A Scalable Shannon Entropy Estimator*

Priyanka Golia[1,2], Brendan Juba[3], Kuldeep S. Meel[2]

[1] Indian Institute of Technology Kanpur
[2] National University of Singapore
[3] Washington University in St. Louis

Abstract. Quantified information flow (QIF) has emerged as a rigorous approach to quantitatively measure confidentiality; the information-theoretic underpinning of QIF allows the end-users to link the computed quantities with the computational effort required on the part of the adversary to gain access to desired confidential information. In this work, we focus on the estimation of Shannon entropy for a given program Π. As a first step, we focus on the case wherein a Boolean formula $\varphi(X, Y)$ captures the relationship between inputs X and output Y of Π. Such formulas $\varphi(X, Y)$ have the property that for every valuation to X, there exists exactly one valuation to Y such that φ is satisfied. The existing techniques require $\mathcal{O}(2^m)$ model counting queries, where $m = |Y|$.
We propose the first efficient algorithmic technique, called EntropyEstimation to estimate the Shannon entropy of φ with PAC-style guarantees, i.e., the computed estimate is guaranteed to lie within a $(1 \pm \varepsilon)$-factor of the ground truth with confidence at least $1-\delta$. Furthermore, EntropyEstimation makes only $\mathcal{O}(\frac{min(m,n)}{\varepsilon^2})$ counting and sampling queries, where $m = |Y|$, and $n = |X|$, thereby achieving a significant reduction in the number of model counting queries. We demonstrate the practical efficiency of our algorithmic framework via a detailed experimental evaluation. Our evaluation demonstrates that the proposed framework scales to the formulas beyond the reach of the previously known approaches.

1 Introduction

Over the past half-century, the cost effectiveness of digital services has led to an unprecedented adoption of technology in virtually all aspects of our modern lives. Such adoption has invariably led to sensitive information being stored in data centers around the world and increasingly complex software accessing the information in order to provide the services that form the backbone of our modern economy and social interactions. At the same time, it is vital that protected information does not leak, as such leakages may have grave financial and societal consequences. Consequently, the detection and prevention of information leakage in software have attracted sustained interest in the security community.

* EntropyEstimation is available open-sourced at https://github.com/meelgroup/entropyestimation. The names of authors are sorted alphabetically and the order does not reflect contribution.

The earliest efforts on information leakage focused on *qualitative* approaches that sought to return a Boolean output of the form "yes" or "no" [11,26,30]. While these qualitative approaches successfully capture situations where part of the code accesses prohibited information, such approaches are not well-suited to situations wherein some information leakage is inevitable. An oft-repeated example of such a situation is a *password checker* wherein every response "incorrect password" does leak information about the *secret password*. As a result, the past decade has seen the rise of quantified information flow analysis (QIF) as a rigorous approach to quantitatively measure confidentiality [7,53,57]. The information-theoretic underpinnings of QIF analyses allow an end-user to link the computed quantities with the probability of an adversary successfully guessing a secret, or the worst-case computational effort required for the adversary to infer the underlying confidential information. Consequently, QIF has been applied in diverse use-cases such as software side-channel detection [40], inferring search-engine queries through auto-complete responses sizes [21], and measuring the tendency of Linux to leak TCP-session sequence numbers [59].

The standard recipe for using the QIF framework is to measure the information leakage from an underlying program Π as follows. In a simplified model, a program Π maps a set of controllable inputs (C) and secret inputs (I) to outputs (O) observable to an attacker. The attacker is interested in inferring I based on the output O. A diverse array of approaches have been proposed to efficiently model Π, with techniques relying on a combination of symbolic analysis [48], static analysis [24], automata-based techniques [4,5,14], SMT-based techniques [47], and the like. For each, the core underlying technical problem is to determine the leakage of information for a given observation. We often capture this leakage using entropy-theoretic notions, such as Shannon entropy [7,16,48,53] or min-entropy [7,44,48,53]. In this work, we focus on computing Shannon entropy.

In this work, we focus on entropy estimation for programs modeled by Boolean formulas; nevertheless, our techniques are general and can be extended to other models such as automata-based frameworks. Let a formula $\varphi(X, Y)$ capture the relationship between X and Y such that for every valuation to X there is atmost one valuation to Y such that φ is satisfied; one can view X as the set of inputs and Y as the set of outputs. Let $m = |Y|$ and $n = |X|$. Let p be a probability distribution over $\{0,1\}^Y$ such that for every assignment to Y, $\sigma : Y \mapsto \{0,1\}$, we have $p_\sigma = \frac{|sol(\varphi(Y \mapsto \sigma))|}{2^n}$, where $sol(\varphi(Y \mapsto \sigma))$ denotes the set of solutions of $\varphi(Y \mapsto \sigma)$. Then, the entropy of φ is defined as $H_\varphi(Y) = \sum_\sigma p_\sigma \log \frac{1}{p_\sigma}$.

The past decade has witnessed a multitude of entropy estimation techniques with varying guarantees on the quality of their estimates [9,17,35,58]. The problem of computing the entropy of a distribution represented by a given circuit is closely related to the EntropyDifference problem considered by Goldreich and Vadhan [34], and shown to be SZK-complete. We therefore do not expect to obtain polynomial-time algorithms for this problem. The techniques that have been proposed to compute $H(\varphi)$ exactly compute p_σ for each σ. Observe that computing p_σ is equivalent to the problem of model counting, which seeks to compute the number of solutions of a given formula. Therefore, the exact tech-

niques require $\mathcal{O}(2^m)$ model-counting queries [13,27,39]; therefore, such techniques often do not scale for large values of m. Accordingly, the state of the art often relies on sampling-based techniques that perform well in practice but can only provide lower or upper bounds on the entropy [37,49]. As is often the case, techniques that only guarantee lower or upper bounds can output estimates that can be arbitrarily far from the ground truth. This raises the question: *can we design efficient techniques for approximate estimation, whose estimates have PAC-style* (ε, δ) *guarantees? I.e., can we compute an estimate that is guaranteed to lie within a* $(1+\varepsilon)$*-factor of the ground truth for all possible values, with confidence at least* $1-\delta$*?*

The primary contribution of our work is the first efficient algorithmic technique (given in our algorithm EntropyEstimation), to estimate $H_\varphi(Y)$ with PAC-style guarantees for all possible values of $H_\varphi(Y)$. In particular, given a formula φ, EntropyEstimation returns an estimate that is guaranteed to lie within a $(1 \pm \varepsilon)$-factor of $H_\varphi(Y)$ with confidence at least $1-\delta$. We stress that we obtain such a multiplicative estimate even when $H_\varphi(Y)$ is very small, as in the case of a password-checker as described above. Furthermore, EntropyEstimation makes only $\mathcal{O}(\frac{min(m,n)}{\varepsilon^2})$ counting and sampling queries even though the support of the distribution specified by φ can be of the size $\mathcal{O}(2^m)$.

While the primary focus of the work is theoretical, we seek to demonstrate that our techniques can be translated into practically efficient algorithms. As such, we focused on developing a prototype using off-the-shelf samplers and counters. As a first step, we use GANAK [52] for model counting queries and SPUR [3] for sampling queries. Our empirical analysis demonstrates that EntropyEstimation can be translated into practice and achieves significant speedup over baseline.

It is worth mentioning that recent approaches in quantified information leakage focus on programs that can be naturally translated to string and SMT constraints, and therefore, employ model counters for string and SMT constraints. Since counting and sampling are closely related, we hope the algorithmic improvements attained by EntropyEstimation will lead to the development of samplers in the context of SMT and string constraints, and would lead to practical implementation of EntropyEstimation for other domains. We stress again that while we present EntropyEstimation for programs modeled as a Boolean formula, our analysis applies other approaches, such as automata-based approaches, modulo access to the appropriate sampling and counting oracles.

in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

We use lower case letters (with subscripts) to denote propositional variables and upper case letters to denote a subset of variables. The formula $\exists Y \varphi(X, Y)$ is existentially quantified in Y, where $X = \{x_1, \cdots, x_n\}$ and $Y = \{y_1, \cdots, y_m\}$. For notational clarity, we use φ to refer to $\varphi(X, Y)$ when clear from the context. We denote $Vars(\varphi)$ as the set of variables appearing in $\varphi(X, Y)$. A literal is a boolean variable or its negation.

A *satisfying assignment* or solution of a formula φ is a mapping $\tau : Vars(\varphi) \to \{0,1\}$, on which the formula evaluates to True. For $V \subseteq Vars(\varphi)$, $\tau_{\downarrow V}$ represents the truth values of variables in V in a satisfying assignment τ of φ. We denote the set of all the solutions of φ as $sol(\varphi)$. For $S \subseteq Vars(\varphi)$, we define $sol(\varphi)_{\downarrow S}$ as the set of solutions of φ projected on S.

The problem of *model counting* is to compute $|sol(\varphi)|$ for a given formula φ. Projected model counting is defined analogously using $sol(\varphi)_{\downarrow S}$ instead of $sol(\varphi)$, for a given projection set[4] $S \subseteq Vars(\varphi)$. A *uniform sampler* outputs a solution $y \in sol(\varphi)$ such that $\mathsf{Pr}[y \text{ is output}] = \frac{1}{|sol(\varphi)|}$.

