A faster FPRAS for #NFA*

KULDEEP S. MEEL, University of Toronto, Canada ⓡ
SOURAV CHAKRABORTY, Indian Statistical Institute, India ⓡ
UMANG MATHUR, National University of Singapore, Singapore ⓡ

Given a non-deterministic finite automaton (NFA) $\mathcal{A}$ with m states, and a natural number $n \in \mathbb{N}$ (presented in unary), the #NFA problem asks to determine the size of the set $\mathcal{L}(\mathcal{A}_n)$ of words of length n accepted by $\mathcal{A}$. While the corresponding decision problem of checking the emptiness of $\mathcal{L}(\mathcal{A}_n)$ is solvable in polynomial time, the #NFA problem is known to be #P-hard. Recently, the long-standing open question — whether there is an FPRAS (fully polynomial time randomized approximation scheme) for #NFA — was resolved by Arenas, Croquevielle, Jayaram, and Riveros in [3]. The authors demonstrated the existence of a fully polynomial randomized approximation scheme with a time complexity of $\widetilde{O}\left(m^{17}n^{17} \cdot \frac{1}{\epsilon^{14}} \cdot \log(1/\delta)\right)$, for a given tolerance ϵ and confidence parameter δ.

Given the prohibitively high time complexity in terms of each of the input parameters, and considering the widespread application of approximate counting (and sampling) in various tasks in Computer Science, a natural question arises: is there a faster FPRAS for #NFA that can pave the way for the practical implementation of approximate #NFA tools? In this work, we answer this question in the positive. We demonstrate that significant improvements in time complexity are achievable, and propose an FPRAS for #NFA that is more efficient in terms of both time and sample complexity.

A key ingredient in the FPRAS due to Arenas, Croquevielle, Jayaram, and Riveros [3] is inter-reducibility of sampling and counting, which necessitates a closer look at the more informative measure — the number of samples maintained for each pair of state q and length $i \leq n$. In particular, the scheme of [3] maintains $O(\frac{m^7n^7}{\epsilon^7})$ samples per pair of state and length. In the FPRAS we propose, we systematically reduce the number of samples required for each state to be only poly-logarithmically dependent on m, with significantly less dependence on n and ϵ, maintaining only $\widetilde{O}(\frac{n^4}{\epsilon^2})$ samples per state. Consequently, our FPRAS runs in time $\widetilde{O}((m^2n^{10} + m^3n^6) \cdot \frac{1}{\epsilon^4} \cdot \log^2(1/\delta))$. The FPRAS and its analysis use several novel insights. First, our FPRAS maintains a weaker invariant about the quality of the estimate of the number of samples for each state q and length $i \leq n$. Second, our FPRAS only requires that the distribution of the samples maintained is close to uniform distribution only in total variation distance (instead of maximum norm). We believe our insights may lead to further reductions in time complexity and thus open up a promising avenue for future work towards the practical implementation of tools for approximate #NFA.

CCS Concepts: • **Theory of computation → Theory and algorithms for application domains**.

Additional Key Words and Phrases: #NFA, counting, estimation, FPRAS

ACM Reference Format:
Kuldeep S. Meel ⓡ Sourav Chakraborty ⓡ Umang Mathur. 2024. A faster FPRAS for #NFA. *Proc. ACM Manag. Data* 2, 2 (PODS), Article 112 (May 2024), 22 pages. https://doi.org/10.1145/3651613

*All authors contributed equally to this research. The symbol ⓡ denotes random author order. The publicly verifiable record of the randomization is available at https://www.aeaweb.org/journals/policies/random-author-order

Authors' addresses: Kuldeep S. Meel, University of Toronto, Toronto, Canada, meel@cs.toronto.edu; Sourav Chakraborty, Indian Statistical Institute, Kolkata, India, sourav@isical.ac.in; Umang Mathur, National University of Singapore, Singapore, Singapore, umathur@comp.nus.edu.sg.

ACM 2836-6573/2024/5-ART112
https://doi.org/10.1145/3651613

1 INTRODUCTION

In this paper, we focus on the following computational problem:

#NFA : Given an NFA $\mathcal{A} = (Q, I, \Delta, F)$ over the binary alphabet $\Sigma = \{0, 1\}$ with m states, and a number n in unary, determine $|\mathcal{L}(\mathcal{A}_n)|$, where $\mathcal{L}(\mathcal{A}_n)$ is the set of strings of length n that are accepted by $\mathcal{A}$.

The problem of #NFA is a fundamental problem in Computer Science, with range of applications in database and information extraction systems; we outline a few below.

Probabilistic Query Evaluation. Given a query Q and a database D, the problem of probabilistic query evaluation (PQE) is to determine the probability of the query holding on a randomly sampled database, in which each row is included independently with probability given by its annotated value. PQE is known to be #P-hard, even in data complexity setting. For a large class of queries, PQE can directly be reduced to the #NFA problem. In particular, when the schema of the database D consists only of binary relations, and when the query Q is a self-join-free path query, then the PQE problem for D and Q reduces to an instance of #NFA, whose size is linear in the size of Q and linear in the size of D [17], and further, the time to compute the reduction is also linear in D and Q. Thus an efficient algorithm for solving #NFA directly yields an efficient algorithm for probabilistic query evaluation.

Counting Answers to Reglar Path Queries. Regular path queries (or *property paths*) [2] form a rich fragment of graph query languages such as SPARQL designed for graph database systems. Here, one abstracts information on edges in the graph database using an alphabet Σ and asks a variety of questions about the set of paths that start from a given source node u, end at a target node v. In particular, a regular path query (uRv) asks to enumerate, count or uniformly sample the set of paths (bounded in length by some fixed number n) from u to v whose labels match the regular expression R. Therefore, the problem of counting the number of answers to the queries reduces to the #NFA problem, for the NFA obtained by the cross product of NFA represented by the database (with u as the intial state and v as the final state) and the NFA that R gets compiled down to. This reduced instance is again linear in the size of the database as well as the query, and the reduction takes the same time [3].

Probabilistic Graph Homomorphism. A probabilistic graph is a pair (H, π), where H is a graph and π is the associated probability labeling over edges. The associated probability space is defined as the set of all subgraphs of H wherein every edge e is sampled independently with probability $\pi(e)$. Given a query Graph G and a probabilistic graph (H, π) the problem of *probabilistic graph homomorphism problem* asks to to compute the probability that a randomly sampled subgraph of H admits a homomorphism from G. In case of 1-way (and 2-way) path queries, the problem was again shown to reduce to #NFA [1].

Beyond databases. From a complexity theoretic viewpoint, #NFA is SpanL-complete and therefore, finds applications in domains beyond databases such as program verification [13], testing of software [16] and distributed systems [14], probabilistic programming, estimation of information leakage in software [5, 7, 15], information extraction [4], automated reasoning using BDDs [4], and even music generation [6].

The direct applications of the #NFA problem brings up the central computational question — how do we count the number of elements of $\mathcal{L}(\mathcal{A}_n)$? It turns out that #NFA is, in fact, provably computationally intractable — it is known to be a #P-hard problem [18]. Given the intractability, and given the widespread applications of the #NFA problem, the next natural question then arises — can we approximate #NFA using a polynomial time algorithm? Until very recently, the only approximation algorithms known for #NFA had running times that were *quasi-polynomial* in

the size of the inputs [8, 11]. In a recent breakthrough by Arenas, Croquevielle, Jayaram and Riveros [3, 4], #NFA was finally shown to admit *fully polynomial time randomized approximation scheme* (FPRAS).

One would assume that the discovery of FPRAS for #NFA would lead to design of scalable tools for all the different applications outlined above. Such has, however, not been the case and the primary hurdle to achieving practical adoption has been the prohibitively large time complexity of the proposed FPRAS– it takes time $\widetilde{O}\left(m^{17}n^{17} \cdot \frac{1}{\epsilon^{14}} \cdot \log(1/\delta)\right)$, where $\widetilde{O}$ hides the poly-logarithmic multiplicative factors in m and n. It is worth highlighting that for most applications, the reduction to #NFA instance often only incurs a blow up that is a low-degree polynomial (linear or quadratic) factor larger than the size of the original problem. This means that the bottleneck to solving the original counting problem is really the high runtime complexity involved in the (approximate) counting algorithm and not the reduction.

In this work, we pursue the goal of designing a faster algorithm, and propose an FPRAS which runs in time $\widetilde{O}((m^2n^{10}+m^3n^6)\cdot\frac{1}{\epsilon^4}\cdot\log^2(1/\delta))$. The reliance of both FPRASes on the inter-reducibility of sampling and counting necessitates a more informative measure, such as the number of samples maintained for each state. The scheme proposed in [4] maintains $O(\frac{m^7n^7}{\epsilon^7})$ samples per state while the scheme proposed in this work maintains only $O(\frac{n^4}{\epsilon^2})$ samples, which is independent of m and with a significant reduction in its dependence on n and ϵ.

1.1 Overview of our FPRAS

At a high-level, our approach exploits the tight connection between approximate counting and the related problem of (almost) uniform generation. In the context of an NFA $\mathcal{A}$ and a length n, the almost uniform generation problem asks for a randomized algorithm (generator) G, such that, on each invocation, the probability with which G returns any given string $x \in \mathcal{L}(\mathcal{A}_n)$ lies in the range $[\frac{1}{|\mathcal{L}(\mathcal{A}_n)|}(1-\epsilon), \frac{1}{|\mathcal{L}(\mathcal{A}_n)|}(1+\epsilon)]$; here $\epsilon \in (0,1)$ is a parameter. Such a random generator is a polynomial time generator if its running time is polynomial in $|\mathcal{A}|$, n and $1/\epsilon$. The existence of a polynomial time almost uniform generator for *self-reducible* [10] problems implies the existence of an FPRAS, and vice versa.

This tight connection between counting and sampling was also the high-level idea behind the work of [4]. Consequently, our algorithm fits the template procedure of Fig. 1, also outlined in [4]. For technical convenience, the algorithm unrolls the automaton $\mathcal{A}$ into an acyclic graph[1] with n levels; each level ℓ containing the ℓ^{th} copy q^ℓ of each state q of $\mathcal{A}$. A key component in the algorithm is to incrementally compute an estimate $\mathsf{N}(q^\ell)$ of the size of the set $\mathcal{L}(q^\ell)$ of words of length ℓ that have a run ending in state q. Next, in order to inductively estimate $\mathsf{N}(q^\ell)$, we in turn rely on the estimates of sets $\mathcal{L}(p_0{}^{\ell-1})$ and $\mathcal{L}(p_1{}^{\ell-1})$; here p_b is a b-predecessor state of q ($\mathsf{b} \in \Sigma = \{0,1\}$), i.e., $(p_\mathsf{b}, \mathsf{b}, q) \in \Delta$ is a transition of $\mathcal{A}$. These estimates can be put to use effectively because:

$$\mathcal{L}(q^\ell) = \Big(\bigcup_{(p_0,\mathtt{0},q)\in\Delta} \mathcal{L}(p_0{}^{\ell-1})\Big)\cdot\{\mathtt{0}\} \uplus \Big(\bigcup_{(p_1,\mathtt{1},q)\in\Delta} \mathcal{L}(p_1{}^{\ell-1})\Big)\cdot\{\mathtt{1}\}.$$

We remark that the sets $\{\mathcal{L}(p^{\ell-1})\}_p$ may not be disjoint for different predecessor states p. Hence, we cannot estimate $\bigcup_{(p_\mathsf{b},\mathsf{b},q)\in\Delta}\mathcal{L}(p_\mathsf{b}{}^{\ell-1})$ by merely summing the individual estimates $\{\mathsf{N}(p_\mathsf{b}{}^{\ell-1}) \mid (p_\mathsf{b},\mathsf{b},q) \in \Delta\}$. To address this, we maintain a set, $\mathsf{S}(p^{\ell-1})$, for each p, which contains samples of strings from $\mathcal{L}(p^{\ell-1})$. One possible idea is to construct the set $\mathsf{S}(p^{\ell-1})$ using strings w that have been sampled uniformly at random from $\mathcal{L}(p^{\ell-1})$ but such a process introduces dependence among the sets $\mathsf{S}(p^{\ell-1})$ for different p. The crucial observation in [4] is that for any

[1]The assumption in the template algorithm of Fig. 1 that $\mathcal{A}$ has only 1 accepting state is without loss of generality.

Fig. 1: Algorithm Template: FPRAS for #NFA

Input : NFA $\mathcal{A}$ with m states Q and a single final state $q_F \in Q$, $n \in \mathbb{N}$ (in unary)

1 Unroll the automaton $\mathcal{A}$ into an acyclic graph $\mathcal{A}_{\mathsf{unroll}}$ by making $n+1$ copies $\{q^\ell\}_{\ell \in [0,n], q \in Q}$ of the states Q and adding transitions between immediate layers

2 **foreach** $\ell \in \{0, \ldots, n\}, q \in Q$ **do**

3 Compute the estimate $\mathsf{N}(q^\ell)$ using prior estimates and samples $\{\mathsf{N}(p^i), \mathsf{S}(p^i)\}_{i<\ell, p \in Q}$

4 Compute a uniform set of samples $\mathsf{S}(q^\ell)$ using $\mathsf{N}(q^\ell)$ and the prior estimates and samples $\{\mathsf{N}(p^i), \mathsf{S}(p^i)\}_{i<\ell, p \in Q}$

5 **return** $\mathsf{N}(q_F{}^n)$

k, the following is true: for any word w of length k, all words from $\mathcal{L}(q^\ell)$ with the suffix w can be expressed as $\left(\bigcup_{p \in P} \mathcal{L}(p^{\ell-k})\right) \cdot \{w\}$ for some set P of states, where P depends only on w, q, and k. This property is referred to as the *self-reducible union* property. This insight allows us to utilize the *self-reducibility* property for sampling. We can partition $\mathcal{L}(p^{\ell-1})$ into sets of strings where the last character is either `0` or `1`, and compute estimates for each partition since each partition can be represented as a union of sets for which we (inductively) have access to both samples and estimates. We then sample the final character in proportion to the size of these estimates. Similarly, we can sample strings, character by character. That is, we can sample strings by progressively extending suffixes of w backwards! Of course, we must also account for the errors that propagate in the estimates of $\mathsf{N}(q^\ell)$.

