at a time. When eliminating a state q_k, every path of the form $q_i \to q_k \to q_j$ is replaced by a direct edge from q_i to q_j whose label is the SRE

$$R_{ij} \;=\; T(q_i, q_k) \cdot \left(\sum_{\sigma} T(q_k, \sigma, q_k)\, \delta_\sigma \right)^*_{1-W_k} \cdot R_{kj}$$

where $W_k = \sum_{\sigma \in \Sigma} \sum_{q' \neq q_k} M_\sigma(q_k, q') + \mu(q_k)$ is the total non-self-loop outgoing weight of q_k. Local stochasticity guarantees that the self-loop weight satisfies $S(q_k) = 1 - W_k < 1$ at every state, so the discounted star $r^*_{1-W_k}$ is always well-defined and the geometric series it encodes always converges.

The backward direction (SRE to WA) proceeds by structural induction on the expression r. Each base case δ_σ yields a two-state automaton with a single transition of weight 1. For the inductive cases:

- **Convex combination.** Given automata $\mathcal{A}_1$ and $\mathcal{A}_2$ realising r_1 and r_2, introduce a fresh initial state and scale its outgoing ε-transitions to $\mathcal{A}_1$ and $\mathcal{A}_2$ by α and $1 - \alpha$ respectively.
- **Cauchy product.** Identify the final states of $\mathcal{A}_1$ with the initial states of $\mathcal{A}_2$ via ε-transitions of weight 1, removing the final weights of $\mathcal{A}_1$.
- **Discounted star.** Introduce a fresh initial state and a fresh final state. After each pass through $\mathcal{A}_1$, transition to the final state with probability α and loop back to the start of $\mathcal{A}_1$ with probability $1-\alpha$, realising the geometric distribution over repetitions.

One can verify by induction that each intermediate automaton is locally stochastic, so the construction stays within the class throughout. For full proofs and worked examples we refer the reader to the ICALP paper **icalp2026**.

Corollary 1 (Closure Properties)**.** *The class of stochastic regular languages is closed under:*

1. *Convex combinations: if f_1, f_2 are stochastic regular, so is $\alpha f_1 + (1-\alpha) f_2$*
2. *Cauchy products: if f_1, f_2 are stochastic regular, so is $f_1 \cdot f_2$*
3. *Discounted Kleene star: if f is stochastic regular, so is f^*_α*

3.5 Identity Testing for Stochastic Languages

A question that we wish to answer with the theorey we have developed is the following: given an unknown system generating strings, can we determine whether it matches a known reference distribution? This is the *identity testing* problem. Formally, we are given sample access to an unknown distribution P over Σ^* and a known reference distribution Q, and we wish to determine whether P and Q are close or far apart, measured in ℓ_1 distance:

$$\|P - Q\|_1 \;=\; \sum_{w \in \Sigma^*} |P(w) - Q(w)|.$$

Given parameters $\varepsilon > 0$ and $\delta \in (0, 1)$, an identity tester is a randomised algorithm that, with probability at least $1-\delta$, accepts if $\|P-Q\|_1 < \varepsilon$ and rejects if $\|P-Q\|_1 > \varepsilon$. The parameter δ is the *confidence parameter*: it controls the probability that the algorithm returns the wrong answer. A tester with $\delta = 0.05$ is wrong at most 5% of the time. In practice however, this parameter doesnt matter much. Given that our algorithm works for a certain confidence factor δ, we can always make it work for another factor δ' with minimal sample overhead.

Identity testing for distribution over finite domains and infinite continuous domains is well studied. The setting we care about is slightly different. Our distributions live on Σ^*, which is *infinite* and *discrete*. The classical algorithms and their sample complexity guarantees do not transfer directly.

3.5.1 Identity Testing via Truncation

We shall describe our approach to identity testing for stochastic languages, which we first developed in **datamod2025**. The central insight is that while Σ^* is infinite, rational stochastic languages have rapidly decaying tails, the probability mass assigned to words of length greater than θ becomes negligible for large enough θ. This suggests a natural reduction: truncate both distributions to strings of bounded length, and then apply a classical finite-domain tester to the resulting finite distributions. The challenge is making this reduction precise and computationally explicit.