We say that φ is a circuit formula if for all assignments $\tau_1, \tau_2 \in sol(\varphi)$, we have $\tau_{1\downarrow X} = \tau_{2\downarrow X} \implies \tau_1 = \tau_2$. It is worth remarking that if φ is a circuit formula, then X is an independent support.

For a circuit formula $\varphi(X,Y)$ and for $\sigma : Y \mapsto \{0,1\}$, we define $p_\sigma = \frac{|sol(\varphi(Y \mapsto \sigma))|}{|sol(\varphi)_{\downarrow X}|}$. Given a circuit formula $\varphi(X,Y)$, we define the entropy of φ, denoted by $H_\varphi(Y)$ as follows: $H_\varphi(Y) = -\sum_{\sigma \in 2^Y} p_\sigma \log(p_\sigma)$.

3 Related Work

The Shannon entropy is a fundamental concept in information theory, and as such have been studied by theoreticians and practitioners alike. While this is the first work, to the best of our knowledge, that provides Probabilistic Approximately Correct (PAC) (ε, δ)-approximation guarantees for all values of the entropy, while requiring only logarithmically (in the size of the support of distribution) many queries, we survey below prior work relevant to ours.

Goldreich and Vadhan [34] showed that the problem of estimating the entropy for circuit formulas is complete for statistical zero-knowledge. Estimation of the entropy via collision probabilities has been considered in the statistical physics community, but these techniques only provide lower bounds [43,55]. Batu et al. [9] considered entropy estimation in a *black-box* model wherein one is allowed to sample $\sigma \in 2^Y$ with probability proportional to p_σ and p_σ is revealed along with the sample σ. Batu et al. showed that any algorithm that can estimate the entropy within a factor of 2 in this model must use $\Omega(2^{m/8})$ samples. Furthermore, Batu et al. proposed a multiplicative approximation scheme assuming a lower bound on H — precisely, it required a number of samples that grow linearly with $1/H$; their scheme also gives rise to an additive approximate scheme. Guha et al. [35] improved Batu et al's scheme to obtain (ϵ, δ) multiplicative estimates using $\mathcal{O}(\frac{m \log \frac{1}{\delta}}{\epsilon^2 H})$ samples, matching Batu et al's lower bound. Note that this grows with $1/H$.

A more restrictive model has been considered wherein we only get access to samples (with the assurance that every σ is sampled with probability proportional to p_σ). Valiant and Valiant [58] obtained an asymptotically optimal algorithm in this setting, which requires $\Theta(\frac{2^m}{\epsilon^2 m})$ samples to obtain an ϵ additive

[4] Projection set has been referred to as sampling set in prior work [19,54]

approximation. Chakraborty et al. [17] considered the problem in a different setting, in which the algorithm is given the ability to sample σ from a *conditional distribution*: the algorithm is permitted to specify a set S, and obtains σ from the distribution conditioned on $\sigma \in S$. We remark that as discussed below, our approach makes use of such conditional samples, by sampling from a modified formula that conjoins the circuit formula to a formula for membership in S. In any case, Chakraborty et al. use $\mathcal{O}(\frac{1}{\epsilon^8} m^7 \log \frac{1}{\delta})$ conditional samples to approximately learn the distribution, and can only provide an additive approximation of entropy. A helpful survey of all of these different models and algorithms was recently given by Canonne [15].

In this paper, we rely on the advances in model counting. Theoretical investigations into model counting were initiated by Valiant in his seminal work that defined the complexity class #P and showed that the problem of model counting is #P-complete. From a practical perspective, the earliest work on model counting [12] focused on improving enumeration-based strategies via partial solutions. Subsequently, Bayardo and Pehoushek [10] observed that if a formula can be partitioned into subsets of clauses, also called components, such that each of the subsets is over disjoint sets of variables, then the model count of the formula is the product of the model counts of each of the components. Building on Bayardo and Pehoushek's scheme, Sang et al. [50] showed how conflict-driven clause learning can be combined with component caching, which has been further improved by Thurley [56] and Sharma et al. [52]. Another line of work focuses on compilation-based techniques, wherein the core approach is to compile the input formula into a subset $\mathcal{L}$ in negation normal form, so that counting is tractable for $\mathcal{L}$. The past five years have witnessed a surge of interest in the design of projected model counters [6,18,20,42,45,52]. In this paper, we employ GANAK [52], the state of the art projected model counter; an entry based on GANAK won the projected model counting track at the 2020 model counting competition [31].

Another crucial ingredient for our technique is access to an efficient sampler. Counting and sampling are closely related problems, and therefore, the development of efficient counters spurred the research on the development of samplers. In a remarkable result, Huang and Darwiche [36] showed that the traces of model counters are in d-DNNF (deterministic Decomposable Negation Normal Form [25]), which was observed to support sampling in polynomial time [51]. Achlioptas, Hammoudeh, and Theodoropoulos [3] observed that one can improve the space efficiency by performing an on-the-fly traversal of the underlying trace of a model counter such as SharpSAT [56].

Our work builds on a long line of work in the QIF community that identified a close relationship between quantified information flow and model counting [4,5,27,33,38,59]. There are also many symbolic execution based approaches for QIF based on model counting that would require model counting calls that are linear in the size of observable domain, that is, exponential in the number of bits represents the domain [8,46]. Another closely related line of the work concerns the use of model counting in side-channel analysis [28,29,33]. Similarly, there exists sampling based approaches for black-box leakage estimation that

either require too many samples, much larger than the product of size of input and output domain [23] to converge or uses ML based approaches that predict the error of the idea classifier for predicting secrets given observable [22]. However, these approaches can not provide PAC guarantees on the estimation. While we focus on the case where the behavior of a program can be modeled with a Boolean formula φ, the underlying technique is general and can extended to cases where programs (and their abstractions) are modeled using automata [4,5,14].

Before concluding our discussion of prior work, we remark that Köpf and Rybalchenko [41] used Batu et al.'s [9] lower bounds to conclude that their scheme could not be improved without usage of structural properties of the program. In this context, our paper continues the direction alluded by Köpf and Rybalchenko and designs the first efficient multiplicative approximation scheme by utilizing white-box access to the program.

4 EntropyEstimation: Efficient Estimation of $H(\varphi)$

In this section, we focus on the primary technical contribution of our work: an algorithm, called EntropyEstimation, that takes a circuit formula $\varphi(X, Y)$ and returns an (ε, δ) estimate of $H(\varphi)$. We first provide a detailed technical overview of the design of EntropyEstimation in Section 4.1, then provide a detailed description of the algorithm, and finally, provide the accompanying technical analysis of the correctness and complexity of EntropyEstimation.

4.1 Technical Overview

At a high level, EntropyEstimation uses a median of means estimator, i.e., we first estimate $H(\varphi)$ to within a $(1\pm\varepsilon)$-factor with probability at least $\frac{5}{6}$ by computing the mean of the underlying estimator and then take the median of many such estimates to boost the probability of correctness to $1-\delta$.

Let us consider a random variable S over the domain $sol(\varphi)_{\downarrow Y}$ such that $\Pr[S=\sigma] = p_\sigma$ wherein $\sigma \in sol(\varphi)_{\downarrow Y}$ and consider the self-information function $g : sol(\varphi)_{\downarrow Y} \to [0,\infty)$, given by $g(\sigma) = \log(\frac{1}{p_\sigma})$. Observe that the entropy $H(\varphi) = \mathsf{E}[g(S)]$. Therefore, a simple estimator would be to sample S using our oracle and then estimate the expectation of $g(S)$ by a sample mean. At this point, we observe that given access to a uniform sampler, UnifSample, we can simply first sample $\tau \in sol(\varphi)$ uniformly at random, and then set $S = \tau_{\downarrow Y}$, which gives $\Pr[S = \tau_{\downarrow Y}] = p_{\tau_{\downarrow Y}}$. Furthermore, observe that $g(\sigma)$ can be computed via a query to a model counter. In their seminal work, Batu et al. [9] observed that the variance of $g(S)$, denoted by $\mathsf{variance}[g(S)]$, can be at most m^2. The required number of sample queries, based on a straightforward analysis, would be $\Theta\left(\frac{\mathsf{variance}[g(S)]}{\varepsilon^2\cdot(\mathsf{E}[g(S)])^2}\right) = \Theta\left(\frac{\sum p_\sigma \log^2 \frac{1}{p_\sigma}}{(\sum p_\sigma \log \frac{1}{p_\sigma})^2}\right)$. However, $\mathsf{E}[g(S)] = H(\varphi)$ can be arbitrarily close to 0, and therefore, this does not provide a reasonable upper bound on the required number of samples.