Since our algorithm also relies on the *self-reducible union* property, the basic structure of our algorithm is similar to that of [4] on high-level. There are, however, significant and crucial technical differences that allow us to obtain an FPRAS with significantly lower time complexity. We highlight these differences below by first highlighting the two key properties underpinning the algorithm and ensuing analysis in [4]:

(ACJR-1). For every level ℓ, the following condition holds with high probability (here, $\kappa = \frac{nm}{\epsilon}$):

$$\xi(\ell) \triangleq \forall q \in Q, \forall P \subseteq Q, \left| \frac{|\mathsf{S}(q^\ell) \setminus \bigcup_{p \in P} \mathcal{L}(p^\ell)|}{|\mathsf{S}(q^\ell)|} - \frac{|\mathcal{L}(q^\ell) \setminus \bigcup_{p \in P} \mathcal{L}(p^\ell)|}{|\mathcal{L}(q^\ell)|} \right| \leq \frac{1}{\kappa^3}$$

(ACJR-2). $\mathsf{S}(q^\ell)$ is close in (L_∞ distance)[2] to the distribution of multi-set obtained by independently sampling (with replacement) every element of $\mathcal{L}(q^\ell)$ uniformly at random.

In contrast, the FPRAS we propose maintains the following two weaker invariants:

(Inv-1). For every state q and level ℓ, the event that $\mathsf{N}(q^\ell) \in (1 \pm \frac{\epsilon}{2n^2})|\mathcal{L}(q^\ell)|$, which we denote as $\mathtt{AccurateN}_{q,\ell}$, happens with high probability.

(Inv-2). The distribution of $\mathsf{S}(q^\ell)$ is close, *in total variation distance*, to the distribution of a multi-set constructed by sampling (with replacement) every element of $\mathcal{L}(q^\ell)$ uniformly at random.

Two remarks are in order. First, $\xi(\ell)$ implies $\mathtt{AccurateN}_{q,\ell}$, and closeness in L_∞ is a stringent condition than closeness in total variation distance. Second, the technical arguments of [4] crucially rely on their invariants and do not work with the aforementioned weaker arguments. Accordingly, this necessitates a significantly different analysis that depends on coupling of the distributions.

[2] In [4], the invariant is stated in a different, but equivalent, formulation: conditioned on $\xi(\ell)$, the set $\mathsf{S}(q^\ell)$ follows the distribution of a multi-set obtained by independently sampling every element of $\mathcal{L}(q^\ell)$ uniformly at random, with replacement.

It's also noteworthy that, due to our reliance on the weaker condition, our method for estimating $\mathsf{N}(q^\ell)$ differs from that in [4]. In fact, our approach is based on an adaptation of the classical Monte Carlo-based computation of the union of sets [12].

For a given state q and subset P, achieving a $\frac{1}{\kappa^3}$-additive approximation for $\frac{|\mathcal{L}(q^\ell)\setminus\cup_{p\in P}\mathcal{L}(p^\ell)|}{|\mathcal{L}(q^\ell)|}$ with a probability of $1-\eta$ requires, according to standard concentration bounds, that the size of $\mathsf{S}(q^\ell)$ be $O(\kappa^6 \log \eta^{-1})$. To ensure this additive approximation for all states q and subsets P, one must consider union bounds over exponentially many events. Specifically, for the analysis to be successful, one must have $\eta^{-1} \in O(2^{mn})$. Consequently, the FPRAS of [4] maintains $O(\kappa^7)$ samples for each state q. In contrast, for $\mathtt{AccurateN}_{q,\ell}$ to hold with a probability of $1-\eta$, our FPRAS only maintains $O\left(\frac{n^4}{\epsilon^2}\right)\log \eta^{-1}$ samples in $\mathsf{S}(q^\ell)$. Furthermore, a union bound over only $O(mn)$ events is required, thus setting $\eta^{-1} = O(mn)$ is adequate.

The time complexity of an algorithm that follows the template outlined in Fig. 1 (including both the FPRAS of [4] as well as the FPRAS developed in this paper), has a quadratic depends on $|\mathsf{S}(q^\ell)|$. Accordingly, it is instructive to compare the bound of $O(\frac{m^7 n^7}{\epsilon^7})$ for $|S(q^\ell)|$ in [4] to $O(\frac{n^4}{\epsilon^2})$ in our approach. Remarkably, the number of samples for every state in our approach is independent of m, the size of the automaton.

Organization. In Section 2 we recall some helpful background as well as introduce a key concept (distribution entanglement) useful for the rest of the paper. In Section 3, we describe our algorithm, together with two auxiliary algorithms and state their formal correctness guarantees, whose analysis we present in Section 4. We conclude in Section 5.

2 PRELIMINARIES

NFA and words. We consider the binary alphabet $\Sigma = \{0, 1\}$. All results in this paper are extendable to alphabets of arbitrary but fixed constant size. A string w over Σ is either the empty string λ (of length 0) or a non-empty finite sequence $w_1 w_2 \ldots w_k$ (of length $|w| = k > 0$) where each $w_i \in \Sigma$. The set of all strings over Σ is denoted by Σ^*. A non-deterministic finite automaton is a tuple $\mathcal{A} = (Q, I, \Delta, F)$ where Q is a finite set of states, $I \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and F is the set of accepting states. A run of w on $\mathcal{A}$ is a sequence $\rho = q_0, q_1 \ldots, q_k$ such that $k = |w|$, $q_0 = I$ and for every $i < k$, $(q_i, w_{i+1}, q_{i+1}) \in \Delta$; ρ is accepting if $q_k \in F$. The word w is accepted by $\mathcal{A}$ if there is an accepting run of w on $\mathcal{A}$. The set of strings accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A})$. For a natural number $n \in \mathbb{N}$, the n^{th} slice of $\mathcal{A}$'s language, $\mathcal{L}(\mathcal{A}_n)$, is the set of strings of length n in $\mathcal{L}(\mathcal{A})$. For simplicity, this paper assumes a singleton set of accepting states, i.e., $F = \{q_F\}$ for some $q_F \in Q$. For $q \in Q$ and $\mathsf{b} \in \Sigma$, the b-predecessors of q are given by the set $\mathsf{Pred}(q, \mathsf{b}) = \{p \mid (p, \mathsf{b}, q) \in \Delta\}$.

Given automaton $\mathcal{A}$ and a number n in unary, we construct the unrolled automaton $\mathcal{A}_{\mathsf{unroll}}$, and assume that all states in $\mathcal{A}_{\mathsf{unroll}}$ are reachable from the initial state. We use the notation q^ℓ to denote the ℓ^{th} copy of state $q \in Q$. For each state q and level $0 \le \ell \le n$, we use $\mathcal{L}(q^\ell)$ to denote the the set of all distinct strings w of length ℓ for which there is a path (labeled by w) from the starting state to q.

Total Variation Distance. The total variation distance between two random variables X, Y over domain Ω is defined as:

$$\mathsf{TV}(X, Y) = \sum_{\omega\in\Omega} \Pr[X = \omega] - \min(\Pr[X = \omega], \Pr[Y = \omega]).$$

2.1 A New Notation: Distribution Entanglement

We introduce a new notation, referred to as *distribution entanglement*, to argue about the probability of events for cases when the distribution of a certain random variable were to mimic distribution of another random variable. For an event $\mathcal{E}$ and random variables X and Y, we use $\mathsf{Pr}_X[\mathcal{E}; Y]$ to denote the probability of $\mathcal{E}$ when X has *distribution entangled to* Y, which is defined as follows:

$$\mathsf{Pr}_X[\mathcal{E}; Y] = \sum_{\omega \in \Omega} \mathsf{Pr}[Y = \omega] \cdot \mathsf{Pr}[\mathcal{E} | X = \omega] \,.$$

where Ω is the support of Y. The above notion is well-defined only when it is the case for all $\omega \in \Omega$, we have $\mathsf{Pr}[X = \omega] > 0$. Likewise, we can also extend this notion for conditioned events. That is, for events $\mathcal{E}, \mathcal{F}$ and random variables X and Y, the conditional probability of $\mathcal{E}$ given $\mathcal{F}$ when X has distribution entangled to $Y$is defined as follows:

$$\mathsf{Pr}_X[\mathcal{E} | \mathcal{F}; Y] = \sum_{\omega \in \Omega} \mathsf{Pr}[Y = \omega] \cdot \mathsf{Pr}[\mathcal{E} | \mathcal{F} \cap X = \omega] \,.$$

Whenever X is clear from the context, we will omit it from the notation.

3 A FASTER FPRAS

In this section we formally present our FPRAS for #NFA, together with the statement of its correctness and running time (Theorem 3). Our FPRAS Algorithm 3 calls two subroutines `AppUnion` (Algorithm 1) and `sample` (Algorithm 2), each of which are presented next, together with their informal descriptions and formal statements of correctness. Algorithm 3 presents the main procedure for approximating the size of $\mathcal{L}(\mathcal{A}_n)$, where $\mathcal{A}$ is the NFA and n is the parameter that is given as input. The algorithm works in a dynamic programming fashion and maintains, for each state q and each length $1 \leq \ell \leq n$, a sampled subset $\mathsf{S}(q^\ell)$ of $\mathcal{L}(q^\ell)$. As a key step, it uses the sampled sets corresponding to $\mathsf{Pred}(q, 0)$ and $\mathsf{Pred}(q, 1)$ to construct the desired sampled subset of $\mathcal{L}(q^\ell)$. We first describe the two subroutines the `AppUnion` and `sample`.

3.1 Approximating Union of Sets

`AppUnion` (Algorithm 1) takes parameters ϵ, δ and inputs ϵ_{sz} and (access to) sets $T_1, \ldots, T_k$. For each set T_i the access is given in the form of a membership oracle O_i, a subset S_i (obtained by sampling T_i with replacements) presented as a list, and an estimate sz_i of the size of T_i. Using these, Algorithm 1 outputs an (ϵ, δ) estimate of the size of the union $\cup_{i=1}^{k} T_i$.

The precise algorithm presented here represents a modification of the classic Monte Carlo-based scheme due to Karp and Luby [12]. The proof of its correctness also shares similarities with [12] and we summarize the intuition here. For a $\sigma \in \cup_{i=1}^{k} T_i$, let $L(\sigma)$ denote the smallest index $i \in \{1, 2, \ldots, k\}$ such that $\sigma \in T_i$. Now, let $\mathcal{U}_{\mathsf{unique}}$ be the set of all pairs $(\sigma, L(\sigma))$ and let $\mathcal{U}_{\mathsf{multiple}}$ be the set of all (σ, i) pairs such that $\sigma \in T_i$. Consequently, the cardinality of the set $\mathcal{U}_{\mathsf{unique}}$ is $|\cup_{i=1}^{k} T_i|$, while the cardinality of $\mathcal{U}_{\mathsf{multiple}}$ is $\sum_{i=1}^{k} |T_i|$. By drawing sufficiently-many (t, to be precise) samples from $\mathcal{U}_{\mathsf{multiple}}$ and assessing the fraction belonging to $\mathcal{U}_{\mathsf{unique}}$, one can estimate $|\mathcal{U}_{\mathsf{unique}}|/|\mathcal{U}_{\mathsf{multiple}}|$, which when multiplied by $|\mathcal{U}_{\mathsf{multiple}}|$ provides an estimate for the size of $\mathcal{U}_{\mathsf{unique}}$, which is the desired output.

After sampling an index i, in Line 7 an element is chosen (and removed) from the sampled list S_i. In the case when the set S_i is constructed by uniformly selecting elements from T_i, this step mimics drawing a random sample from the actual set T_i. Initially, each set (more precisely, list) S_i is guaranteed to have a size of at least $\mathtt{thresh} = 24 \cdot \frac{(1+\epsilon_{sz})^2}{\epsilon^2} \cdot \log(\frac{4k}{\delta})$, surpassing the expected number of samples required from T_i during the t iterations of the loop. If the algorithm ever needs more samples than $|S_i|$ during the t iterations, it counts as an error, but the probability of this

Algorithm 1: Approximating Union of Sets

1 **function** $\mathtt{AppUnion}_{(\epsilon,\delta)}(\epsilon_{sz}, (O_1, S_1, sz_1), \ldots, (O_k, S_k, sz_k))$
2 $\quad m \leftarrow \left\lceil \frac{\sum_j \mathrm{sz}_j}{\max_j \mathrm{sz}_j} \right\rceil$
3 $\quad t \leftarrow \frac{12 \cdot (1+\epsilon_{sz})^2 \cdot m}{\epsilon^2} \cdot \log(\frac{4}{\delta})$
4 $\quad Y \leftarrow 0$
5 $\quad$ **repeat** t **times:**
6 $\quad\quad$ Sample $i \in \{1, \ldots, k\}$ with probability $p_i = \frac{\mathrm{sz}_i}{\sum_j \mathrm{sz}_j}$
7 $\quad\quad$ **if** $S_i \neq \emptyset$ **then** $\sigma \leftarrow S_i$.deque()
8 $\quad\quad$ **else break**
9 $\quad\quad$ **if** $\neg(\exists j < i, O_j.\mathtt{contains}(\sigma))$ **then** $Y \leftarrow Y + 1$
10 $\quad$ **return** $\frac{Y}{t} \cdot (\sum_i \mathrm{sz}_i)$

occurrence will be demonstrated to be very low. Finally, Line 9 verifies whether the sample from $\mathcal{U}_{\mathsf{multiple}}$ belongs to $\mathcal{U}_{\mathsf{unique}}$ by asking a membership question to the oracle O_i. If all the sampled sets S_i's are constructed uniformly, then the variable Y tallies the number of samples from $\mathcal{U}_{\mathsf{unique}}$, and after t iterations, Y/t provides a estimation for $|\mathcal{U}_{\mathsf{unique}}|/|\mathcal{U}_{\mathsf{multiple}}|$. The final output is the product of this value with $(\sum_i \mathrm{sz}_i)$ (which is the estimate for $|\mathcal{U}_{\mathsf{multiple}}|$).