Given a reference distribution Q represented as a weighted automaton $\mathcal{A}$ with joint transition matrix M, the tail decay rate is governed by the spectral radius $\rho(M)$. Specifically, the total mass of Q on words of length greater than θ is bounded above by a quantity that decays geometrically in θ at rate $\rho(M)$. This gives us a principled way to compute a truncation length θ such that

$$Q_{>\theta} \;:=\; \sum_{w \in \Sigma^{>\theta}} Q(w) \;<\; \frac{\varepsilon}{3}.$$

With this θ in hand, the identity testing algorithm proceeds as follows. Draw N samples from P, discard all samples of length greater than θ, and construct the empirical distribution $\hat{P}_{\leq\theta}$ over the finite domain $\Sigma^{\leq\theta}$. Separately, compute the restriction $Q_{\leq\theta}$ of Q to $\Sigma^{\leq\theta}$ from the automaton. Then run a finite-domain tolerant identity tester **canonne2022** on $\hat{P}_{\leq\theta}$ and $Q_{\leq\theta}$ with distance thresholds $(\varepsilon/3, \varepsilon)$.

The correctness of this reduction follows from a clean decomposition of the ℓ_1 distance. The total distance $\|P - Q\|_1$ splits into a head term over $\Sigma^{\leq\theta}$ and a tail term over $\Sigma^{>\theta}$. The tail term is bounded by $2\varepsilon/3$ by construction of θ, so the finite tester need only resolve the head term to within $\varepsilon/3$ — which it does by design. The sample complexity of the overall algorithm is $N = \Theta(\sqrt{k}/\varepsilon^2 + k/\log k)$ where $k = |\Sigma^{\leq\theta}|$ is the size of the truncated domain.

This is a meaningful and correct result. The truncation argument is robust, the reduction to the finite setting is tight, and the sample complexity matches what one would hope for given the effective support size k. The fact that the macine representation is a weighted automaton means that we can very efficiently calculate the spectral radius and therefore efficiently bound the maximum size of the word that we need to consider.

That said, the approach has a structural limitation that becomes apparent when one asks how θ depends on the reference distribution Q. The truncation length is derived from a global spectral computation on the automaton $\mathcal{A}$: one must compute eigenvalues, identify the dominant one, and bound the associated constants. Beyond the efficient calculation of the bounding length and the spectral radius we haven't properly utilised the unique form of weighted automata.

3.5.2 Towards Instance-Optimal Testing

Pearson's chi-squared test **pearson1900** is the classical starting point for identity testing. It aggregates normalised squared deviations between observed and expected frequencies, but its sample complexity of $O(n/\varepsilon^2)$ for distributions over a domain of size n is suboptimal.

Valiant and Valiant **valiant2017** gave a significant refinement. They propose a modified chi-squared style tester and show that it achieves *instance-optimal* sample complexity. Their tester distinguishes $p = q$ from $\|p - q\|_1 \geq \varepsilon$ using $f(p, \varepsilon)$ samples with success probability greater than $2/3$, and no tester can distinguish $p = q$ from $\|p - q\|_1 \geq c \cdot \varepsilon$ using fewer than $c' \cdot f(p, \varepsilon)$ samples, for universal constants c, c'. The function $f(p, \varepsilon)$ is upper bounded by a multiple of $\|p\|_{2/3}/\varepsilon^2$, where $\|p\|_{2/3}$ is the $\ell_{2/3}$ norm of the reference distribution. The key quantity driving this bound is

$$p_{\max} \;=\; \max_x Q(x),$$

the largest probability assigned by Q to any single element of the domain. If Q is highly concentrated, identity testing is easier and fewer samples suffice. What we sought to understand in our new work is whether, when Q is a rational stochastic language presented as an SRE, we can compute $p_{\max}$ directly from the syntax of the expression, making the instance-optimal guarantee of Valiant and Valiant a practically achievable target.

3.5.3 Computing the Maximum-Probability Word

We describe a branch-and-bound procedure that computes this quantity exactly when $\mathcal{A}$ is presented as a weighted automaton over $\mathbb{R}_{\geq 0}$ with $\rho(M) < 1$.

For a prefix w, let the forward vector be $\alpha(w) := \lambda^\top M$, so $\mathrm{wt}(w) = \alpha(w) \cdot \mu$. Fixing any submultiplicative norm, contractivity gives constants $C > 0$ and $\gamma \in (0, 1)$ with $\|M\| \leq C\gamma^{|x|}$ for every x. Summing $\mathrm{wt}(wx) \leq \|\alpha(w)\| \cdot C \cdot \|\mu\| \cdot \gamma^{|x|}$ over all extensions yields the admissible upper bound

$$\mathrm{UB}(w) \;:=\; \mathrm{wt}(w) + K \cdot \|\alpha(w)\|, \qquad K := \frac{C\|\mu\|}{1-\gamma}.$$

The algorithm maintains a priority queue of pairs $(w, \alpha(w))$ ordered by UB, together with the best weight B found so far. At each step it extracts the pair with largest UB; if $\mathrm{UB}(w) \leq B$ it terminates, otherwise it updates B if $\mathrm{wt}(w) > B$ and pushes each child $(wc, \alpha M_c)$ whose upper bound still exceeds B.