To address the lack of lower bound on $H(\varphi)$, we observe that for φ to have $H(\varphi) < 1$, there must exist $\sigma_{high} \in sol(\varphi)_{\downarrow Y}$ such that $p_{(\sigma_{high})} > \frac{1}{2}$. We then observe that given access to a sampler and counter, we can identify such a σ_{high} with high probability, thereby allowing us to consider the two cases separately: (A) $H(\varphi) > 1$ and (B) $H(\varphi) < 1$. Now, for case (A), we could use Batu et al's bound for $\mathsf{variance}[g(S)]$ [9] and obtain an estimator that would require $\Theta\left(\frac{\mathsf{variance}[g(S)]}{\varepsilon^2 \cdot (\mathsf{E}[g(S)])^2}\right)$ sampling and counting queries. It is worth remarking that the bound $\mathsf{variance}[g(S)] \leq m^2$ is indeed tight as a uniform distribution over $sol(\varphi)_{\downarrow X}$ would achieve the bound. Therefore, we instead focus on the expression $\frac{\mathsf{variance}[g(S)]}{(\mathsf{E}[g(S)])^2}$ and prove that for the case when $\mathsf{E}[g(S)] = H(\varphi) > h$, we can upper bound $\frac{\mathsf{variance}[g(S)]}{(\mathsf{E}[g(S)])^2}$ by $\frac{(1+o(1))\cdot m}{h \cdot \varepsilon^2}$, thereby reducing the complexity from m^2 to m (Observe that we have $H(\varphi) > 1$, that is, we can take $h = 1$).

Now, we return to the case (B) wherein we have identified $\sigma_{high} \in sol(\varphi)_{\downarrow Y}$ with $p_{\sigma_{high}} > \frac{1}{2}$. Let $r = p_{\sigma_{high}}$ and $H_{rem} = \sum_{\sigma \in sol(\varphi)_{\downarrow Y} \setminus \sigma_{high}} p_\sigma \log \frac{1}{p_\sigma}$. Note that $H(\varphi) = r \log \frac{1}{r} + H_{rem}$. Therefore, we focus on estimating H_{rem}. To this end, we define a random variable T that takes values in $sol(\varphi)_{\downarrow Y} \setminus \sigma_{high}$ such that $\Pr[T = \sigma] = \frac{p_\sigma}{1-r}$. Using the function g defined above, we have $H_{rem} = (1-r) \cdot \mathsf{E}[g(T)]$. Again, we have two cases, depending on whether $H_{rem} \geq 1$ or not; if it is, then we can bound the ratio $\frac{\mathsf{variance}[g(T)]}{\mathsf{E}[g(T)]^2}$ similarly to case (A). If not, we observe that the denominator is at least 1 for $r \geq 1/2$. And, when H_{rem} is so small, we can upper bound the numerator by $(1 + o(1))m$, giving overall $\frac{\mathsf{variance}[g(T)]}{(\mathsf{E}[g(T)])^2} \leq (1 + o(1)) \cdot \frac{1}{\varepsilon^2} \cdot m$. We can thus estimate H_{rem} using the median of means estimator.

4.2 Algorithm Description

Algorithm 1 presents the proposed algorithmic framework EntropyEstimation. EntropyEstimation takes a formula $\varphi(X, Y)$, a tolerance parameter ε, a confidence parameter δ as input, and returns an estimate $\hat{h}$ of the entropy $H_\varphi(Y)$, that is guaranteed to lie within a $(1 \pm \varepsilon)$-factor of $H_\varphi(Y)$ with confidence at least $1 - \delta$. Algorithm 1 assumes access to following subroutines:

ComputeCount: The subroutine ComputeCount takes a formula $\varphi(X, Y)$ and a projection set $V \subseteq X \cup Y$ as input, and returns a projected model count of $\varphi(X, Y)$ over V.

UnifSample: The subroutine UnifSample takes a formula $\varphi(X, Y)$ as an input and returns a uniformly sampled satisfying assignment of $\varphi(X, Y)$.

SampleEst: Algorithm 2 presents the subroutine SampleEst, which also assumes access to the ComputeCount and UnifSample subroutines. SampleEst takes as input a formula $\varphi(X, Y)$; the projected model count of $\varphi(X, Y)$ over X, z; the number of required samples, t; and a confidence parameter δ, and returns a median-of-means estimate of the entropy. Algorithm 2 starts off by computing the value of T, the required number of repetitions to ensure

Algorithm 1 EntropyEstimation$(\varphi(X,Y),\varepsilon,\delta)$

1: $m \leftarrow |Y|; n \leftarrow |X|$
2: $z \leftarrow \mathsf{ComputeCount}(\varphi(X,Y),X)$
3: **for** $i \in [1,\log(10/\delta)]$ **do**
4: $\quad \tau \leftarrow \mathsf{UnifSample}(\varphi)$
5: $\quad r = z^{-1} \cdot \mathsf{ComputeCount}(\varphi(X,Y) \wedge (Y \leftrightarrow \tau_{\downarrow Y}),X)$
6: $\quad$ **if** $r > \frac{1}{2}$ **then**
7: $\quad\quad \hat{\varphi} \leftarrow \varphi \wedge (Y \not\leftrightarrow \tau_{\downarrow Y})$
8: $\quad\quad t \leftarrow \frac{6}{\epsilon^2} \cdot \min\left\{\frac{n}{2\log\frac{1}{1-r}}, m + \log(m + \log m + 2.5)\right\}$
9: $\quad\quad \hat{h}_{rem} \leftarrow \mathsf{SampleEst}(\hat{\varphi},z,t,0.9\cdot\delta)$
10: $\quad\quad \hat{h} \leftarrow (1-r)\hat{h}_{rem} + r\log(\frac{1}{r})$
11: $\quad\quad$ **return** $\hat{h}$
12: $t \leftarrow \frac{6}{\epsilon^2} \cdot (\min\{n, m + \log(m + \log m + 1.1)\} - 1)$
13: $\hat{h} \leftarrow \mathsf{SampleEst}(\varphi,z,t,0.9\cdot\delta)$
14: **return** $\hat{h}$

Algorithm 2 SampleEst(φ,z,t,δ)

1: $C \leftarrow [\,]$
2: $T \leftarrow \frac{9}{2}\log\frac{2}{\delta}$
3: **for** $i \in [1,T]$ **do**
4: $\quad est \leftarrow 0$
5: $\quad$ **for** $j \in [1,t]$ **do**
6: $\quad\quad \tau \leftarrow \mathsf{UnifSample}(\varphi)$
7: $\quad\quad r = z^{-1} \cdot \mathsf{ComputeCount}(\varphi(X,Y) \wedge (Y \leftrightarrow \tau_{\downarrow Y}),X)$
8: $\quad\quad est \leftarrow est + \log(1/r)$
9: $\quad C.\text{Append}(\frac{est}{t})$
10: **return** Median(C)

at least $1-\delta$ confidence for the estimate. The algorithm has two loops—one outer loop (Lines 3-9), and one inner loop (Lines 5-8). The outer loop runs for $[\frac{9}{2}\log(\frac{2}{\delta})]$ rounds, where in each round, Algorithm 2 updates a list C with the mean estimate, *est*. In the inner loop, in each round, Algorithm 2 updates the value of *est*: Line 6 draws a sample τ using the $\mathsf{UnifSample}(\varphi(X,Y))$ subroutine. At Line 7, value of r is computed as the ratio of the projected model count of X in $\varphi(X,Y) \wedge (Y \leftrightarrow \tau_{\downarrow Y})$ to z. To compute the projected model count, Algorithm 2 calls the subroutine ComputeCount on input $(\varphi(X,Y) \wedge (Y \leftrightarrow \tau_{\downarrow Y}),X)$. At line 8, *est* is updated with $\log(\frac{1}{r})$, and at line 9, the final *est* is added to C. Finally, at line 10, Algorithm 2 returns the median of C.

Returning back to Algorithm 1, it starts by computing the value of z as the projected model count of $\varphi(X,Y)$ over X at line 2. The projected model count is computed by calling the ComputeCount subroutine. Next, Algorithm 1 attempts to determine whether there exists an output τ_{high} with probability greater than

$1/2$ or not by iterating over lines 3-11 for $\lceil \log(10/\delta) \rceil$ rounds. Line 4, draws a sample τ by calling the $\mathsf{UnifSample}(\varphi(X,Y))$ subroutine. Line 5 computes the value of r by taking the ratio of the projected model count of $\varphi(X,Y) \wedge (Y \leftrightarrow \tau_{\downarrow Y})$ to z. Line 6 checks whether the value of r is greater than $1/2$ or not, and chooses one of the two paths based on the value of r:

1. If the value of r turns out to be greater than $1/2$, the formula $\varphi(X,Y)$ is updated to $\varphi(X,Y) \wedge (Y \not\leftrightarrow \tau_{\downarrow Y})$ at line 7. The resulting formula is denoted by $\hat{\varphi}(X,Y)$. Then, the value of required number of samples, t, is calculated as per the calculation shown at line 8. At line 9, the subroutine $\mathsf{SampleEst}$ is called with $\hat{\varphi}(X,Y)$, z, t, and $0.9 \times \delta$ as arguments to compute the estimate $\hat{h}_{rem}$. Finally, it computes the estimate $\hat{h}$ at line 10.
2. If the value of r is at most $1/2$ in every round, the number of samples we use, t, is calculated as per the calculation shown at line 12. At line 13, the subroutine $\mathsf{SampleEst}$ is called with $\varphi(X,Y)$, z, t, and $0.9 \times \delta$ as arguments to compute the estimate $\hat{h}$.