We next present (in Theorem 1) the formal correctness statement of this algorithm using distribution entanglement (see Section 2.1). For this, we set the source random variable X to be the one that corresponds to the product distribution of the input sampled sets $S_1, \ldots, S_k$, while the target random variable Y corresponds to the product distribution of $\mathsf{U}_1, \ldots, \mathsf{U}_k$, where, U_i, is the random variable that obeys the distribution obtained when constructing a subset of T_i (of at least $\mathtt{thresh}$ many elements) by uniformly picking each element of T_i. Since the random variables $S_1, \ldots, S_k$ are clear from context, we will use $\mathsf{Pr}\left[\mathcal{E}; \mathsf{U}_{1,\ldots,k}\right]$ to denote the probability of event $\mathcal{E}$ in the resulting entanglement distribution. For well definedness, we require that $\mathsf{Pr}[S_i = \omega] > 0$ for every subset $\omega \subseteq T_i$ (with $\omega \geq \mathtt{thresh}$), and will ensure this each time we invoke this statement.

Theorem 1. *Let $\epsilon, \delta > 0$, and let Ω be some set. Let $T_1, T_2, \ldots, T_k \subseteq \Omega$ be sets with membership oracles $O_1, \ldots, O_k$ respectively. Let $\epsilon_{sz} \geq 0$ and let $sz_1, \ldots, sz_k \in \mathbb{N}$ be such that for every $i \leq k$, we have $\frac{|T_i|}{(1+\epsilon_{sz})} \leq sz_i \leq (1+\epsilon_{sz})\,|T_i|$. Let $\mathtt{thresh} = 24 \cdot \frac{(1+\epsilon_{sz})^2}{\epsilon^2} \cdot \log(\frac{4k}{\delta})$. For each $i \leq k$, let S_i be some sequence of samples (with possible repetitions), obtained according to some distribution such that $|S_i| \geq \mathtt{thresh}$ and for every $\omega \subseteq T_i$ with $|\omega| \geq \mathtt{thresh}$, $\mathsf{Pr}[S_i = \omega] > 0$. Then, on input $\epsilon_{sz}, (O_1, S_1, sz_1), \ldots, (O_k, S_k, sz_k)$, the algorithm $\mathtt{AppUnion}$ with parameters ϵ, δ (Algorithm 1) outputs sz that satisfies:*

$$\mathsf{Pr}\left[\frac{|\cup_{i=1}^{k} T_i|}{(1+\epsilon)(1+\epsilon_{sz})} \leq |sz| < (1+\epsilon)(1+\epsilon_{sz})|\cup_{i=1}^{k} T_i|\,;\mathsf{U}_{1,\ldots,k}\right] > 1-\delta.$$

The algorithm makes $O\left(k \cdot (1+\epsilon_{sz})^2 \cdot \frac{1}{\epsilon^2} \cdot \log\left(\frac{k}{\delta}\right)\right)$ number of membership calls to the oracles $O_1, \ldots, O_k$ in the worst case and the rest of the computation takes a similar amount of time.

3.2 Sampling Subroutine

The sampling subroutine (Algorithm 2) takes as input a number ℓ, a set of states P^ℓ (at level ℓ of the unrolled automaton $\mathcal{A}_{\mathsf{unroll}}$), a string w (of length $n-\ell$), a real number φ (representing a probability value), an error parameter β and a confidence parameter η, and outputs a string sampled from the

Algorithm 2: Sampling subroutine for $\mathcal{A}_{\mathsf{unroll}}$, length n

```
1  function sample(ℓ, P^ℓ, w, φ, β, η)
2      η' ← η/4n
3      β' ← (1+β)^{ℓ-1} − 1
4      if ℓ = 0 then
5          if φ > 1 then return ⊥
6          w.p. (φ) : return w  ||  w.p. (1−φ) : return ⊥
7      else
8          foreach b ∈ {0,1} do
9              P^ℓ_b ← ⋃_{p∈P^ℓ} Pred(p, b)
10             let P^ℓ_b be the set {p_{b,1}, ..., p_{b,k_b}}
11             sz_b ← AppUnion_{β,η'}( β', (S(p^{ℓ-1}_{b,1}), N(p^{ℓ-1}_{b,1})), ..., (S(p^{ℓ-1}_{b,k_b}), N(p^{ℓ-1}_{b,k_b})), A_unroll )
12         pr_0 ← sz_0 / (sz_0 + sz_1)
13         w.p. (pr_0) : b ← 0  ||  w.p. (1 − pr_0) : b ← 1
14         P^{ℓ-1} ← P^ℓ_b
15         w ← b · w
16     return sample(ℓ − 1, P^{ℓ-1}, w, φ/pr_b, β, η)
```

language $\cup_{q\in P^\ell}\mathcal{L}(q^\ell)$. Algorithm 2 is a recursive algorithm. In the base case ($\ell = 0$), it outputs the input string w with probability φ, and with the remaining probability, it outputs $\perp$. In the inductive case ($\ell > 0$), the algorithm computes estimates $\{\mathsf{sz}_b\}_{b\in\{0,1\}}$, where sz_b is the estimate of the size of $\cup_{p_{b,i}\in\mathsf{Pred}(q,b)}\mathcal{L}(p_{b,i}{}^{\ell-1})$. For this, it calls $\mathsf{AppUnion}_{(\epsilon,\delta)}$ with previously computed estimates of states at previous levels $\{\mathsf{N}(q^{\ell-1})\}_{q\in Q}$ and sampled subsets of strings $\{\mathsf{S}(q^{\ell-1})\}_{q\in Q}$, and uses the unrolled automaton $\mathcal{A}_{\mathsf{unroll}}$ as a proxy for membership oracles (see the last argument to $\mathsf{AppUnion}_{(\epsilon,\delta)}$ in Line 11). Next, b is chosen randomly to be 0 or 1 with probability proportional to the estimates sz_1 and sz_0 respectively (Line 13). Once b is chosen the set P^ℓ is updated to $P^{\ell-1}_b$ and the string w is updated to $b \cdot w$. Subsequently, the `sample()` subroutine is called recursively with the updated subset of states and the updated w.

We next tread towards formally characterizing the guarantee of Algorithm 2. To this end, we will introduce few notations:

- For each $1 \le i \le \ell$ and each $q \in Q$, let $\mathsf{AccurateN}_{i,q}$ be the event that $\frac{|\mathcal{L}(q^i)|}{(1+\beta)^i} \le \mathsf{N}(q^i) \le (1+\beta)^i|\mathcal{L}(q^i)|$. Also, let $\mathsf{AccurateN}_i = \bigcap_{q\in Q}\mathsf{AccurateN}_{i,q}$ and $\mathsf{AccurateN}_{\le i} = \bigcap_{1\le j\le i}\mathsf{AccurateN}_j$.
- For a state q^ℓ, we use $\mathsf{U}(q^\ell)$ to represent the random variable denoting the sequence of samples (with possible repetitions) constructed by repeatedly sampling $|\mathsf{S}(q^\ell)|$ number of elements uniformly from $\mathcal{L}(q^\ell)$. Let $\mathsf{S}_{\le i}$ be the random variable corresponding to the product of $\{\mathsf{S}(q^j)\}_{q\in Q,0\le j\le i}$. Also, let $\mathsf{U}_{\le i}$ represent the random variable obtained by taking the ordered profuct of all $\{\mathsf{U}(q^j)\}_{q\in Q,0\le j\le i}$. In our algorithms, we will use $\Pr[\mathcal{E};\mathsf{U}_{\le i}]$ (resp. $\Pr[\mathcal{E}|\mathcal{F};\mathsf{U}_{\le i}]$) to denote the probability (resp. conditional probability) of $\mathcal{E}$ (resp. $\mathcal{E}$ conditioned on $\mathcal{F}$) when $\mathsf{S}_{\le i}$ is distribution entangled to $\mathsf{U}_{\le i}$.
- Consider the random trial corresponding to the call to the procedure $\mathsf{sample}(\ell, \{q^\ell\}, \lambda, \gamma_0, \beta, \eta)$ for $\ell > 0$ and let $\mathsf{ret}_{\mathsf{sample}}$ denote the random variable representing the return value thus obtained. Observe that, in each such trial, the function AppUnion is called 2ℓ times, twice for each level $1 \le i \le \ell$. Let us now define some events of interest.

- For each $1 \le i \le \ell$ and $b \in \{\emptyset, 1\}$, by $\texttt{AccurateAppUnion}_{i,b}$ we denote the event that the b^{th} call to AppUnion corresponding to level i returns a value sz_b^i that satisfies $\frac{1}{(1+\beta)^i} \cdot |\mathcal{L}(P_b^i)| \le sz_b^i \le (1+\beta)^i \cdot |\mathcal{L}(P_b^i)|$. Here, P^i is the argument of the call at level i, and P_b^i is the b-predecessor of P^i. For ease of notation, we also define $\texttt{AccurateAppUnion} = \bigcap_{\substack{1 \le i \le \ell \\ b \in \{\emptyset,1\}}} \texttt{AccurateAppUnion}_{i,b}$
- Let $\texttt{Fail}_1$ be the event that $\texttt{ret}_{\texttt{sample}} = \perp$ is returned because $\varphi > 1$ at the time of return (Line 5). Let $\texttt{Fail}_2$ be the event that $\texttt{ret}_{\texttt{sample}} = \perp$ is returned at Line 6. Finally, let $\texttt{Fail} = \texttt{Fail}_1 \uplus \texttt{Fail}_2$.

The following presents the correctness of Algorithm 2, while its proof is presented in Section 4.2.

Theorem 2. *Let* $\gamma_0 = \frac{2}{3e} \cdot \frac{1}{\mathsf{N}(q^\ell)}, \beta \le \frac{1}{4 \cdot n^2}$ *and* $\{\mathsf{S}(p^i)\}_{1 \le i < \ell, p \in Q}$ *satisfy*

$$\forall\, 1 \le i \le \ell, \forall p \in Q, \quad |\mathsf{S}(p^i)| \ge \frac{24 \cdot e}{\beta^2} \log\left(\frac{16mn}{\eta}\right).$$

Then for the trial corresponding to the invocation $\texttt{sample}(\ell, \{q^\ell\}, \lambda, \gamma_0, \beta, \eta)$ *the following hold*

(1) For each $w \in \mathcal{L}(q^\ell)$, $\Pr\left[\texttt{ret}_{\texttt{sample}} = w \;\middle|\; \texttt{AccurateAppUnion} \cap \texttt{AccurateN}_{\ell,q}\right] = \gamma_0$

(2) $\Pr\left[\texttt{Fail} \mid \texttt{AccurateAppUnion} \cap \texttt{AccurateN}_{\ell,q}\right] \le 1 - \frac{2}{3e^2}$

(3) $\Pr[\texttt{AccurateAppUnion} \mid \texttt{AccurateN}_{\le \ell-1}; \mathsf{U}_{\le \ell-1}] \ge 1 - \frac{\eta}{2}$

3.3 Main Algorithm

Algorithm 3 first constructs (in Line 4) the labeled directed acyclic graph $\mathcal{A}_{\text{unroll}}$. The algorithm then goes over all the states and all levels $0 \le \ell \le n$ to construct the sampled sets and the size estimates inductively. In Lines 6-10, the algorithm caters for the base case ($\ell = 0$). In the inductive case (Lines 12-15) the algorithm computes the estimate sz_1 and $sz_\emptyset$, where sz_b is the estimate of the size of $\cup_{p_{b,i} \in \mathsf{Pred}(q,b)} \mathcal{L}(p_{b,i}{}^{\ell-1})$. The size estimates of, and a sample subsets of the sets $\{\mathcal{L}(q^{\ell-1})\}_{q \in Q}$ are available inductively at this point. Further, membership in these languages can be easily checked. As a result, the algorithm uses the subroutine AppUnion to compute the size of the union (in Line 15), with $\mathcal{A}_{\text{unroll}}$ as the membership oracle as before. Once the size estimates $sz_\emptyset$ and sz_1 are determined, the algorithm constructs the sampled subset of $\mathcal{L}(q^\ell)$ by making xns calls to the subroutine sample (Lines 21-25); if fewer than ns many strings were sampled, the algorithm simply pads one fixed string from $\mathcal{L}(q^\ell)$ so that $\mathsf{S}(q^\ell)$ eventually has size ns (see Lines 27-30). Theorem 3 presents the desired correctness statement of our FPRAS Algorithm 3, and its proof is presented in Section 4.3.