Correctness is immediate: pruning is safe because UB upper-bounds every continuation, and termination occurs exactly when no queued prefix can improve on B. For finite termination, $\|\alpha(w)\|$ decays as $\gamma^{|w|}$, so $\mathrm{UB}(w) \leq B$ beyond depth $O(\log B / \log(1/\gamma))$; the search tree is therefore finitely explored, at worst-case cost $O(|\Sigma|^{d_{\mathrm{eff}}})$ times a matrix–vector product per node.

4 Extensions and Future Work

The most compelling open directions sit at the interface between the SRE characterisation and distribution testing problems.

Instance-optimal testing via SRE syntax. Our truncation-based tester is correct but not instance-optimal in the sense of Valiant and Valiant, whose framework relies on $p_{\max} = \max_w [\![\mathcal{A}]\!](w)$ and n such that $\sum_{|w|>n} [\![\mathcal{A}]\!](w) < \varepsilon/16$. When $\mathcal{A}$ is given as an SRE one would like to compute $p_{\max}$ and n directly from the syntax. The difficulty is that probability mass does not filter up cleanly through the operators. Under Cauchy product, each word w receives a contribution from every decomposition $w = uv$, so

the heaviest word is in general not the concatenation of the heaviest words from the components of the SRE. We face similar problems with the convex combination. Our branch-and-bound procedure gives a workable computational handle on $p_{\max}$ using the spectral radius, but not a closed-form syntactic recursion. Either a polynomial-time syntactic algorithm for $p_{\max}$ or an unconditional hardness result showing that none exists would be valuable: the former delivers an instance-optimal tester for rational stochastic languages, while the latter would itself sharpen our understanding of what the SRE representation does and does not expose.

Tropical and other semirings. Our normalisation extends beyond $\mathbb{R}_{\geq 0}$ to the tropical semiring, where the cycle mean plays the role of the spectral radius. Whether a tropical analogue of the SRE grammar exists, and what properties a semiring must satisfy to admit a Kleene-style stochastic characterisation, remain open.

5 Conclusion

This thesis studied the class of weighted automata over $\mathbb{R}_{\geq 0}$ that define probability distributions over strings, and developed a new characterisation of this class together with its consequences for distribution testing.

We began by showing that whether a weighted automaton defines a probability distribution is a global rather than local property of the machine, settled by the spectral radius of its joint transition matrix. Building on this, we gave a normalisation procedure that rewrites any finite-mass weighted automaton into an equivalent locally stochastic one, in which each state's outgoing and final weights sum to exactly one. The procedure decomposes the automaton along its strongly connected components, normalises each SCC using the expected future mass vector, and then normalises the resulting acyclic structure in reverse topological order.

With locally stochastic automata in hand, we introduced stochastic regular expressions: a grammar built from Dirac distributions, convex combination, Cauchy product, and a discounted Kleene star, each of which preserves the property of being a probability distribution. We proved a Kleene–Schützenberger theorem for stochastic languages, establishing that SREs and locally stochastic weighted automata define the same class of distributions over Σ^*. This gives a compositional, syntactically transparent account of rational stochastic languages and, as a consequence, closure of the class under convex combination, Cauchy product, and discounted Kleene star.

Finally, we turned to identity testing. We gave a truncation-based algorithm that reduces identity testing for rational stochastic languages to the classical finite-domain setting, using the spectral radius to bound the tail mass and a finite-domain tolerant tester on the truncated support. We then outlined how the SRE characterisation

opens a path toward instance-optimal testing in the sense of Valiant and Valiant, and identified the syntactic computation of $p_{\max}$ as the central obstruction.

Taken together, these results give a unified picture: weighted automata with finite mass are probabilistic machines up to normalisation, SREs are their grammar characterisation, and future work can use the inductive structure of SREs to come up with good distribution testing algorithms.

6 Acknowledgment

I would dearly like to thank Professor Aalok Thakkar for the constant support and the time-consuming training that he provided to me over the course of the thesis. I am certain that a thesis with anyone else would not have been as fruitful as it has been with Professor Thakkar.