4.3 Theoretical Analysis

Theorem 1. *Given a circuit formula φ with $|Y| \geq 2$, a tolerance parameter $\varepsilon > 0$, and confidence parameter $\delta > 0$, the algorithm* $\mathsf{EntropyEstimation}$ *returns $\hat{h}$ such that*

$$\Pr\left[(1-\varepsilon)H_\varphi(Y) \leq \hat{h} \leq (1+\varepsilon)|H_\varphi(Y)|\right] \geq 1-\delta$$

We first analyze the median-of-means estimator computed by $\mathsf{SampleEst}$.

Lemma 1. *Given a circuit formula φ and $z \in \mathbb{N}$, an accuracy parameter $\varepsilon > 0$, a confidence parameter $\delta > 0$, and a batch size $t \in \mathbb{N}$ for which*

$$\frac{1}{t\epsilon^2} \cdot \left(\frac{\sum_{\sigma \in 2^Y} \frac{|sol(\varphi(Y \mapsto \sigma))|}{|sol(\varphi)_{\downarrow X}|} (\log \frac{z}{|sol(\varphi(Y \mapsto \sigma))|})^2}{\left(\sum_{\sigma \in 2^Y} \frac{|sol(\varphi(Y \mapsto \sigma))|}{|sol(\varphi)_{\downarrow X}|} \log \frac{z}{|sol(\varphi(Y \mapsto \sigma))|}\right)^2} - 1 \right) \leq 1/6$$

the algorithm $\mathsf{SampleEst}$ *returns an estimate $\hat{h}$ such that with probability $1-\delta$,*

$$\hat{h} \leq (1+\epsilon) \sum_{\sigma \in 2^Y} \frac{|sol(\varphi(Y \mapsto \sigma))|}{|sol(\varphi)_{\downarrow X}|} \log \frac{z}{|sol(\varphi(Y \mapsto \sigma))|} \text{ and}$$

$$\hat{h} \geq (1-\epsilon) \sum_{\sigma \in 2^Y} \frac{|sol(\varphi(Y \mapsto \sigma))|}{|sol(\varphi)_{\downarrow X}|} \log \frac{z}{|sol(\varphi(Y \mapsto \sigma))|}.$$

Proof. Let R_{ij} be the random value taken by r in the ith iteration of the outer loop and jth iteration of the inner loop. We observe that $\{R_{ij}\}_{(i,j)}$ are a family of i.i.d. random variables. Let $C_i = \sum_{j=1}^{t} \frac{1}{t} \log \frac{1}{R_{ij}}$ be the value appended to C at

the end of the ith iteration of the loop. Clearly $\mathsf{E}[C_i] = \mathsf{E}[\log \frac{1}{R_{ij}}]$. Furthermore, we observe that by independence of the R_{ij},

$$\mathsf{variance}[C_i] = \frac{1}{t}\mathsf{variance}[\log \frac{1}{R_{ij}}] = \frac{1}{t}(\mathsf{E}[(\log R_{ij})^2] - \mathsf{E}[\log \frac{1}{R_{ij}}]^2).$$

By Chebyshev's inequality, now,

$$\begin{aligned}\Pr\left[|C_i - \mathsf{E}[\log \frac{1}{R_{ij}}]| > \epsilon \mathsf{E}[\log \frac{1}{R_{ij}}]\right] &< \frac{\mathsf{variance}[C_i]}{\epsilon^2 \mathsf{E}[\log \frac{1}{R_{ij}}]^2} \\ &= \frac{\mathsf{E}[(\log R_{ij})^2] - \mathsf{E}[\log \frac{1}{R_{ij}}]^2}{t \cdot \epsilon^2 \mathsf{E}[\log \frac{1}{R_{ij}}]^2} \\ &\leq 1/6\end{aligned}$$

by our assumption on t.

Let $L_i \in \{0,1\}$ be the indicator random variable for the event that $C_i < \mathsf{E}[\log \frac{1}{R_{ij}}] - \epsilon \mathsf{E}[\log \frac{1}{R_{ij}}]$, and let $H_i \in \{0,1\}$ be the indicator random variable for the event that $C_i > \mathsf{E}[\log \frac{1}{R_{ij}}] + \epsilon \mathsf{E}[\log \frac{1}{R_{ij}}]$. Similarly, since these are disjoint events, $B_i = L_i + H_i$ is also an indicator random variable for the union. So long as $\sum_{i=1}^{T} L_i < T/2$ and $\sum_{i=1}^{T} H_i < T/2$, we note that the value returned by SampleEst is as desired. By the above calculation, $\Pr[L_i = 1] + \Pr[H_i = 1] = \Pr[B_i = 1] < 1/6$, and we note that $\{(B_i, L_i, H_i)\}_i$ are a family of i.i.d. random variables. Observe that by Hoeffding's inequality,

$$\Pr\left[\sum_{i=1}^{T} L_i \geq \frac{T}{6} + \frac{T}{3}\right] \leq \exp(-2T\frac{1}{9}) = \frac{\delta}{2}$$

and similarly $\Pr\left[\sum_{i=1}^{T} H_i \geq \frac{T}{2}\right] \leq \frac{\delta}{2}$. Therefore, by a union bound, the returned value is adequate with probability at least $1 - \delta$ overall.

The analysis of SampleEst relied on a bound on the ratio of the first and second "moments" of the self-information in our truncated distribution. Suppose for all assignments σ to Y, $p_\sigma \leq 1/2$. We observe that then $H_\varphi(Y) \geq \sum_{\sigma \in 2^Y} p_\sigma \cdot 1 = 1$. We also observe that on account of the uniform distribution on X, any σ in the support of the distribution has $p_\sigma \geq 1/2^{|X|}$. Such bounds allow us to bound the relative variance of the self information:

Lemma 2. *Let $\{p_\sigma \in [1/2^{|X|}, 1]\}_{\sigma \in 2^Y}$ be given. Then,*

$$\sum_{\sigma \in 2^Y} p_\sigma (\log p_\sigma)^2 \leq |X| \sum_{\sigma \in 2^Y} p_\sigma \log \frac{1}{p_\sigma}$$

Proof. We observe simply that

$$\sum_{\sigma \in 2^Y} p_\sigma (\log p_\sigma)^2 \leq \log 2^{|X|} \sum_{\sigma \in 2^Y} p_\sigma \log \frac{1}{p_\sigma} = |X| \sum_{\sigma \in 2^Y} p_\sigma \log \frac{1}{p_\sigma}.$$

Lemma 3. *Let* $\{p_\sigma \in [0,1]\}_{\sigma\in 2^Y}$ *be given with* $\sum_{\sigma\in 2^Y} p_\sigma \le 1$ *and*

$$H = \sum_{\sigma\in 2^Y} p_\sigma \log \frac{1}{p_\sigma} \ge 1.$$

Then

$$\frac{\sum_{\sigma\in 2^Y} p_\sigma (\log p_\sigma)^2}{\left(\sum_{\sigma\in 2^Y} p_\sigma \log \frac{1}{p_\sigma}\right)^2} \le \left(1 + \frac{\log(|Y| + \log|Y| + 1.1)}{|Y|}\right)|Y|.$$

Similarly, if $H \le 1$ *and* $|Y| \ge 2$,

$$\sum_{\sigma\in 2^Y} p_\sigma (\log p_\sigma)^2 \le |Y| + \log(|Y| + \log|Y| + 2.5).$$

Concretely, both cases give a bound that is at most $2|Y|$ for $|Y| \ge 3$; $|Y| = 8$ gives a bound that is less than $1.5 \times |Y|$ in both cases, $|Y| = 64$ gives a bound that is less than $1.1 \times |Y|$, etc.