Theorem 3. *Given an NFA* $\mathcal{A}$ *with* m *states and* $n \in \mathbb{N}$ *(presented in unary), Algorithm 3 returns* Est *such that the following holds:*

$$\Pr\left[\frac{|\mathcal{L}(\mathcal{A}_n)|}{1+\epsilon} \le \mathsf{Est} \le (1+\epsilon)|\mathcal{L}(\mathcal{A}_n)|\right] \ge 1 - \delta$$

Moreover, the algorithm has time complexity $\widetilde{O}((m^2n^{10} + m^3n^6) \cdot \frac{1}{\epsilon^4} \cdot \log^2(1/\delta))$, *where the tilde hides for polynomial factors of* $\log(m+n)$.

4 TECHNICAL ANALYSIS

We now present the detailed technical analyses of the algorithms presented in Section 3.

4.1 Correctness of Algorithm 1

The proof is in the same lines as that of Karp and Luby [12]. For the sake of completeness, we present a full proof.

Algorithm 3: Algorithm for estimating $|\mathcal{L}(\mathcal{A}_n)|$

Input : NFA $\mathcal{A}$ with m states, $n \in \mathbb{N}$ in unary
Parameters: Accuracy parameter ε and confidence parameter δ

1 $\beta \leftarrow \frac{\varepsilon}{4n^2}; \eta \leftarrow \frac{\delta}{2 \cdot n \cdot m}$
2 $\mathsf{ns} \leftarrow \frac{4096 \cdot e \cdot n^4}{\epsilon^2} \log\left(\frac{4096 \cdot m^2 \cdot n^2 \cdot \log(\varepsilon^{-2})}{\delta}\right)$
3 $\mathsf{xns} \leftarrow \mathsf{ns} \cdot 12 \cdot (1 - \frac{2}{3e^2})^{-1} \cdot \log(8/\eta)$
4 Construct the labeled directed acyclic graph $\mathcal{A}_{\mathsf{unroll}}$ from $\mathcal{A}$
5 **for** $\ell \in \{0, \ldots, n\}$ *and* $q \in Q$ **do**
6 **if** $\ell = 0$ **then**
7 **if** $q = I$ **then**
8 $\big(\mathsf{N}(q^\ell), \mathsf{S}(q^\ell)\big) \leftarrow (1, \lambda)$
9 **else**
10 $\big(\mathsf{N}(q^\ell), \mathsf{S}(q^\ell)\big) \leftarrow (0, \varnothing)$
11 **else**
12 **foreach** $\mathsf{b} \in \{\mathtt{0}, \mathtt{1}\}$ **do**
13 **let** $\{p_{\mathsf{b},1}, \ldots, p_{\mathsf{b},k_\mathsf{b}}\}$ be the set $\mathsf{Pred}(q, \mathsf{b})$
14 $\beta' \leftarrow (1+\beta)^{\ell-1} - 1$
15 $\mathsf{sz}_\mathsf{b} \leftarrow$ $\mathsf{AppUnion}_{\beta, \frac{\eta}{2} \cdot (1 - \frac{1}{2^{n+1}})}\Big(\beta', (\mathsf{S}(p_{\mathsf{b},1}{}^{\ell-1}), \mathsf{N}(p_{\mathsf{b},1}{}^{\ell-1})), \ldots, (\mathsf{S}(p_{\mathsf{b},k_\mathsf{b}}{}^{\ell-1}), \mathsf{N}(p_{\mathsf{b},k_\mathsf{b}}{}^{\ell-1})), \mathcal{A}_{\mathsf{unroll}}\Big)$
16 **w.p.** $1 - \frac{\eta}{2^n}$:
17 $\mathsf{N}(q^\ell) \leftarrow \mathsf{sz}_\mathtt{0} + \mathsf{sz}_\mathtt{1}$
18 **w.p.** $\frac{\eta}{2^n}$:
19 $\mathsf{N}(q^\ell) \xleftarrow{\text{Unif-smp}} \{0, 1, \ldots, 2^\ell\}$
20 $\mathsf{S}(q^\ell) \leftarrow \varnothing$
21 **repeat** xns **times:**
22 **if** $|\mathsf{S}(q^\ell)| < \mathsf{ns}$ **then**
23 $w \leftarrow \mathsf{sample}(\ell, \{q^\ell\}, \lambda, \frac{2}{3e} \cdot \frac{1}{\mathsf{N}(q^\ell)}, \beta, \frac{\eta}{2 \cdot \mathsf{xns}})$
24 **if** $w \neq \perp$ **then**
25 $\mathsf{S}(q^\ell) \leftarrow \mathsf{S}(q^\ell) \cdot \mathsf{append}(w)$
26 **else break**
27 **let** $\alpha_{q^\ell} = \mathsf{ns} - |\mathsf{S}(q^\ell)|$
28 **if** $\alpha_{q^\ell} > 0$ **then**
29 **let** w_{q^ℓ} be some word in $\mathcal{L}(q^\ell)$
30 **repeat** α_{q^ℓ} **times:** $\mathsf{S}(q^\ell) \leftarrow \mathsf{S}(q^\ell) \cdot \mathsf{append}(w_{q^\ell})$
31 **return** $\mathsf{N}(q_F^n)$

Proof (of Theorem 1). Let Fail be the event that the output of the algorithm is not within $\frac{|\cup_{i=1}^k T_i|}{(1+\epsilon)(1+\epsilon_{sz})}$ and $|\cup_{i=1}^k T_i|(1+\epsilon)(1+\epsilon_{sz})$. We will show that $\Pr\left[\mathsf{Fail}; \mathsf{U}_{1,\ldots,k}\right]$ is upper-bounded by δ.

Note that the algorithm in each of the t runs of the loop (Lines 5-9) tries to draw an element from some S_i. So if we assume that the size of the sets S_i for all i is more than t, then the condition in the **if** (in Line 7) will be satisfied and in that case the **else** clause in Line 8 is redundant.

We will prove the correctness of Algorithm 1 in two parts. In the first part we will prove that if the size of the sets S_i is greater than t then $\Pr\left[\mathsf{Fail}; \mathsf{U}_{1,\ldots,k}\right] \leq \frac{\delta}{2}$. In the second part we will prove that if the sets S_i has size $\mathtt{thresh}$ (much smaller than t) then the probability that the algorithm will ever need the **else** clause in Line 8 is less that $\delta/2$. The two parts combined proves that if the sets S_is are of size $\mathtt{thresh}$, then $\Pr\left[\mathsf{Fail}; \mathsf{U}_{1,\ldots,k}\right] \leq \delta$ which is what is claimed in the theorem.

For all $\sigma \in \cup_{i=1}^{k} T_i$ let $L(\sigma)$ denote the smallest index i such that $\sigma \in T_i$. In other words, σ is in $T_{L(\sigma)}$ and for all $i < L(\sigma)$, $\sigma \notin T_i$. Observe $\mathcal{U}_{\mathsf{unique}} := \left\{(\sigma, L(\sigma)) \mid \sigma \in \cup_{i=1}^{k} T_i\right\}$ has size exactly $\left|\cup_{i=1}^{k} T_i\right|$. Furthermore, $\mathcal{U}_{\mathsf{multiple}} := \{(\sigma, i) \mid \sigma \in T_i\}$ has size $\sum_{i=1}^{k} |T_i|$.

Part 1: If $\forall i$, $|S_i| \geq t$ then $\Pr\left[\mathsf{Fail}; \mathsf{U}_{1,\ldots,k}\right] \leq \frac{\delta}{2}$. If $|S_i| \leq t$ for all i, then the condition in the **if** clause in Line 7 is always satisfied. For a run of the loop (Lines 5-9), consider the Lines 6-7. We say the pair (σ, i) is sampled in round j if in the j^{th} iteration of the loop i is sampled in Line 6 and then in that same iteration σ is obtained from S_i.dequeue() in the Line 7.

For any $i_0 \in \{1, \ldots, k\}$, $\sigma_0 \in S_{i_0}$ and $j_0 \in \{1, \ldots, t\}$, we have

$$\Pr\left[(\sigma_0, i_0) \text{ is sampled in round } j_0; \mathsf{U}_{1,\ldots,k}\right] = \begin{array}{r}\Pr\left[i_0 \text{ is sampled in Line 6 in round } j_0; \mathsf{U}_{1,\ldots,k}\right] \\ \times \Pr\left[\begin{array}{l}\sigma_0 \text{ is obtained in Line 7 in round} \\ j_0 \mid i_0 \text{ is sampled in Line 6}; \mathsf{U}_{1,\ldots,k}\end{array}\right]\end{array} \tag{1}$$

Since the choice of the set to pick i_0 is independent of the distributions of any of the S_i's, we have,

$$\Pr\left[i_0 \text{ is sampled in Line 6}; \mathsf{U}_{1,\ldots,k}\right] = \Pr[i_0 \text{ is sampled in Line 6}] = \frac{\mathsf{sz}_{i_0}}{\sum_i^k \mathsf{sz}_i}.$$

Next observe that the second term in Equation (1) is $\Pr\left[\sigma_0 \text{ is sampled in round } j_0 \text{ from } S_{i_0}; \mathsf{U}_{1,\ldots,k}\right]$. This term, in turn, can be computed by considering the disjoint union of the events corresponding to how many times i_0 has been chosen so far: $\Pr\left[\sigma_0 \text{ is sampled in round } j_0 \text{ from } S_{i_0}; \mathsf{U}_{1,\ldots,k}\right]$

$$= \sum_{m=1}^{t} \Pr\left[\begin{array}{r}i_0 \text{ was sampled } m-1 \text{ times before round } j_0 \\ \text{and } \sigma_0 \text{ is sampled in round } j_0 \text{ from } S_{i_0}\end{array}; \mathsf{U}_{1,\ldots,k}\right] = \sum_{m=1}^{t} \Pr\left[S_{i_0}[m] = \sigma_0; \mathsf{U}_{1,\ldots,k}\right]$$

$$= \sum_{\omega} \sum_{m=1}^{t} \Pr\left[\omega[m] = \sigma_0 \mid S_{i_0} = \omega\right] \cdot \Pr\left[U_{i_0} = \omega\right] = \sum_{\omega} \Pr\left[U_{i_0} = \omega \text{ and } \omega[m] = \sigma_0\right] = \frac{1}{|T_{i_0}|}$$

Therefore, $\frac{1}{(1+\epsilon_{\mathsf{sz}})\sum_{i=1}^{k} \mathsf{sz}_i} \leq \Pr\left[(\sigma_0, i_0) \text{ is sampled in round } j_0; \mathsf{U}_{1,\ldots,k}\right] \leq (1+\epsilon_{\mathsf{sz}})\frac{1}{\sum_{i=1}^{k} \mathsf{sz}_i}$.

For any $i_0 \in [t]$ and $\sigma_0 \in S_{i_0}$, we have $\Pr\left[(\sigma_0, i_0) \text{ is sampled }; \mathsf{U}_{1,\ldots,k}\right]$

$$= \Pr\left[i_0 \text{ is sampled in Line 6}; \mathsf{U}_{1,\ldots,k}\right] \times \Pr\left[\sigma \text{ is obtained in Line 7} \mid i_0 \text{ is sampled in Line 6}; \mathsf{U}_{1,\ldots,k}\right]$$

$$= \frac{\mathsf{sz}_{i_0}}{\sum_i^k \mathsf{sz}_i} \times \frac{1}{|T_{i_0}|}$$

Probability that i_0 is sampled in Line 6 is independent of the distributions of the samples. The last equality is because we have assumed $\forall i$, $|S_i| \geq t$, so the **if** condition in Line 7 is always satisfied. Hence once i_0 is sampled in Line 7 probability that σ_0 is obtained is exactly $1/|T_{i_0}|$, since the set S_{i_0} contains elements sampled uniformly (with replacement) from T_{i_0}. Therefore, $\frac{1}{(1+\epsilon_{\mathsf{sz}})\sum_{i=1}^{k} \mathsf{sz}_i} \leq \Pr\left[\sigma_0, i_0) \text{ is sampled }; \mathsf{U}_{1,\ldots,k}\right] \leq (1+\epsilon_{\mathsf{sz}})\frac{1}{\sum_{i=1}^{k} \mathsf{sz}_i}$.

Let Y_j be the random variable denoting whether the counter Y is increased in the jth iteration of the loop. Note that $Y_j = 1$ if the pair (σ_0, i_0) sampled in the jth iteration is in $\mathcal{U}_{\mathsf{unique}}$. This is because in Line 9 it is checked if the sampled pair is in $\mathcal{U}_{\mathsf{unique}}$.