Proof. By induction on the size of the support, denoted as supp and defined as $|\{\sigma \in 2^Y | p_\sigma > 0\}|$, we'll show that when $H \ge 1$, the ratio is at most $\log|\text{supp}| + \log(\log|\text{supp}| + \log\log|\text{supp}| + 1.1)$. The base case is when there are only two elements ($|Y| = 1$), in which case we must have $p_0 = p_1 = 1/2$, and the ratio is uniquely determined to be 1. For the induction step, observe that whenever any subset of the p_σ take value 0, this is equivalent to a distribution with smaller support, for which by induction hypothesis, we find the ratio is at most

$$\begin{aligned}&\log(|\text{supp}| - 1) + \log(\log(|\text{supp}| - 1) + \log\log(|\text{supp}| - 1) + 1.1)\\ &\qquad < \log|\text{supp}| + \log(\log|\text{supp}| + \log\log|\text{supp}| + 1.1).\end{aligned}$$

Consider any value of $H_\varphi(Y) = H$. With the entropy fixed, we need only maximize the numerator of the ratio with $H_\varphi(Y) = H$. Indeed, we've already ruled out a ratio of $|\text{supp}(Y)|$ for solutions in which any of the p_σ take value 0, and clearly we cannot have any $p_\sigma = 1$, so we only need to consider interior points that are local optima. We use the method of Lagrange multipliers: for some λ, all p_σ must satisfy $\log^2 p_\sigma + 2\log p_\sigma - \lambda(\log p_\sigma - 1) = 0$, which has solutions

$$\log p_\sigma = \frac{\lambda}{2} - 1 \pm \sqrt{(1 - \frac{\lambda}{2})^2 - \lambda} = \frac{\lambda}{2} - 1 \pm \sqrt{1 + \lambda^2/4}.$$

We note that the second derivatives with respect to p_σ are equal to $\frac{2\log p_\sigma}{p_\sigma} + \frac{2-\lambda}{p_\sigma}$ which are negative iff $\log p_\sigma < \frac{\lambda}{2} - 1$, hence we attain local maxima only for the solution $\log p_\sigma = \frac{\lambda}{2} - 1 - \sqrt{1 + \lambda^2/4}$. Thus, there is a single p_σ, which by the entropy constraint, must satisfy $|\text{supp}| p_\sigma \log \frac{1}{p_\sigma} = H$ which we'll show gives

$$p_\sigma = \frac{H}{|\text{supp}|(\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho)}$$

for some $\rho \leq 1.1$. For $|\text{supp}| = 3$, we know $1 \leq H \leq \log 3$, and we can verify numerically that $\log\left(\frac{\log \frac{3}{H} + \log\log \frac{3}{H} + \rho}{\log \frac{3}{H}}\right) \in (0.42, 0.72)$ for $\rho \in [0,1]$. Hence, by Brouwer's fixed point theorem, such a choice of $\rho \in [0,1]$ exists. For $|\text{supp}| \geq 4$, observe that $\frac{|\text{supp}|}{H} \geq 2$, so $\log\left(\frac{\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H}}{\log \frac{|\text{supp}|}{H}}\right) > 0$. For $|\text{supp}| = 4$, $\log\left(\frac{\log \frac{4}{H} + \log\log \frac{4}{H} + \rho}{\log \frac{4}{H}}\right) \in [0,1]$, and similarly for all integer values of $|\text{supp}|$ up to 15, $\log\left(\frac{\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + 1.1}{\log \frac{|\text{supp}|}{H}}\right) < 1.1$, so we can obtain $\rho \in (0, 1.1)$. Finally, for $|\text{supp}| \geq 16$, we have $\frac{|\text{supp}|}{H} \leq 2^{|\text{supp}|/2H}$, and hence $\frac{\log\log \frac{|\text{supp}|}{H} + \rho}{\log \frac{|\text{supp}|}{H}} \leq 1$, so

$$\begin{aligned}|\text{supp}|\frac{H(\log \frac{|\text{supp}|}{H} + \log(\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho))}{|\text{supp}|(\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho)} \\ \leq H\frac{\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + 1}{\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho}\end{aligned}$$

Hence it is clear that this gives H for some $\rho \leq 1$. Observe that for such a choice of ρ, using the substitution above, the ratio we attain is

$$\begin{aligned}\frac{|\text{supp}| \cdot H}{H^2 \cdot |\text{supp}|(\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho)} \left(\log \frac{|\text{supp}|(\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho)}{H}\right)^2 \\ = \frac{1}{H}(\log \frac{|\text{supp}|}{H} + \log(\log \frac{|\text{supp}|}{H} + \log\log \frac{|\text{supp}|}{H} + \rho))\end{aligned}$$

which is monotone in $1/H$, so using the fact that $H \geq 1$, we find it is at most

$$\log|\text{supp}| + \log(\log|\text{supp}| + \log\log|\text{supp}| + \rho)$$

which, recalling $\rho < 1.1$, gives the claimed bound.

For the second part, observe that by the same considerations, for fixed H,

$$\sum_{\sigma \in 2^Y} p_\sigma (\log p_\sigma)^2 = H \log \frac{1}{p_\sigma}$$

for the unique choice of p_σ for $|Y|$ and H as above, i.e., we will show that for $|Y| \geq 2$, it is

$$H\left(\log \frac{2^{|Y|}}{H} + \log(\log \frac{2^{|Y|}}{H} + \log\log \frac{2^{|Y|}}{H} + \rho)\right)$$

for some $\rho \in (0, 2.5)$. Indeed, we again consider the function

$$f(\rho) = \frac{\log(\log \frac{2^{|Y|}}{H} + \log\log \frac{2^{|Y|}}{H} + \rho)}{\log\log \frac{2^{|Y|}}{H}},$$

and observe that for $2^{|Y|}/H > 2$, $f(0) > 0$. Now, when $|Y| \geq 2$ and $H \leq 1$, $2^{|Y|}/H \geq 4$. We will see that the function $d(\rho) = f(\rho) - \rho$ has no critical points for $2^{|Y|}/H \geq 4$ and $\rho > 0$, and hence its maximum is attained at the boundary, i.e., at $\frac{2^{|Y|}}{H} = 4$, at which point we see that $f(2.5) < 2.5$. So, for such values of $\frac{2^{|Y|}}{H}$, $f(\rho)$ maps $[0, 2.5]$ into $[0, 2.5]$ and hence by Brouwer's fixed point theorem again, for all $|Y| \geq 4$ and $H \geq 1$ some $\rho \in (0, 2.5)$ exists for which $p_\sigma = \log \frac{2^{|Y|}}{H} + \log(\log \frac{2^{|Y|}}{H} + \log\log \frac{2^{|Y|}}{H} + \rho)$ gives $\sum_{p_\sigma \in 2^Y} p_\sigma \log \frac{1}{p_\sigma} = H$.

Indeed, $d'(\rho) = \frac{1}{\ln 2(\log \frac{2^{|Y|}}{H} + \log\log \frac{2^{|Y|}}{H} + \rho)\log \frac{2^{|Y|}}{H}} - 1$, which has a singularity at $\rho = -\log\log \frac{2^{|Y|}}{H} - \log\log \frac{2^{|Y|}}{H}$, and otherwise has a critical point at $\rho = \frac{\ln 2}{\log\log \frac{2^{|Y|}}{H}} - \log \frac{2^{|Y|}}{H} - \log\log \frac{2^{|Y|}}{H}$. Since $\log \frac{2^{|Y|}}{H} \geq 2$ and $\log\log \frac{2^{|Y|}}{H} \geq 1$ here, these are both clearly negative.

Now, we'll show that this expression (for $|Y| \geq 2$) is maximized when $H = 1$. Observe first that the expression $H(|Y| + \log \frac{1}{H})$ as a function of H does not have critical points for $H \leq 1$: the derivative is $|Y| + \log \frac{1}{H} - \frac{1}{\ln 2}$, so critical points require $H = 2^{|Y| - (1/\ln 2)} > 1$. Hence we see that this expression is maximized at the boundary, when $H = 1$. Similarly, the rest of the expression,

$$H \log(|Y| + \log \frac{1}{H} + \log(|Y| + \log \frac{1}{H}) + 2.5)$$

viewed as a function of H, only has critical points for

$$\log(|Y| + \log \frac{1}{H} + \log(|Y| + \log \frac{1}{H}) + 2.5) = \frac{\frac{1}{\ln 2}(1 + \frac{1}{|Y| + \log \frac{1}{H}})}{|Y| + \log \frac{1}{H} + \log(|Y| + \log \frac{1}{H}) + 2.5}$$

i.e., it requires

$$(|Y| + \log \frac{1}{H} + \log(|Y| + \log \frac{1}{H}) + 2.5)\log(|Y| + \log \frac{1}{H} + \log(|Y| + \log \frac{1}{H}) + 2.5) = \frac{1}{\ln 2}(1 + \frac{1}{|Y| + \log \frac{1}{H}}).$$

But, the right-hand side is at most $\frac{3}{2\ln 2} < 3$, while the left-hand side is at least 13. Thus, it also has no critical points, and its maximum is similarly taken at the boundary, $H = 1$. Thus, overall, when $H \leq 1$ and $|Y| \geq 2$ we find

$$\sum_{\sigma \in 2^Y} p_\sigma (\log p_\sigma)^2 \leq |Y| + \log(|Y| + \log|Y| + 2.5).$$

Although the assignment of probability mass used in the bound did not sum to 1, nevertheless this bound is nearly tight. For any $\gamma > 0$, and letting $H = 1 + \Delta$ where $\Delta = \frac{1}{\log^\gamma(2^{|Y|} - 2)}$, the following solution attains a ratio of $(1 - o(1))|Y|^{1-\gamma}$: for any two $\sigma_1^*, \sigma_2^* \in 2^Y$, set $p_{\sigma_i^*} = \frac{1}{2} - \frac{\epsilon}{2}$ and set the rest to $\frac{\epsilon}{2^{|Y|} - 2}$, for ϵ chosen

below. To obtain

$$H = 2 \cdot (\frac{1}{2} - \frac{\epsilon}{2}) \log \frac{2}{1-\epsilon} + (2^{|Y|} - 2) \cdot \frac{\epsilon}{2^{|Y|} - 2} \log \frac{2^{|Y|} - 2}{\epsilon}$$
$$= (1-\epsilon)(1 + \log(1 + \frac{\epsilon}{1-\epsilon})) + \epsilon \log \frac{2^{|Y|} - 2}{\epsilon}$$

observe that since $\log(1+x) = \frac{x}{\ln 2} + \Theta(x^2)$, we will need to take

$$\epsilon = \frac{\Delta}{\log(2^{|Y|} - 2) + \log \frac{1-\epsilon}{\epsilon} - (1 + \frac{1}{\ln 2}) + \Theta(\epsilon^2)}$$
$$= \frac{\Delta}{\log(2^{|Y|} - 2) + \log\log(2^{|Y|} - 2) + \log \frac{1}{\Delta} - (1 + \frac{1}{\ln 2}) - \frac{\epsilon}{\ln 2} + \Theta(\epsilon^2)}.$$

For such a choice, we indeed obtain the ratio

$$\frac{(1-\epsilon)\log^2 \frac{2}{1-\epsilon} + \epsilon \log^2 \frac{(2^{|Y|}-2)}{\epsilon}}{H^2} \geq (1 - o(1))|Y|^{1-\gamma}.$$

Using these bounds, we are finally ready to prove Theorem 1:

Proof. We first consider the case where no $\sigma \in sol(\varphi)$ has $p_\sigma > 1/2$; here, the condition in line 6 of EntropyEstimation never passes, so we return the value obtained by SampleEst on line 12. Note that we must have $H_\varphi(Y) \geq 1$ in this case. So, by Lemma 3,

$$\frac{\sum_{\sigma \in 2^Y} p_\sigma (\log p_\sigma)^2}{\left(\sum_{\sigma \in 2^Y} p_\sigma \log \frac{1}{p_\sigma}\right)^2} \leq \min\left\{|X|, \left(1 + \frac{\log(|Y| + \log|Y| + 1.1)}{|Y|}\right)|Y|\right\}$$

and hence, by Lemma 1, using $t \geq \frac{6 \cdot \min\{|X|, |Y| + \log(|Y| + \log|Y| + 1.1)\} - 1)}{\varepsilon^2}$ suffices to ensure that the returned $\hat{h}$ is satisfactory with probability $1 - \delta$.

Next, we consider the case where some $\sigma^* \in sol(\varphi)$ has $p_{\sigma^*} > 1/2$. Since the total probability is 1, there can be at most one such σ^*. So, in the distribution conditioned on $\sigma \neq \sigma^*$, i.e., $\{p'_\sigma\}_{\sigma \in 2^Y}$ that sets $p'_{\sigma^*} = 0$, and $p'_\sigma = \frac{p_\sigma}{1 - p_{\sigma^*}}$ otherwise, we now need to show that t satisfies

$$\frac{1}{t\varepsilon^2}\left(\frac{\sum_{\sigma \neq \sigma^*} p'_\sigma (\log \frac{1}{(1-p_{\sigma^*})p'_\sigma})^2}{(\sum_{\sigma \neq \sigma^*} p'_\sigma \log \frac{1}{(1-p_{\sigma^*})p'_\sigma})^2} - 1\right) < \frac{1}{6}$$

to apply Lemma 1. We first rewrite this expression. Letting $H = \sum_{\sigma \neq \sigma^*} p'_\sigma \log \frac{1}{p'_\sigma}$ be the entropy of this conditional distribution,

$$\frac{\sum_{\sigma \neq \sigma^*} p'_\sigma (\log \frac{1}{(1-p_{\sigma^*})p'_\sigma})^2}{(\sum_{\sigma \neq \sigma^*} p'_\sigma \log \frac{1}{(1-p_{\sigma^*})p'_\sigma})^2} = \frac{\sum_{\sigma \neq \sigma^*} p'_\sigma (\log \frac{1}{p'_\sigma})^2 + 2H \log \frac{1}{1-p_{\sigma^*}} + (\log \frac{1}{1-p_{\sigma^*}})^2}{(H + \log \frac{1}{1-p_{\sigma^*}})^2}$$
$$= \frac{\sum_{\sigma \neq \sigma^*} p'_\sigma (\log \frac{1}{p'_\sigma})^2 - H^2}{(H + \log \frac{1}{1-p_{\sigma^*}})^2} + 1.$$

Lemma 2 now gives rather directly that this quantity is at most

$$\frac{H|X| - H^2}{(H + \log \frac{1}{1-p_{\sigma^*}})^2} + 1 < \frac{|X|}{2 \log \frac{1}{1-p_{\sigma^*}}} + 1.$$

For the bound in terms of $|Y|$, there are now two cases depending on whether H is greater than 1 or less than 1. When it is greater than 1, the first part of Lemma 3 again gives

$$\frac{\sum_{\sigma \in 2^Y} p'_\sigma (\log p'_\sigma)^2}{H^2} \leq |Y| + \log(|Y| + \log|Y| + 1.1).$$

When $H < 1$, on the other hand, recalling $p_{\sigma^*} > 1/2$ (so $\log \frac{1}{1-p_{\sigma^*}} \geq 1$), the second part of Lemma 3 gives that our expression is less than

$$\frac{|Y| + \log(|Y| + \log|Y| + 2.5)) - H^2}{(H + \log \frac{1}{1-p_{\sigma^*}})^2} < |Y| + \log(|Y| + \log|Y| + 2.5).$$

Thus, by Lemma 1,

$$t \geq \frac{6 \cdot \min\{\frac{|X|}{2 \log \frac{1}{1-p_{\sigma^*}}}, |Y| + \log(|Y| + \log|Y| + 2.5)\}}{\varepsilon^2}$$

suffices to obtain $\hat{h}$ such that $\hat{h} \leq (1+\varepsilon) \sum_{\sigma \neq \sigma^*} \frac{p_\sigma}{1-p_{\sigma^*}} \log \frac{1}{p_\sigma}$ and $\hat{h} \geq (1-\varepsilon) \sum_{\sigma \neq \sigma^*} \frac{p_\sigma}{1-p_{\sigma^*}} \log \frac{1}{p_\sigma}$; hence we obtain such a $\hat{h}$ with probability at least $1 - 0.9 \cdot \delta$ in line 10, if we pass the test on line 6 of Algorithm 1, thus identifying σ^*. Note that this value is adequate, so we need only guarantee that the test on line 6 passes on one of the iterations with probability at least $1 - 0.1 \cdot \delta$.

To this end, note that each sample$(\tau_{\downarrow Y})$ on line 4 is equal to σ^* with probability $\frac{|sol(\varphi(Y \mapsto \sigma^*))|}{|sol(\varphi)_{\downarrow X}|} > \frac{1}{2}$ by hypothesis. Since each iteration of the loop is an independent draw, the probability that the condition on line 6 is not met after $\log \frac{10}{\delta}$ draws is less than $(1 - \frac{1}{2})^{\log \frac{10}{\delta}} = \frac{\delta}{10}$, as needed.

4.4 Beyond Boolean Formulas

We now focus on the case where the relationship between X and Y is modeled by an arbitrary relation $\mathcal{R}$ instead of a Boolean formula φ. As noted in Section 1, program behaviors are often modeled with other representations such as automata [4,5,14]. The automata-based modeling often has X represented as the input to the given automaton $\mathcal{A}$ while every realization of Y corresponds to a state of $\mathcal{A}$. Instead of an explicit description of $\mathcal{A}$, one can rely on a symbolic description of $\mathcal{A}$. Two families of techniques are currently used to estimate the entropy. The first technique is to enumerate the possible *output* states and, for each such state s, estimate the number of strings accepted by $\mathcal{A}$ if s was the only accepting state of $\mathcal{A}$. The other technique relies on uniformly sampling a string σ,

noting the final state of $\mathcal{A}$ when run on σ, and then applying a histogram-based technique to estimate the entropy.

In order to use the algorithm EntropyEstimation one requires access to a sampler and model counter for automata; the past few years have witnessed the design of efficient counters for automata to handle string constraints. In addition, EntropyEstimation requires access to a conditioning routine to implement the substitution step, i.e., $Y \mapsto \tau_{\downarrow Y}$, which is easy to accomplish for automata via marking the corresponding state as a non-accepting state.

5 Empirical Evaluation

To evaluate the runtime performance of EntropyEstimation, we implemented a prototype in Python that employs SPUR [3] as a uniform sampler and GANAK [52] as a projected model counter. We experimented with 96 Boolean formulas arising from diverse applications ranging from QIF benchmarks [32], plan recognition [54], bit-blasted versions of SMTLIB benchmarks [54,52], and QBFEval competitions [1,2]. The value of $n = |X|$ varies from 5 to 752 while the value of $m = |Y|$ varies from 9 to 1447.

In all of our experiments, the parameters δ and ε were set to 0.09, 0.8 respectively. All of our experiments were conducted on a high-performance computer cluster with each node consisting of a E5-2690 v3 CPU with 24 cores, and 96GB of RAM with a memory limit set to 4GB per core. Experiments were run in single-threaded mode on a single core with a timeout of 3000 seconds.