So, $\mathbb{E}[Y_j; \mathsf{U}_{1,\ldots,k}] = \sum_{(\sigma,i)\in\mathcal{U}_{\mathsf{unique}}} \Pr\big[(\sigma,i) \text{ is sampled in round } j; \mathsf{U}_{1,\ldots,k}\big]$; here the ';' in the expectation has an analogous definition to the case when used for probability. Thus,

$$\frac{|\mathcal{U}_{\mathsf{unique}}|}{(1+\epsilon_{\mathsf{sz}})\sum_{i=1}^{k}\mathsf{sz}_i} \le \mathbb{E}[Y_j; \mathsf{U}_{1,\ldots,k}] \le (1+\epsilon_{\mathsf{sz}})\frac{|\mathcal{U}_{\mathsf{unique}}|}{\sum_{i=1}^{k}\mathsf{sz}_i}$$

Note that $Y = \sum_i^t Y_i$. Thus at the end of the algorithm the expected value of Y is between $\frac{t|\mathcal{U}_{\mathsf{unique}}|}{(1+\epsilon_{\mathsf{sz}})\sum_{i=1}^{k}\mathsf{sz}_i}$ and $(1+\epsilon_{\mathsf{sz}})^2\frac{t|\mathcal{U}_{\mathsf{unique}}|}{\sum_{i=1}^{k}\mathsf{sz}_i}$. Now by Chernoff bound we have,

$$\Pr\Big[Y \notin (1\pm\frac{\epsilon}{2})\mathbb{E}[Y_j; \mathsf{U}_{1,\ldots,k}]; \mathsf{U}_{1,\ldots,k}\Big] \le 2exp(-\frac{\epsilon^2\mathbb{E}[Y_j; \mathsf{U}_{1,\ldots,k}]}{12}).$$

We also know that,

$$\mathbb{E}[Y; \mathsf{U}_{1,\ldots,k}] \ge \frac{t|\mathcal{U}_{\mathsf{unique}}|}{(1+\epsilon_{\mathsf{sz}})\sum_{i=1}^{k}\mathsf{sz}_i} \ge \frac{t\max_i|T_i|}{(1+\epsilon_{\mathsf{sz}})\sum_{i=1}^{k}\mathsf{sz}_i} \ge \frac{t\max_i \mathsf{sz}_i}{(1+\epsilon_{\mathsf{sz}})^2\sum_{i=1}^{k}\mathsf{sz}_i} \ge 12\log\left(\frac{4}{\delta}\right)$$

The last inequality is due to the setting of the parameter t. Thus, by Chernoff bound, probability that the output of the algorithm is not between $(1-\frac{\epsilon}{2})\frac{|\mathcal{U}_{\mathsf{unique}}|}{(1+\epsilon_{\mathsf{sz}})}$ and $(1+\frac{\epsilon}{2})(1+\epsilon_{\mathsf{sz}})|\mathcal{U}_{\mathsf{unique}}|$ is at most $\delta/2$. Since $(1-\frac{\epsilon}{2}) \ge \frac{1}{1+\epsilon}$, so we have $\Pr\big[\mathsf{Fail}; \mathsf{U}_{1,\ldots,k}\big] \le \frac{\delta}{2}$.

Part 2: If for all i, $|S_i| \ge$ `thresh` then $\Pr\big[$ Line 8 is ever reached $; \mathsf{U}_{1,\ldots,k}\big] \le \frac{\delta}{2}$.

We first observe that since the probability of reaching Line 8 is independent of the distributions of the sample, we have that $\Pr\big[$ Line 8 is ever reached $; \mathsf{U}_{1,\ldots,k}\big] = \Pr[$ Line 8 is ever reached $]$, and thus we will try to upper bound the latter quantity instead. Note that, the **else** clause in Line 8 is ever reached, if for some i, the number of times i is sampled in the t iterations is more than `thresh`.

Let Bad_i denote the event that the index i is sampled (in Line 6) more that `thresh` number of times during the course of the algorithm, that is during the t runs of the loop (Lines 5-9). And let the event Bad be $\cup_{i=1}^{k}\mathsf{Bad}_i$. We will now upper bound, $\Pr[\mathsf{Bad}_i]$. Let W_i be the random variable counting the number times i is sampled: $\mathbb{E}[W_i] = t \times \frac{\mathsf{sz}_i}{\sum_{i=1}^{k}\mathsf{sz}_i}$ which is less than $\frac{12(1+\epsilon_{\mathsf{sz}})^2}{\epsilon^2}\log(4/\delta)$. By simple application of Chernoff Bound we see that

$$\Pr[W_i \ge (1+\Delta)\mathbb{E}[W_i]] \le exp\left(-\frac{\Delta^2\mathbb{E}[W_i]}{2+\Delta}\right) \le exp\left(-\frac{\Delta\mathbb{E}[W_i]}{2}\right),$$

the last inequality is for $\Delta \ge 2$. For our purpose if we put $\Delta = \log(\frac{2k}{\delta})\frac{1}{\mathbb{E}[W_i]}$, since $|S_i| \ge \mathtt{thresh} \ge \frac{12\cdot(1+\epsilon_{\mathsf{sz}})^2}{\varepsilon^2}\cdot\log(\frac{4}{\delta}) + \log(\frac{2k}{\delta})$, we have $\Pr[\mathsf{Bad_i}] \le \frac{\delta}{2k}$. So by union bound $\Pr[\mathsf{Bad}] \le \delta/2$. Observe that for large k, $\Delta \ge 2$ as desired. $\Pr[\mathsf{Bad_i}] \le \frac{\delta}{2k}$.Observe that for large k, $\Delta \ge 2$ as desired. So by union bound, we have: $\Pr\big[\mathsf{Bad}; \mathsf{U}_{1,\ldots,k}\big] = \Pr[\mathsf{Bad}] \le \delta/2$. □

4.2 Correctness of Algorithm 2

We prove the three parts of Theorem 2 individually.

Proof of Theorem 2(1)

Consider an execution of Algorithm 2 and let $w = w_1w_2\ldots w_\ell \in \mathcal{L}(q^\ell)$ be the string constructed right before entering the branch at Line 5. Let $p_{\emptyset,i}$ be the value of the variable $\mathsf{pr}_\emptyset$ at level i and let v be the value of φ before the function returns. Then we have,

$$v = \frac{\gamma_0}{p_{w_\ell,\ell}\cdot p_{w_{\ell-1},\ell-1}\cdots p_{w_1,1}}$$

Let $\mathsf{sz}^{\ell}_{\mathtt{0}}, \mathsf{sz}^{\ell}_{\mathtt{1}}, \mathsf{sz}^{\ell-1}_{\mathtt{0}}, \mathsf{sz}^{\ell-1}_{\mathtt{1}}, \ldots \mathsf{sz}^{1}_{\mathtt{0}}, \mathsf{sz}^{1}_{\mathtt{1}}$ be the estimates obtained in the 2ℓ calls to AppUnion during the run of sample(). Then,

$$p_{w_i,i} = \frac{\mathsf{sz}^i_{w_i}}{\mathsf{sz}^i_{w_i} + \mathsf{sz}^i_{1-w_i}}$$

Thus, $v = \frac{\gamma_0}{\prod_{i=1}^{\ell} \frac{\mathsf{sz}^i_{w_i}}{\mathsf{sz}^i_{w_i}+\mathsf{sz}^i_{1-w_i}}} = \gamma_0 \cdot \prod_{i=1}^{\ell} \frac{\mathsf{sz}^i_{w_i}+\mathsf{sz}^i_{1-w_i}}{\mathsf{sz}^i_{w_i}}$

Recall that P^i is the argument of the call at level i, and $P^i_{\mathtt{b}}$ is the $\mathtt{b}$-predecessor of P^i. Observe that $\mathcal{L}(P^i) = \mathcal{L}(P^i_{\mathtt{0}}) \uplus \mathcal{L}(P^i_{\mathtt{1}})$, and thus, $|\mathcal{L}(P^i_{\mathtt{0}})| + |\mathcal{L}(P^i_{\mathtt{1}})| = |\mathcal{L}(P^i)|$

Hence, under the event $\mathtt{AccurateAppUnion}_{i,\mathtt{0}} \cap \mathtt{AccurateAppUnion}_{i,\mathtt{1}}$ we have

$$\frac{\mathsf{sz}^i_{w_i} + \mathsf{sz}^i_{1-w_i}}{\mathsf{sz}^i_{w_i}} \leq (1+\beta)^{2i} \cdot \frac{|\mathcal{L}(P^i)|}{|\mathcal{L}(P^{i-1})|} \tag{2}$$

This gives us,

$$v \leq \gamma_0 \cdot \prod_{i=1}^{\ell} (1+\beta)^{2i} \cdot \tfrac{|\mathcal{L}(P^i)|}{|\mathcal{L}(P^{i-1})|} = \gamma_0 \cdot \left(\prod_{i=1}^{\ell} (1+\beta)^{2i}\right) \cdot \left(\prod_{i=1}^{\ell} \tfrac{|\mathcal{L}(P^i)|}{|\mathcal{L}(P^{i-1})|}\right)$$

Now, observe that $|\mathcal{L}(P^0)| = 1$ and $P^\ell = \{q^\ell\}$.

Thus, under the event $\left(\bigcap_{\substack{1\leq i\leq \ell \\ \mathtt{b}\in\{\mathtt{0},\mathtt{1}\}}} \mathtt{AccurateAppUnion}_{i,\mathtt{b}}\right) = \mathtt{AccurateAppUnion}$, we have

$$v \leq \gamma_0 \cdot \left(\prod_{i=1}^{\ell} (1+\beta)^{2i}\right) \cdot |\mathcal{L}(q^\ell)| = \gamma_0 \cdot (1+\beta)^{\ell(\ell+1)} \cdot |\mathcal{L}(q^\ell)| \tag{3}$$

Assuming the event $\mathtt{AccurateN}_{\ell,q}$ also holds, we have $\frac{|\mathcal{L}(q^\ell)|}{\mathsf{N}(q^\ell)} \leq (1+\beta)^\ell$. Under this assumption, substituting $\gamma_0 = \frac{2}{3e} \cdot \frac{1}{\mathsf{N}(q^\ell)}$, $\beta \leq \frac{1}{4\cdot n^2}$, and using the inequality $(1+1/4n^2)^n \leq e$, we have,

$$v \leq \tfrac{2}{3e} \cdot (1+\beta)^{\ell(\ell+2)} \leq \tfrac{2}{3e} \cdot e \leq 1$$

It then follows that $\Pr\left[\mathtt{Fail}_1 \mid \mathtt{AccurateAppUnion} \cap \mathtt{AccurateN}_{\ell,q}\right]$ is equal to 0.. Substituting, $v = \frac{\gamma_0}{p_{w_\ell,\ell} \cdot p_{w_{\ell-1},\ell-1} \cdots p_{w_1,1}} = \frac{\gamma_0}{\prod_{i=\ell}^{1} p_{\mathtt{0},i}}$, we then have

$$\Pr\left[\mathtt{ret}_{\mathtt{sample}} = w \mid \mathtt{AccurateAppUnion} \cap \mathtt{AccurateN}_{\ell,q}\right] = v \cdot \prod_{i=\ell}^{1} p_{\mathtt{0},i} = \frac{\gamma_0}{\prod_{i=\ell}^{1} p_{\mathtt{0},i}} \cdot \prod_{i=\ell}^{1} p_{\mathtt{0},i} = \gamma_0$$

Proof of Theorem 2(2)

We will now estimate the (conditional) probability of the event Fail. First, observe that for each string $w' \notin \mathcal{L}(q^\ell)$, we have $\Pr\left[\mathtt{ret}_{\mathtt{sample}} = w'\right] = 0$ because any string w that the algorithm outputs is such that there is a path from the initial state to q^ℓ labeled with w, and (b) the unrolled automata is such that any path from start state to some state p^i corresponds to a string in $\mathcal{L}(p^i)$.

We observe that $\overline{\texttt{Fail}}$ is the event that $\texttt{ret}_{\texttt{sample}}$ is a string in $\mathcal{L}(q^\ell)$, each of which is a disjoint event. Thus,

$$\begin{aligned}
&\Pr\left[\texttt{Fail} \mid \texttt{AccurateAppUnion} \cap \texttt{AccurateN}_{\ell,q}\right] \\
&= 1 - \sum_{w \in \mathcal{L}(q^\ell)} \Pr\left[\texttt{ret}_{\texttt{sample}} = w \mid \texttt{AccurateAppUnion} \cap \texttt{AccurateN}_{\ell,q}\right] \\
&= 1 - \sum_{w \in \mathcal{L}(q^\ell)} \gamma_0 = 1 - \gamma_0 |\mathcal{L}(q^\ell)| \leq 1 - \frac{2}{3e} \cdot \frac{|\mathcal{L}(q^\ell)|}{\mathsf{N}(q^\ell)} \leq 1 - \frac{2}{3e} \cdot \frac{1}{(1+\beta)^\ell} \leq 1 - \frac{2}{3e^2}
\end{aligned}$$

Proof of Theorem 2(3)

We would like to use Theorem 1 to prove Theorem 2(3). Towards this, we first establish that the pre-conditions of Theorem 1 hold.

(1) For $i \leq \ell$, if we substitute $\beta' = (1+\beta)^{i-1} - 1, \eta' = \eta/4n$, we get $\texttt{thresh} = 24 \cdot \frac{(1+\beta)^{2i-2}}{\beta^2} \cdot \log\left(\frac{16 k_{\mathsf{b}} n}{\eta}\right)$, where k_{b} is the number of sets in the argument of AppUnion. Observe that, $k_{\mathsf{b}} \leq m$ and that, for each $i \leq \ell$, we have $(1+\beta)^{2i-2} \leq (1+\beta)^{4n^2} \leq e$. Thus for each q^i, we have

$$\texttt{thresh} \leq \frac{24 \cdot e}{\beta^2} \log\left(\frac{16mn}{\eta}\right) \leq |\mathsf{S}(q^{i-1})|.$$

(2) Since we are considering the distribution obtained when $\mathsf{S}(q^i)$ is replaced with $\mathsf{U}(q^i)$ for each $i \leq \ell - 1$ (observe the $\Pr[;\mathsf{U}_{\leq \ell-1}]$ in the statement of Theorem 2(3)), we directly have that each input $\{S_i\}_i$ to every call to AppUnion has at least thresh elements (as established) elements obtained by repeatedly sampling $\{T_i\}_i$ uniformly.