Baseline: As our baseline, we implemented the following approach to compute the entropy exactly, which is representative of the current state of the art approaches [13,27,39][5]. For each valuation $\sigma \in sol(\varphi)_{\downarrow Y}$, we compute $p_\sigma = \frac{|sol(\varphi(Y \mapsto \sigma))|}{|sol(\varphi)_{\downarrow X}|}$, where $|sol(\varphi(Y \to \sigma))|$ is the count of satisfying assignments of $\varphi(Y \mapsto \sigma)$, and $|sol(\varphi)_{\downarrow X}|$ represents the projected model count of φ over X. Then, finally the entropy is computed as $\sum_{\sigma \in 2^Y} p_\sigma \log(\frac{1}{p_\sigma})$.

Our evaluation demonstrates that EntropyEstimation can scale to the formulas beyond the reach of the enumeration-based baseline approach. Within a given timeout of 3000 seconds, EntropyEstimation is able to estimate the entropy for all the benchmarks, whereas the baseline approach could terminate only for 14 benchmarks. Furthermore, EntropyEstimation estimated the entropy within the allowed tolerance for *all* the benchmarks.

5.1 Scalability of EntropyEstimation

Table 1 presents the performance of EntropyEstimation vis-a-vis the baseline approach for 20 benchmarks.[6] Column 1 of Table 1 gives the names of the bench-

[5] We wish to emphasize that none of the previous approaches could provide theoretical guarantees of (ε, δ) without enumerating over all possible assignments to Y.

[6] The complete analysis for all of the benchmarks is deferred to the technical report https://arxiv.org/pdf/2206.00921.pdf.

Benchmarks	$\|X\|$	$\|Y\|$	Baseline		EntropyEstimation	
			Time(s)	count queries	Time(s)	count/sample queries
pwd-backdoor	336	64	-	1.84×10^{19}	5.41	1.25×10^{2}
case31	13	40	201.02	1.02×10^{3}	125.36	5.65×10^{2}
case23	14	63	420.85	2.05×10^{3}	141.17	6.10×10^{2}
s1488_15_7	14	927	1037.71	3.84×10^{3}	150.29	6.10×10^{2}
bug1-fix-4	53	17	373.52	1.76×10^{3}	212.37	9.60×10^{2}
s832a_15_7	23	670	-	2.65×10^{6}	247	1.04×10^{3}
dyn-fix-1	40	48	-	3.30×10^{4}	252.2	1.83×10^{3}
s1196a_7_4	32	676	-	4.22×10^{7}	343.68	1.46×10^{3}
backdoor-2x16	168	32	-	1.31×10^{5}	405.7	1.70×10^{3}
CVE-2007	752	32	-	4.29×10^{9}	654.54	1.70×10^{3}
subtraction32	65	218	-	1.84×10^{19}	860.88	3.00×10^{3}
case_1_b11_1	48	292	-	2.75×10^{11}	1164.36	2.20×10^{3}
s420_15_7-1	235	116	-	3.52×10^{7}	1187.23	5.72×10^{3}
case145	64	155	-	7.04×10^{13}	1243.11	2.96×10^{3}
floor64-1	405	161	-	2.32×10^{27}	1764.2	7.85×10^{3}
s641_7_4	54	453	-	1.74×10^{12}	1849.84	2.48×10^{3}
decomp64	381	191	-	6.81×10^{38}	2239.62	9.26×10^{3}
squaring2	72	813	-	6.87×10^{10}	2348.6	3.33×10^{3}
stmt5_731_730	379	311	-	3.49×10^{10}	2814.58	1.49×10^{4}

Table 1: "-" represents that entropy could not be estimated due to timeout. Note that $m = |Y|$ and $n = |X|$.

marks, while columns 2 and 3 list the numbers of X and Y variables. Columns 4 and 5 respectively present the time taken, number of samples used by baseline approach, and columns 6 and 7 present the same for EntropyEstimation. The required number of samples for the baseline approach is $|sol(\varphi)_{\downarrow Y}|$.

Table 1 clearly demonstrates that EntropyEstimation outperforms the baseline approach. As shown in Table 1, there are some benchmarks for which the projected model count on V is greater than 10^{30}, i.e., the baseline approach would need 10^{30} valuations to compute the entropy exactly. By contrast, the proposed algorithm EntropyEstimation needed at most $\sim 10^4$ samples to estimate the entropy within the given tolerance and confidence. The number of samples required to estimate the entropy is reduced significantly with our proposed approach, making it scalable.

5.2 Quality of Estimates

There were only 14 benchmarks out of 96 for which the enumeration-based baseline approach finished within a given timeout of 3000 seconds. Therefore, we compared the entropy estimated by EntropyEstimation with the baseline for those 14

benchmarks only. Figure 1 shows how accurate were the estimates of the entropy by EntropyEstimation. The y-axis represents the observed error, which was calculated as $max(\frac{\text{Estimated}}{\text{Exact}} - 1, \frac{\text{Exact}}{\text{Estimated}} - 1)$, and the x-axis represents the benchmarks ordered in ascending order of observed error; that is, a bar at x represents the observed error for a benchmark—the lower, the better.

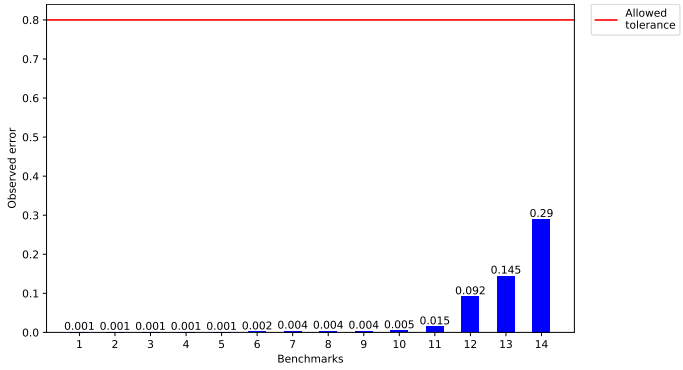

Fig. 1: The accuracy of estimated entropy using EntropyEstimation for 14 benchmarks. $\varepsilon = 0.8, \delta = 0.09$.

The red horizontal line in Figure 1 indicates the maximum allowed tolerance (ε), which was set to 0.80 in our experiments. We observe that for *all* 14 benchmarks, EntropyEstimation estimated the entropy within the allowed tolerance; in fact, the observed error was greater than 0.1 for just 2 out of the 14 benchmarks, and the maximum error observed was 0.29.

Alternative Baselines: As we discussed earlier, several other algorithms have been proposed for estimating the entropy. For example, Valiant and Valiant's algorithm [58] obtains an ε-additive approximation using $\mathcal{O}(\frac{2^m}{\varepsilon^2 m})$ samples, and Chakraborty et al. [17] compute such approximations using $\mathcal{O}(\frac{m^7}{\varepsilon^8})$ samples. We stress that neither of these is exact, and thus could not be used to assess the accuracy of our method as presented in Figure 1. Moreover, based on Table 1, we observe that the number of sampling or counting calls that could be computed within the timeout was roughly 2×10^4, where m ranges between 10^1–10^3. Thus, the method of Chakraborty et al. [17], which would take 10^7 or more samples on all benchmarks, would not be competitive with our method, which never used 2×10^4 calls. The method of Valiant and Valiant, on the other hand, would likely allow a few more benchmarks to be estimated (perhaps up to a fifth of the benchmarks). Still, it would not be competitive with our technique except in the smallest benchmarks (for which the baseline required $< 10^6$ samples, about a third of our benchmarks), since we were otherwise more than a factor of m faster than the baseline.

6 Conclusion

In this work, we considered estimating the Shannon entropy of a distribution specified by a circuit formula $\varphi(X, Y)$. Prior work relied on $\mathcal{O}(2^m)$ model counting queries and, therefore, could not scale to instances beyond small values of m. In contrast, we propose a novel technique, called EntropyEstimation, for estimation of entropy that takes advantage of the access to the formula φ via conditioning. EntropyEstimation makes only $\mathcal{O}(\min(m, n))$ model counting and sampling queries, and therefore scales significantly better than the prior approaches.

Acknowledgments: This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme[NRF-NRFFAI1-2019-0004], Ministry of Education Singapore Tier 2 grant [MOE-T2EP20121-0011], NUS ODPRT grant [R-252-000-685-13], an Amazon Research Award, and NSF awards IIS-1908287, IIS-1939677, and IIS-1942336. We are grateful to the anonymous reviewers for constructive comments to improve the paper. The computational work was performed on resources of the National Supercomputing Centre, Singapore: https://www.nscc.sg.

References

1. QBF solver evaluation portal 2017, http://www.qbflib.org/qbfeval17.php
2. QBF solver evaluation portal 2018, http://www.qbflib.org/qbfeval18.php
3. Achlioptas, D., Hammoudeh, Z., Theodoropoulos, P.: Fast sampling of perfectly uniform satisfying assignments. In: Proc. of SAT (2018)
4. Aydin, A., Bang, L., Bultan, T.: Automata-based model counting for string constraints. In: Proc. of CAV. pp. 255–272. Springer (2015)
5. Aydin, A., Eiers, W., Bang, L., Brennan, T., Gavrilov, M., Bultan, T., Yu, F.: Parameterized model counting for string and numeric constraints. In: Prc. of ESEC/FSE. pp. 400–410 (2018)
6. Aziz, R.A., Chu, G., Muise, C., Stuckey, P.: exists sat: Projected model counting. In: Proc. of SAT. pp. 121–137. Springer (2015)
7. Backes, M., Köpf, B., Rybalchenko, A.: Automatic discovery and quantification of information leaks. In: Proc. of SP (2009)
8. Bang, L., Aydin, A., Phan, Q.S., Păsăreanu, C.S., Bultan, T.: String analysis for side channels with segmented oracles. In: Proc. of SIGSOFT (2016)
9. Batu, T., Dasgupta, S., Kumar, R., Rubinfeld, R.: The complexity of approximating the entropy. SIAM Journal on Computing **35**(1), 132–150 (2005)
10. Bayardo Jr, R.J., Pehoushek, J.D.: Counting models using connected components. In: AAAI/IAAI. pp. 157–162 (2000)
11. Bevier, W.R., Cohen, R.M., Young, W.D.: Connection policies and controlled interference. In: Proc. of CSF (1995)
12. Birnbaum, E., Lozinskii, E.L.: The good old davis-putnam procedure helps counting models. Journal of Artificial Intelligence Research **10**, 457–477 (1999)
13. Borges, M., Phan, Q.S., Filieri, A., Păsăreanu, C.S.: Model-counting approaches for nonlinear numerical constraints. In: NASA Formal Methods Symposium. pp. 131–138. Springer (2017)
14. Bultan, T.: Quantifying information leakage using model counting constraint solvers. In: Working Conference on Verified Software: Theories, Tools, and Experiments. pp. 30–35. Springer (2019)

15. Canonne, C.L.: A survey on distribution testing: Your data is big. but is it blue? Theory of Computing pp. 1–100 (2020)
16. Cernỳ, P., Chatterjee, K., Henzinger, T.A.: The complexity of quantitative information flow problems. In: Proc. of CSF (2011)
17. Chakraborty, S., Fischer, E., Goldhirsh, Y., Matsliah, A.: On the power of conditional samples in distribution testing. SIAM Journal on Computing (2016)
18. Chakraborty, S., Fremont, D.J., Meel, K.S., Seshia, S.A., Vardi, M.Y.: Distribution-aware sampling and weighted model counting for SAT. In: AAAI. pp. 1722–1730. AAAI Press (2014)
19. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Balancing scalability and uniformity in SAT witness generator. In: Proc. of DAC (2014)
20. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic sat calls. In: IJCAI (2016)
21. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-channel leaks in web applications: A reality today, a challenge tomorrow. In: Proc. of SP (2010)
22. Cherubin, G., Chatzikokolakis, K., Palamidessi, C.: F-BLEAU: fast black-box leakage estimation. In: Proc. of SP (2019)
23. Chothia, T., Kawamoto, Y., Novakovic, C.: Leakwatch: Estimating information leakage from java programs. In: Proc. of ESORICS (2014)
24. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. Journal of Computer Security (2007)
25. Darwiche, A.: On the tractable counting of theory models and its application to truth maintenance and belief revision. Journal of Applied Non-Classical Logics **11**(1-2), 11–34 (2001)
26. Denning, D.E.: A lattice model of secure information flow. Communications of the ACM (1976)
27. Eiers, W., Saha, S., Brennan, T., Bultan, T.: Subformula caching for model counting and quantitative program analysis. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 453–464. IEEE (2019)
28. Eldib, H., Wang, C., Schaumont, P.: Formal verification of software countermeasures against side-channel attacks. ACM Transactions on Software Engineering and Methodology (TOSEM) **24**(2), 1–24 (2014)
29. Eldib, H., Wang, C., Taha, M., Schaumont, P.: Qms: Evaluating the side-channel resistance of masked software from source code. In: 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2014)
30. Ferrari, E., Samarati, P., Bertino, E., Jajodia, S.: Providing flexibility in information flow control for object oriented systems. In: Proc. of SP (1997)
31. Fichte, J.K., Hecher, M., Hamiti, F.: The model counting competition 2020. arXiv preprint arXiv:2012.01323 (2020), https://arxiv.org/pdf/2012.01323.pdf
32. Fremont, D., Rabe, M., Seshia, S.: Maximum model counting. In: Proc. of AAAI (2017)
33. Gao, P., Zhang, J., Song, F., Wang, C.: Verifying and quantifying side-channel resistance of masked software implementations. ACM Transactions on Software Engineering and Methodology (TOSEM) **28**(3), 1–32 (2019)
34. Goldreich, O., Vadhan, S.: Comparing entropies in statistical zero knowledge with applications to the structure of szk. In: Proc. of CCC. pp. 54–73. IEEE (1999)
35. Guha, S., McGregor, A., Venkatasubramanian, S.: Sublinear estimation of entropy and information distances. ACM Transactions on Algorithms (TALG) **5**(4), 1–16 (2009)

36. Huang, J., Darwiche, A.: The language of search. Journal of Artificial Intelligence Research **29**, 191–219 (2007)
37. Kadron, İ.B., Rosner, N., Bultan, T.: Feedback-driven side-channel analysis for networked applications. In: Proc. of SIGSOFT (2020)
38. Kim, S., McCamant, S.: Bit-vector model counting using statistical estimation. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 133–151. Springer (2018)
39. Klebanov, V.: Precise quantitative information flow analysis using symbolic model counting. QASA (2012)
40. Köpf, B., Basin, D.: An information-theoretic model for adaptive side-channel attacks. In: Proc. of CCS (2007)
41. Köpf, B., Rybalchenko, A.: Approximation and randomization for quantitative information-flow analysis. In: Proc. of CSF. pp. 3–14. IEEE (2010)
42. Lagniez, J.M., Marquis, P.: A recursive algorithm for projected model counting. In: Proc. of AAAI. vol. 33, pp. 1536–1543 (2019)
43. Ma, S.k.: Calculation of entropy from data of motion. Journal of Statistical Physics **26**(2), 221–240 (1981)
44. Meng, Z., Smith, G.: Calculating bounds on information leakage using two-bit patterns. In: Proc. of PLAS (2011)
45. Möhle, S., Biere, A.: Dualizing projected model counting. In: Proc. of ICTAI. pp. 702–709. IEEE (2018)
46. Phan, Q.S., Bang, L., Pasareanu, C.S., Malacaria, P., Bultan, T.: Synthesis of adaptive side-channel attacks. In: Proc. of CSF (2017)
47. Phan, Q.S., Malacaria, P.: Abstract model counting: a novel approach for quantification of information leaks. In: Proc. of CCS (2014)
48. Phan, Q.S., Malacaria, P., Tkachuk, O., Păsăreanu, C.S.: Symbolic quantitative information flow. Proc. of ACM SIGSOFT (2012)
49. Rosner, N., Kadron, I.B., Bang, L., Bultan, T.: Profit: Detecting and quantifying side channels in networked applications. In: Proc. of NDSS (2019)
50. Sang, T., Bacchus, F., Beame, P., Kautz, H.A., Pitassi, T.: Combining component caching and clause learning for effective model counting. SAT **4**, 7th (2004)
51. Sharma, S., Gupta, R., Roy, S., Meel, K.S.: Knowledge compilation meets uniform sampling. In: Proc. of LPAR (2018)
52. Sharma, S., Roy, S., Soos, M., Meel, K.S.: Ganak: A scalable probabilistic exact model counter. In: Proc. IJCAI (2019)
53. Smith, G.: On the foundations of quantitative information flow. In: Proc. of FOSSAC (2009)
54. Soos, M., Gocht, S., Meel, K.S.: Tinted, detached, and lazy CNF-XOR solving and its applications to counting and sampling. In: Proc. of CAV (2020)
55. Strong, S.P., Koberle, R., Van Steveninck, R.R.D.R., Bialek, W.: Entropy and information in neural spike trains. Physical review letters **80**(1), 197 (1998)
56. Thurley, M.: sharpsat–counting models with advanced component caching and implicit bcp. In: International Conference on Theory and Applications of Satisfiability Testing. pp. 424–429. Springer (2006)
57. Val, C.G., Enescu, M.A., Bayless, S., Aiello, W., Hu, A.J.: Precisely measuring quantitative information flow: 10k lines of code and beyond. In: Proc. of EuroS&P. pp. 31–46. IEEE (2016)
58. Valiant, G., Valiant, P.: Estimating the unseen: Improved estimators for entropy and other properties. J. ACM **64**(6), 1–41 (2017)
59. Zhou, Z., Qian, Z., Reiter, M.K., Zhang, Y.: Static evaluation of noninterference using approximate model counting. In: Proc. of SP (2018)