(3) The event $\texttt{AccurateN}_{\leq \ell-1}$ demands that for each $i \leq \ell - 1$, we have

$$\frac{|\mathcal{L}(q^i)|}{(1+\beta)^i} \leq \mathsf{N}(q^i) \leq (1+\beta)^i |\mathcal{L}(q^i)|.$$

From Theorem 1, it thus follows that, for all i,

$$\Pr\left[\texttt{AccurateAppUnion}_{i,\mathsf{b}} \;\middle|\; \texttt{AccurateN}_{\leq \ell-1}; \mathsf{U}_{\leq \ell-1}\right] \geq 1 - \eta'$$

Therefore, applying union-bound, we have

$$\Pr\left[\bigcap_{\substack{1 \leq i \leq \ell \\ \mathsf{b} \in \{0,1\}}} \texttt{AccurateAppUnion}_{i,\mathsf{b}} \;\middle|\; \texttt{AccurateN}_{\leq \ell-1}; \mathsf{U}_{\leq \ell-1}\right] \geq 1 - 2n \cdot \eta' = 1 - \frac{\eta}{2}$$

4.3 Correctness of Algorithm 3

We prove the main result (Theorem 3) by induction on the level ℓ. For each level ℓ, we first characterize the accuracy of the computed estimates $\{\mathsf{N}(q^\ell)\}_{q \in Q}$ and the quality of the sampled sets $\{\mathsf{S}(q^\ell)\}_{q \in Q}$, assuming that these were computed when the samples at the lower levels $1, 2, \ldots, \ell - 1$ are perfectly uniform and have size thresh. After this, we characterize the real computed estimates and samples by arguing that the distance of the corresponding random variables, from those computed using perfectly uniform samples, is small. We first establish some helpful auxiliary results (Lemma 4 and Lemma 5) towards our main proof. As before, we will often invoke distribution entanglement of $\mathsf{S}_{\leq \ell-1}$ by $\mathsf{U}_{\leq \ell-1}$ for each level ℓ.

Lemma 4. *For each $\ell \le n$ and $q \in Q$, we have*

$$\Pr\left[\mathtt{AccurateN}_{q,\ell} \;\middle|\; \mathtt{AccurateN}_{\le \ell-1}; \mathsf{U}_{\le \ell-1}\right] \ge 1-\eta$$

Proof. Let us denote by $\mathcal{L}(\mathsf{Pred}(q^\ell, \mathsf{b}))$ the set $\bigcup_{p \in \mathsf{Pred}(q^\ell, \mathsf{b})} \mathcal{L}(p)$. Observe that,

$$\mathcal{L}(q^\ell) = \mathcal{L}(\mathsf{Pred}(q^\ell, \mathtt{0})) \cdot \{\mathtt{0}\} \uplus \mathcal{L}(\mathsf{Pred}(q^\ell, \mathtt{1})) \cdot \{\mathtt{1}\}.$$

Now, we would like to apply Theorem 1 to show that sz_b (at level ℓ) is a *good* estimate. To this end, we first show that the pre-conditions stated in Theorem 1 hold. This is because, (1) under the event $\mathtt{AccurateN}_{\le \ell-1}$, the estimates $\{\mathsf{N}(p^{\ell-1})\}_{p \in Q}$ satisfy desired requirements. Next, (2) for each $i \le \ell - 1$, the samples $\{\mathsf{S}(p^{\ell-1})\}_{p \in Q}$ also obey the desired requirements. This is because $\mathtt{thresh}$ (in Algorithm 1), after substituting β, η, satisfies

$$\mathtt{thresh} = 24 \cdot \frac{(1+\beta)^{2\ell-2}}{\beta^2} \cdot \log\left(\frac{4k_\mathsf{b}}{\eta'}\right) \le \frac{24 \cdot e}{\beta^2} \log\left(\frac{8m^2 n}{\delta}\right) \le \frac{384 \cdot e \cdot n^4}{\epsilon^2} \log\left(\frac{8m^2}{\delta}\right) \le \mathsf{ns}$$

Therefore, from Theorem 1, we have

$$\Pr\left[\frac{|L(\mathsf{Pred}(q^\ell, \mathsf{b}))|}{(1+\beta)^\ell} \le \mathsf{sz}_\mathsf{b} \le (1+\beta)^\ell \; |L(\mathsf{Pred}(q^\ell, \mathsf{b}))| \;\middle|\; \mathtt{AccurateN}_{\le \ell-1}; \mathsf{U}_{\le \ell-1}\right]$$

is at least $(1 - \frac{\eta}{2})$. Since $\mathsf{N}(q^\ell) = \mathsf{sz}_\mathtt{0} + \mathsf{sz}_\mathtt{1}$, we have

$$\Pr\left[\frac{|\mathcal{L}(q^\ell)|}{(1+\beta)^\ell} \le \mathsf{N}(q^\ell) \le (1+\beta)^\ell |\mathcal{L}(q^\ell)| \;\middle|\; \mathtt{AccurateN}_{\le \ell-1}; \mathsf{U}_{\le \ell-1}\right] \ge 1-\eta \qquad \square$$

In addition to the previously introduced notation $\Pr[\,; \mathsf{U}_{\le i}]$, we will also use the notation $\mathsf{TV}(X, Y \;; \mathsf{U}_{\le i})$ (resp. $\mathsf{TV}(X, Y \,|\, \mathcal{F} \;; \mathsf{U}_{\le i})$) to denote the total variational distance between the distribution induced by random variables X, Y (resp. distribution induced by X, Y conditioned on the event $\mathcal{F}$) when $\mathsf{S}_{\le i}$ is distribution entangled to $\mathsf{U}_{\le i}$. Formally,

$$\mathsf{TV}(X, Y \mid \mathcal{F} \;; \mathsf{U}_{\le i}) = \sum_{\omega'} \Pr[\mathsf{U}_{\le i} = \omega'] \left(\sum_{\omega} \left(\Pr[X = \omega \mid \mathcal{F} \cap (\mathsf{S}_{\le i} = \omega')] - \min\left(\Pr[X = \omega \mid \mathcal{F} \cap (\mathsf{S}_{\le i} = \omega')], \Pr[Y = \omega \mid \mathcal{F} \cap (\mathsf{S}_{\le i} = \omega')]\right) \right) \right)$$

In the above expression, ω ranges over the union of domains of X and Y while ω' ranges over the union of domains of $\mathsf{S}_{\le i}$ and $\mathsf{U}_{\le i}$.

Lemma 5. *For each $\ell \le n$ and $q \in Q$, we have*

$$\mathsf{TV}(\mathsf{S}(q^\ell), \mathsf{U}(q^\ell)) \;\middle|\; \mathtt{AccurateN}_{\le \ell} \;; \mathsf{U}_{\le \ell-1}) \le \eta$$

Proof. Fix ℓ and q. Let $\mathtt{Fail}^j$ denote the event that the j^{th} call to $\mathtt{sample}$ (in the loop starting at Line 21) for $i = j$ returns $\perp$. Similarly, let $\mathtt{AccurateAppUnion}^j$ denote the event $\mathtt{AccurateAppUnion}$ corresponding to the call to $\mathtt{sample}$ for $i = j$. Using Iverson bracket notation [9], we denote the indicator variable $[\mathtt{Fail}^j]$. Also, we use SmallS to denote the event that $\alpha_{q^\ell} > 0$. Observe that SmallS occurs if and only if $\sum_{j=1}^{\mathsf{xns}} [\mathtt{Fail}^j] \ge \mathsf{xns} - \mathsf{ns}$.

Recall, we have, for each j, $\Pr\left[[\mathtt{Fail}^j] \;\middle|\; \mathtt{AccurateAppUnion}^j \cap \mathtt{AccurateN}_{\le \ell}; \mathsf{U}_{\le \ell-1}\right]$ is at most $(1 - \frac{2}{3e^2})$. Therefore, from Chernoff bound, we have

$$\Pr\left[\textsc{SmallS} \;\middle|\; \bigcap_j \mathtt{AccurateAppUnion}^j \cap \mathtt{AccurateN}_{\le \ell}; \mathsf{U}_{\le \ell-1}\right] \le \frac{\eta}{2}.$$

Furthermore, from Theorem 2 (part 3), we have

$$\Pr\left[\overline{\bigcap_j \texttt{AccurateAppUnion}^j} \;\middle|\; \texttt{AccurateN}_{\le \ell}; \mathsf{U}_{\le \ell-1}\right] \le \frac{\eta}{2}.$$

Note that given the event $\texttt{AccurateN}_{\le \ell}$ and under the distributions obtained using $\mathsf{U}_{\le \ell-1}$, if the event SmallS does not happen and the event $\bigcap_j \texttt{AccurateAppUnion}^j$ holds then the distribution of $\mathsf{S}(q^\ell)$ and $\mathsf{U}(q^\ell)$ is identical. So,

$$\begin{aligned}
&\mathsf{TV}(S(q^\ell), U(q^\ell) \;\middle|\; \texttt{AccurateN}_{\le \ell}\;;\; \mathsf{U}_{\le \ell-1}) \\
&\le \Pr\left[\textsc{SmallS} \cup \overline{\bigcap_j \texttt{AccurateAppUnion}^j} \;\middle|\; \texttt{AccurateN}_{\le \ell}; \mathsf{U}_{\le \ell-1}\right] \\
&\le \Pr\left[\textsc{SmallS} \;\middle|\; \bigcap_j \texttt{AccurateAppUnion}^j \cap \texttt{AccurateN}_{\le \ell}; \mathsf{U}_{\le \ell-1}\right] \\
&\qquad + \Pr\left[\overline{\bigcap_j \texttt{AccurateAppUnion}^j} \;\middle|\; \texttt{AccurateN}_{\le \ell}; \mathsf{U}_{\le \ell-1}\right] \le \eta/2 + \eta/2 = \eta \qquad \square
\end{aligned}$$

4.3.1 Proof of Theorem 3. Our proof of the Theorem 3 relies on upper bounding the probability with which the estimate of Algorithm 3 is outside the desired range. For this, we will set up new random variables and relate its joint distribution with the statsitical distance between the samples constructed by the algorithm and the ideal set of uniform samples. Recall that $\{\mathsf{S}(q^\ell)\}_{\ell \in [0,n], q \in Q}$ represent the set of random variables corresponding to the (multi)sets of samples we obtain during run of Algorithm 3. Furthermore, $\{\mathsf{U}(q^\ell)\}_{\ell \in [0,n], q \in Q}$ represent the set of random variables that correspond to repeatedly sampling ns elements (with replacement) uniformly at random from $\mathcal{L}(q^\ell)$. It is worth observing that while, for all q, p, ℓ, ℓ', $\mathsf{S}(q^\ell)$ and $\mathsf{S}(p^{\ell'})$ are dependent but $\mathsf{U}(q^\ell)$ and $\mathsf{U}(q^{\ell'})$ are independent. Also, the definition of distribution $\mathsf{U}(q^\ell)$ is independent of Algorithm 3 as it is defined solely based on the set of $\mathcal{L}(\mathsf{S}(q^\ell))$. Since we are talking about the sets of random variables, there is an implicit assumption that these sets are ordered.

We will now define two sequences of random variables:

$$X = \{X_\ell^q\}_{\ell \in [0,n], q \in Q} \;\text{ and }\; Y = \{Y_\ell^q\}_{\ell \in [0,n], q \in Q},$$

Instead of explicitly specifying them, we will instead specify properties about their joint distribution. We will use the notation $X_{\le i}$ (and resp. $Y_{\le i}$)) to denote the ordered set $\{X_\ell^q\}_{\ell \in [1,i], q \in Q}$ (resp. the ordered set $\{Y_\ell^q\}_{\ell \in [1,i], q \in Q}$). The joint distribution of (X, Y) has the following properties:

(1) For every $q \in Q$, $X_0^q = Y_0^q = \mathsf{S}_0^q$.
(2) For every $q \in Q, \ell \in [1, n]$, $X_\ell^q = S(q^\ell)$.
(3) For all sets ω, sequences of sets ω', $\ell \in [1, n]$, and $q \in Q$, we have

$$\begin{aligned}
&\Pr\left[X_\ell^q = Y_\ell^q = \omega \mid (X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \texttt{AccurateN}_{\le \ell}\right] \\
&\qquad = \min\left(\Pr\left[U(q^\ell) = \omega\right], \Pr\left[X_\ell^q = \omega \mid (X_{\le \ell-1} = \omega') \cap \texttt{AccurateN}_{\le \ell}\right]\right) \\
&\Pr\left[X_\ell^q = \omega \mid (X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \texttt{AccurateN}_{\le \ell}\right] \le \Pr\left[X_\ell^q = \omega \mid (X_{\le \ell-1} = \omega') \cap \texttt{AccurateN}_{\le \ell}\right]
\end{aligned}$$

We remark that such a joint distribution always exists. This is because (3) only restricts the case when $X_\ell^q = Y_\ell^q$, thereby, providing enough choices for assignments of probability values to the cases where $X_\ell^q \neq Y_\ell^q$ such that Y_ℓ^q has a well-defined probability distribution. In Appendix A.1 we give a concrete distribution that realizes these properties.

We will begin with the following simple observation, whose proof is deferred to Appendix.

Claim 6. $\Pr[(X_{\le \ell-1} = Y_{\le \ell-1} = \omega) \cap \mathtt{AccurateN}_{\le \ell-1}] \le \Pr[U_{\le \ell-1} = \omega]$

Next, we relate the random variables defined above to the desired statistical distance:

Claim 7.

$$\Pr\left[(X_\ell^q \neq Y_\ell^q) \cap (X_{\le \ell-1} = Y_{\le \ell-1}) \cap \mathtt{AccurateN}_{\le \ell}\right] \le \mathsf{TV}(\mathsf{S}(q^\ell), \mathsf{U}(q^\ell) \;\middle|\; \mathtt{AccurateN}_{\le \ell}\,; \mathsf{U}_{\le \ell-1})$$

Proof.

$$\begin{aligned}
&\Pr\left[(X_\ell^q \neq Y_\ell^q) \cap (X_{\le \ell-1} = Y_{\le \ell-1}) \cap \mathtt{AccurateN}_{\le \ell}\right] \\
&= \sum_{\omega,\omega'} \Pr\left[(X_\ell^q = \omega) \cap (X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}\right] \\
&\quad - \sum_{\omega,\omega'} \Pr\left[(X_\ell^q = Y_\ell^q = \omega) \cap (X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}\right] \\
&= \sum_{\omega'} \Pr[(X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}] \cdot \\
&\qquad \left(\sum_{\omega} \Pr\left[(X_\ell^q = \omega \mid (X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}\right] \right. \\
&\qquad \left. - \sum_{\omega} \Pr\left[(X_\ell^q = Y_\ell^q = \omega \mid (X_{\le \ell-1} = Y_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}\right] \right) \\
&\le \sum_{\omega'} \Pr[U_{\le \ell-1} = \omega'] \cdot \Big(\sum_{\omega} \Big(\Pr\left[(X_\ell^q = \omega \mid (X_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}\right] \\
&\qquad\qquad - \min\left(\Pr\left[U(q^\ell) = \omega\right],\ \Pr\left[X_\ell^q = \omega \mid (X_{\le \ell-1} = \omega') \cap \mathtt{AccurateN}_{\le \ell}\right]\right)\Big)\Big) \\
&= \mathsf{TV}(\mathsf{S}(q^\ell), \mathsf{U}(q^\ell) \;\middle|\; \mathtt{AccurateN}_{\le \ell}\,; \mathsf{U}_{\le \ell-1})
\end{aligned}$$

□

We now focus on deriving an upper bound on $\mathtt{AccurateN}_n$ as stated in the following claim, whose proof is deferred to Appendix.

Claim 8.

$$\overline{\mathtt{AccurateN}_n} \subseteq \bigcup_{\ell=1}^{n-1} (\overline{\mathtt{AccurateN}_\ell} \cap \mathtt{AccurateN}_{\le \ell-1} \cap (X_{\le \ell-1} = Y_{\le \ell-1})) \cup \bigcup_{\ell=1}^{n-1} ((X_\ell \neq Y_\ell) \cap (X_{\le \ell-1} = Y_{\le \ell-1}) \cap \mathtt{AccurateN}_{\le \ell})$$

We can now finish the proof of Theorem 3. We first observe that:

$$\begin{aligned}
&\Pr\left[\overline{\mathtt{AccurateN}_\ell} \cap \mathtt{AccurateN}_{\le \ell-1} \cap (X_{\le \ell-1} = Y_{\le \ell-1})\right] \\
&\le \sum_{\omega,q} \Pr[\mathtt{AccurateN}_{\le \ell-1} \cap (X_{\le \ell-1} = Y_{\le \ell-1} = \omega)] \times \\
&\qquad\qquad \Pr\left[\overline{\mathtt{AccurateN}_{q,\ell}} \mid (X_{\le \ell-1} = Y_{\le \ell-1} = \omega) \cap \mathtt{AccurateN}_{\le \ell-1}\right] \\
&\le \sum_{\omega,q} \Pr[U_{\le \ell-1} = \omega] \Pr\left[\overline{\mathtt{AccurateN}_{q,\ell}} \mid (X_{\le \ell-1} = Y_{\le \ell-1} = \omega) \cap \mathtt{AccurateN}_{\le \ell-1}\right] \\
&= \sum_{\omega,q} \Pr[\mathtt{AccurateN}_{\le \ell-1} = \omega] \Pr\left[\overline{\mathtt{AccurateN}_{q,\ell}} \mid (\mathsf{S}_{\le \ell-1} = \omega \cap Y_{\le \ell-1} = \omega \cap \mathtt{AccurateN}_{\le \ell-1}\right] \\
&\le \sum_{q} \Pr\left[\overline{\mathtt{AccurateN}_{q,\ell}} \;\middle|\; \mathtt{AccurateN}_{\le \ell-1}; \mathsf{U}_{\le \ell-1}\right]
\end{aligned}$$

The final step in the derivation uses the observation that given $\mathsf{S}_{\leq \ell-1}$ and $\mathtt{AccurateN}_{\leq \ell-1}$, the probability of the event $\mathtt{AccurateN}_{q,\ell}$ is only dependent on the randomness of the algorithm AppUnion. Thus, given $\mathsf{S}_{\leq \ell-1}$ and $\mathtt{AccurateN}_{\leq \ell-1}$, the event $\mathtt{AccurateN}_{q,\ell}$ is conditionally independent of $Y_{\leq \ell-1}$.

Putting together, we have:

$$\Pr\left[\overline{\mathtt{AccurateN}_n}\right] \leq \sum_{\ell\in[1,n];q\in Q} \Pr\left[\overline{\mathtt{AccurateN}_{q,\ell}} \;\middle|\; \mathtt{AccurateN}_{\leq \ell-1}; \mathsf{U}_{\leq \ell-1}\right] + \sum_{\ell\in[1,n-1];q\in Q} \mathsf{TV}(\mathsf{S}(q^\ell), \mathsf{U}(q^\ell) \mid \mathtt{AccurateN}_{\leq \ell}\,; \mathsf{U}_{\leq \ell-1})$$

By Lemma 4 and Lemma 5 and applying union bound over all $\ell \in [1, n]$ and all $q \in Q$ we have

$$\Pr\left[\overline{\mathtt{AccurateN}_n}\right] \leq n \cdot m \cdot \eta + n \cdot m \cdot \eta \leq \delta.$$

To calculate the time complexity of Algorithm 3 we first observe that the **for** loop goes over nm elements. Now for each ℓ and q there is one call to $\mathtt{AppUnion}_{\beta,\eta}$ with the number of sets being $O(m)$ and xns calls to sample(). So by Theorem 1, the time taken by AppUnion is the time it takes to do $O(m/\beta^2 \log(m/\eta))$ number of calls to the membership oracle. So the number of calls to the membership oracle is $\widetilde{O}(mn^4/\epsilon^2 \log(1/\delta))$, ignoring the factors of $\log(n+m)$. The method sample() on the other hand is a recursive function which calls $\mathtt{AppUnion}_{\beta,\eta}()$ in every recursive step and the depth of the recursion is $O(n)$. So the time complexity for each of the xns calls to sample() is the time it takes for $\widetilde{O}(mn^5/\epsilon^2 \log(1/\delta))$ calls to the oracle. Now, we note that the time complexity of the membership calls can be amortized as follows. First, for every string $w \in \bigcup_{q,\ell}\{\mathsf{S}(q^\ell)\}$, we pre-compute and store the set of reachable states of w in $\mathcal{A}$ in time $m^2|w| \in O(m^2 n)$ time. This takes a total of $\widetilde{O}(m^2 n \cdot mn^5/\epsilon^2 \log(1/\delta)) = \widetilde{O}(m^3 n^6/\epsilon^2 \log(1/\delta))$ time. The membership checks (in Algorithm 1, Line 9) can now be performed in $O(1)$ time. So in total the time complexity of the Algorithm 3 is $\widetilde{O}(mn \cdot \mathtt{xns} \cdot \frac{mn^5}{\epsilon^2} \log(1/\delta) + m^3 n^6/\epsilon^4 \log(1/\delta))$ which is $\widetilde{O}((m^2 n^{10} + m^3 n^6) \cdot \frac{1}{\epsilon^4} \cdot \log^2(1/\delta))$.

5 CONCLUSIONS AND FUTURE WORK

We consider the approximate counting problem #NFA and propose a fully polynomial time randomized approximation scheme (FPRAS) that significantly improves the prior FPRAS [3]. Given the wide range of applications of counting and sampling from the language of an NFA in applications including databases, program analysis, testing and more broadly in Computer Science, we envision that further improvements in the complexity of approximating #NFA is a worthwhile avenue for future work.

ACKNOWLEDGMENTS

We want to express our sincerest gratitude to Weiming Feng for carefully reviewing the proofs and identifying several issues present in earlier versions of the paper. We are grateful to Yash Pote, Uddalok Sarkar, and the anonymous reviewers for their constructive feedback, which enhanced the presentation and brought attention to relevant prior research. Umang Mathur was partially supported by a Singapore Ministry of Education (MoE) Academic Research Fund (AcRF) Tier 1 grant. Kuldeep Meel's work was partly supported in part by the National Research Foundation Singapore [NRF-NRFFAI1-2019-0004], Ministry of Education Singapore Tier 2 grant MOE-T2EP20121-0011, and Ministry of Education Singapore Tier 1 Grant [R-252-000-B59-114].

REFERENCES

[1] Antoine Amarilli, Timothy van Bremen, and Kuldeep S. Meel. 2024. Conjunctive Queries on Probabilistic Graphs: The Limits of Approximability. In *27th International Conference on Database Theory, ICDT*, Vol. 290. 15:1–15:20. https://doi.org/10.4230/LIPICS.ICDT.2024.15

[2] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. 2017. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.* 50, 5, Article 68 (2017). https://doi.org/10.1145/3104031

[3] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. 2019. Efficient Logspace Classes for Enumeration, Counting, and Uniform Generation. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. 59–73. https://doi.org/10.1145/3294052.3319704

[4] Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. 2021. # NFA Admits an FPRAS: Efficient Enumeration, Counting, and Uniform Generation for Logspace Classes. *Journal of the ACM (JACM)* 68, 6 (2021), 1–40.

[5] Lucas Bang, Abdulbaki Aydin, Quoc-Sang Phan, Corina S. Păsăreanu, and Tevfik Bultan. 2016. String Analysis for Side Channels with Segmented Oracles. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 193–204.

[6] Alexandre Donzé, Rafael Valle, Ilge Akkaya, Sophie Libkind, Sanjit A. Seshia, and David Wessel. 2014. Machine Improvisation with Formal Specifications. In *Music Technology meets Philosophy - From Digital Echos to Virtual Ethos: Joint Proceedings of the 40th International Computer Music Conference, ICMC*. https://hdl.handle.net/2027/spo.bbp2372.2014.196

[7] Pengfei Gao, Jun Zhang, Fu Song, and Chao Wang. 2019. Verifying and Quantifying Side-Channel Resistance of Masked Software Implementations. *ACM Trans. Softw. Eng. Methodol.* 28, 3, Article 16 (jul 2019), 32 pages. https://doi.org/10.1145/3330392

[8] Vivek Gore, Mark Jerrum, Sampath Kannan, Z. Sweedyk, and Stephen R. Mahaney. 1997. A Quasi-Polynomial-Time Algorithm for Sampling Words from a Context-Free Language. *Inf. Comput.* 134, 1 (1997), 59–74. https://doi.org/10.1006/inco.1997.2621

[9] Kenneth E Iverson. 1962. A programming language. In *Proceedings of the May 1-3, 1962, spring joint computer conference*. 345–351.

[10] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. 1986. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theor. Comput. Sci.* 43 (1986), 169–188. https://doi.org/10.1016/0304-3975(86)90174-X

[11] Sampath Kannan, Z. Sweedyk, and Stephen R. Mahaney. 1995. Counting and Random Generation of Strings in Regular Languages. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 551–557.

[12] Richard M. Karp and Michael Luby. 1985. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *J. Complex.* 1, 1 (1985), 45–64. https://doi.org/10.1016/0885-064X(85)90021-4

[13] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical Model Checking: An Overview. In *Runtime Verification*. 122–135.

[14] Burcu Kulahcioglu Ozkan, Rupak Majumdar, and Simin Oraee. 2019. Trace Aware Random Testing for Distributed Systems. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 180 (oct 2019), 29 pages. https://doi.org/10.1145/3360606

[15] Seemanta Saha, Surendra Ghentiyala, Shihua Lu, Lucas Bang, and Tevfik Bultan. 2023. Obtaining Information Leakage Bounds via Approximate Model Counting. *Proc. ACM Program. Lang.* 7, PLDI, Article 167 (jun 2023). https://doi.org/10.1145/3591281

[16] Michael Sutton, Adam Greene, and Pedram Amini. 2007. *Fuzzing: Brute Force Vulnerability Discovery*. Addison-Wesley Professional.

[17] Timothy van Bremen and Kuldeep S. Meel. 2023. Probabilistic Query Evaluation: The Combined FPRAS Landscape. In *PODS*. 339–347.

[18] Carme Álvarez and Birgit Jenner. 1993. A very hard log-space counting class. *Theoretical Computer Science* 107, 1 (1993), 3–30. https://doi.org/10.1016/0304-3975(93)90252-O

A DETAILS FROM SECTION 4.3

A.1 Joint Distribution on X and Y

Here, we will outline a concrete joint distribution that satisfies the properties we describe in Section 4.3. We will present the definition inductively, first defining the slice $\{X_1^q, Y_1^q\}_{q\in Q}$, and then use the slices $\{X_j^q, Y_j^q\}_{q\in Q, 0\le j\le i}$ to define the slice $\{X_{i+1}^q, Y_{i+1}^q\}_{q\in Q}$. Our inductive definition will also allow us to establish the properties we outlined. In the following, we use the notation $\Omega_{X,i,q}$ (resp. $\Omega_{Y,i,q}$) to denote the domain of X_i^q (resp. Y_i^q). Analogously, we will use the notations $\Omega_{X,i}$, $\Omega_{Y,i}$, $\Omega_{X,\le i}$ and $\Omega_{Y,\le i}$ to denote respectively $\Pi_{q\in Q}\Omega_{X,i,q}$, $\Pi_{q\in Q}\Omega_{Y,i,q}$, $\Pi_{j=1}^{i}\Omega_{X,j}$, and $\Pi_{j=1}^{i}\Omega_{Y,j}$.

Base Case(i=0). Follows from the definition.

Inductive Case. We will define the distribution inductively by defining the conditional probabilities $\Pr\left[X_{i+1}^q = \omega_{X,i+1,q} \cap Y_{i+1}^q = \omega_{Y,i+1,q} | X_{\le i} = \omega_{X,\le i} \cap Y_{\le i} = \omega_{Y,\le i} \cap \mathtt{AccurateN}_{\le i}\right]$ and $\Pr\left[X_{i+1}^q = \omega_{X,i+1,q} \cap Y_{i+1}^q = \omega_{Y,i+1,q} | X_{\le i} = \omega_{X,\le i} \cap Y_{\le i} = \omega_{Y,\le i} \cap \overline{\mathtt{AccurateN}_{\le i}}\right]$.
For each $A \in \{\mathtt{AccurateN}_{\le i}, \overline{\mathtt{AccurateN}_{\le i}}\}$, we have:

$$\Pr\left[X_{i+1}^q = \omega_{X,i+1,q} \cap Y_{i+1}^q = \omega_{Y,i+1,q} | X_{\le i} = \omega_{X,\le i} \cap Y_{\le i} = \omega_{Y,\le i} \cap A\right]$$

$$= \begin{cases} \min\left(\begin{array}{l} \Pr\left[\mathsf{U}(q^{i+1}) = \omega_{X,i+1,q}\right], \\ \Pr\left[\mathsf{S}(q^{i+1}) = \omega_{X,i+1,q} | \mathsf{S}_{\le i} = \omega_{X,\le i} \cap A\right] \end{array}\right) & \text{if } \omega_{X,i+1,q} = \omega_{Y,i+1,q} \\ \dfrac{\left[\begin{array}{l} \Pr\left[\mathsf{S}(q^{i+1}) = \omega_{X,i+1,q} | \mathsf{S}_{\le i} = \omega_{X,\le i} \cap A\right] \\ - \\ \min\left(\begin{array}{l} \Pr\left[\mathsf{U}(q^{i+1}) = \omega_{X,i+1,q}\right], \\ \Pr\left[\mathsf{S}(q^{i+1}) = \omega_{X,i+1,q} | \mathsf{S}_{\le i} = \omega_{X,\le i} \cap A\right] \end{array}\right) \end{array}\right]}{|\Omega_{Y,i+1,q}| - 1} & \text{otherwise} \end{cases}$$

Let us first observe that for each $A \in \{\mathtt{AccurateN}_{\le i}, \overline{\mathtt{AccurateN}_{\le i}}\}$ and $\omega_{X,\le i}$, we have

$$\Pr\left[X_{i+1}^q = \omega_{X,i+1,q} | X_{\le i} = \omega_{X,\le i} \cap A\right] = \Pr\left[\mathsf{S}(q^{i+1}) = \omega_{X,i+1,q} | \mathsf{S}_{\le i} = \omega_{X,\le i} \cap A\right]$$

Also, it is clear that $\sum_{\omega_{X,\le i+1}, \omega_{Y,\le i+1}} \Pr\left[X_{\le i+1} = \omega_{X,\le i+1} \cap Y_{\le i+1} = \omega_{Y,\le i+1}\right] = 1$.

It is easy to see that the above joint distribution satisfies both the desired properties.

A.2 Proof of Claim 6

Claim 6. $\Pr[(X_{\le \ell-1} = Y_{\le \ell-1} = \omega) \cap \mathtt{AccurateN}_{\le \ell-1}] \le \Pr[\mathsf{U}_{\le \ell-1} = \omega]$

Proof.

$$\begin{aligned} &\Pr[(X_{\le \ell-1} = Y_{\le \ell-1} = \omega) \cap \mathtt{AccurateN}_{\le \ell-1}] \\ =&\Pr[(X_1 = Y_1 = \omega_1) \cap \mathtt{AccurateN}_1] \times \\ &\quad \Pr[(X_2 = Y_2 = \omega_2) \cap \mathtt{AccurateN}_2 \mid (X_1 = Y_1 = \omega_1) \cap \mathtt{AccurateN}_1] \times \\ &\quad\quad \Pr[(X_3 = Y_3 = \omega_3) \cap \mathtt{AccurateN}_3 \mid (X_{\le 2} = Y_{\le 2} = \omega_1\omega_2) \cap \mathtt{AccurateN}_{\le 2}] \times \ldots \\ =&\prod_{k=1}^{\ell} \Pr\left[(X_k = Y_k = \omega_k) \cap \mathtt{AccurateN}_k \mid (X_{\le (k-1)} = Y_{\le (k-1)} = \omega') \cap \mathtt{AccurateN}_{\le (k-1)}\right] \end{aligned}$$

Now, since $\Pr[A \cap B \mid C] = \frac{\Pr[A \cap B \cap C]}{\Pr[C]} \leq \frac{\Pr[A \cap B \cap C]}{\Pr[B \cap C]} = \Pr[A \mid B \cap C]$ so,

$$\begin{aligned}&\Pr\Big[(X_k = Y_k = \omega_k) \cap \mathtt{AccurateN}_k \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k-1)}\Big]\\ \leq&\Pr\Big[(X_k = Y_k = \omega_k) \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k)}\Big]\end{aligned}$$

So,

$$\begin{aligned}&\Pr\Big[(X_k = Y_k = \omega_k) \cap \mathtt{AccurateN}_k \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k-1)}\Big]\\ \leq&\Pr\Big[(X_k = Y_k = \omega_k) \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k)}\Big]\end{aligned}$$

From the second condition of the joint random variables X and Y we have

$$\Pr\Big[(X_k = Y_k = \omega_k) \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k)}\Big] \leq \Pr[U_{\leq k} = \omega]$$

So we have,

$$\begin{aligned}&\Pr[(X_{\leq\ell-1} = Y_{\leq\ell-1} = \omega) \cap \mathtt{AccurateN}_{\leq\ell-1}]\\ =&\prod_{k=1}^{\ell} \Pr\Big[(X_k = Y_k = \omega_k) \cap \mathtt{AccurateN}_k \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k-1)}\Big]\\ \leq&\prod_{k=1}^{\ell} \Pr\Big[(X_k = Y_k = \omega_k) \mid (X_{\leq(k-1)} = Y_{\leq(k-1)} = \omega') \cap \mathtt{AccurateN}_{\leq(k)}\Big]\\ \leq&\prod_{k=1}^{\ell} \Pr[U_{\leq k} = \omega] = \Pr[U_{\leq\ell-1} = \omega],\end{aligned}$$

the last equality since the random variables U_k are are uniformly generated and hence independent. □

A.3 Proof of Claim 8

Claim 8. $\overline{\mathtt{AccurateN}_n} \subseteq \bigcup_{\ell=1}^{n-1}(\overline{\mathtt{AccurateN}_\ell} \cap \mathtt{AccurateN}_{\leq\ell-1} \cap (X_{\leq\ell-1} = Y_{\leq\ell-1})) \cup \bigcup_{\ell=1}^{n-1}((X_\ell \neq Y_\ell) \cap (X_{\leq\ell-1} = Y_{\leq\ell-1}) \cap \mathtt{AccurateN}_{\leq\ell})$

Proof.

$$\overline{\mathtt{AccurateN}_n} \subseteq (\overline{\mathtt{AccurateN}_n} \cap (X_{\leq n-1} = Y_{\leq n-1})) \cup (X_{\leq n-1} \neq Y_{\leq n-1})$$

Expanding further, we have

$$\begin{aligned}\overline{\mathtt{AccurateN}_n} \subseteq&(\overline{\mathtt{AccurateN}_n} \cap \mathtt{AccurateN}_{\leq n-1} \cap (X_{\leq n-1} = Y_{\leq n-1}))\\ &\cup (\overline{\mathtt{AccurateN}_n} \cap \overline{\mathtt{AccurateN}_{\leq n-1}} \cap (X_{\leq n-1} = Y_{\leq n-1}))\\ &\cup ((X_{\leq n-1} \neq Y_{\leq n-1}) \cap \overline{\mathtt{AccurateN}_{\leq n-1}}) \cup ((X_{\leq n-1} \neq Y_{\leq n-1}) \cap \mathtt{AccurateN}_{\leq n-1})\end{aligned}$$

Observing $(\overline{\mathtt{AccurateN}_n} \cap \overline{\mathtt{AccurateN}_{\leq n-1}}) \subseteq \overline{\mathtt{AccurateN}_{\leq n-1}}$ and then combining second and third terms, we have

$$\begin{aligned}\overline{\mathtt{AccurateN}_n} \subseteq&(\overline{\mathtt{AccurateN}_n} \cap \mathtt{AccurateN}_{\leq n-1} \cap (X_{\leq n-1} = Y_{\leq n-1})) \cup \overline{\mathtt{AccurateN}_{\leq n-1}}\\ &\cup ((X_{\leq n-1} \neq Y_{\leq n-1}) \cap \mathtt{AccurateN}_{\leq n-1})\end{aligned}$$

Observe that $\overline{\mathtt{AccurateN}_{\leq n-1}} = \bigcup_{\ell=1}^{n-1} \overline{\mathtt{AccurateN}_\ell}$. Therefore, we can further applying the above recurrence, in particular, for every ℓ, we will use the following upper bound.

$$\overline{\mathtt{AccurateN}_\ell} \subseteq (\overline{\mathtt{AccurateN}_\ell} \cap (X_{\leq\ell-1} = Y_{\leq\ell-1})) \cup (X_{\leq\ell-1} \neq Y_{\leq\ell-1})$$

$$\text{Therefore,}\quad \overline{\texttt{AccurateN}_n} \subseteq \bigcup_{\ell=1}^{n}(\overline{\texttt{AccurateN}_\ell} \cap \texttt{AccurateN}_{\leq \ell-1} \cap (X_{\leq \ell-1} = Y_{\leq \ell-1})) \cup \bigcup_{\ell=1}^{n-1}((X_{\leq \ell-1} \neq Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell-1})$$

Let us focus on two successive terms of the second expression and using the equation $A = A \cup (\overline{A} \cap B)$

$$\underbrace{((X_{\leq \ell-1} \neq Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell-1})}_{A} \cup \underbrace{((X_{\leq \ell} \neq Y_{\leq \ell}) \cap \texttt{AccurateN}_{\leq \ell})}_{B}$$
$$= ((X_{\leq \ell-1} \neq Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell-1}) \cup ((X_{\leq \ell-1} = Y_{\leq \ell-1}) \cap (X_{\leq \ell} \neq Y_{\leq \ell}) \cap \texttt{AccurateN}_{\leq \ell})$$
$$\cup (\overline{\texttt{AccurateN}_{\leq \ell-1}} \cap (X_{\leq \ell} \neq Y_{\leq \ell}) \cap \texttt{AccurateN}_{\leq \ell})$$

Noting that $((X_{\leq \ell-1} = Y_{\leq \ell-1}) \cap (X_{\leq \ell} \neq Y_{\leq \ell}) = ((X_{\leq \ell-1} = Y_{\leq \ell-1}) \cap (X_\ell \neq Y_\ell)$ and the fact that $(\overline{\texttt{AccurateN}_{\leq \ell-1}} \cap \texttt{AccurateN}_{\leq \ell}) = \emptyset$ we have,

$$((X_{\leq \ell-1} \neq Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell-1} \cup ((X_{\leq \ell} \neq Y_{\leq \ell}) \cap \texttt{AccurateN}_{\leq \ell})$$
$$= ((X_{\leq \ell-1} \neq Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell-1}) \cup ((X_\ell \neq Y_\ell) \cap (X_{\leq \ell-1} = Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell})$$

Applying the above simplification for all terms,

$$\bigcup_{\ell=1}^{n-1}((X_{\leq \ell-1} \neq Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell-1}) = \bigcup_{\ell=1}^{n-1}((X_\ell \neq Y_\ell) \cap (X_{\leq \ell-1} = Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell})$$

$$\text{Therefore, } \overline{\texttt{AccurateN}_n} \subseteq \bigcup_{\ell=1}^{n-1}(\overline{\texttt{AccurateN}_\ell} \cap \texttt{AccurateN}_{\leq \ell-1} \cap (X_{\leq \ell-1} = Y_{\leq \ell-1})) \cup \bigcup_{\ell=1}^{n-1}((X_\ell \neq Y_\ell) \cap (X_{\leq \ell-1} = Y_{\leq \ell-1}) \cap \texttt{AccurateN}_{\leq \ell})$$

□

Received December 2023; revised February 2024; accepted March 2024