On Testing of Horn Samplers

A Project Report Submitted
in Partial Fulfilment of the Requirements
for the Degree of

Master of Technology
in
Computer Science

by

Uddalok Sarkar
[Roll No. CS2028]

Supervisor

Dr. Sourav Chakraborty

Advanced Computing and Microelectronics Unit
Indian Statistical Institute

to

INDIAN STATISTICAL INSTITUTE
CALCUTTA - 700 018, INDIA
July 6, 2022

DECLARATION

I, **Uddalok Sarkar (Roll No: CS2028)**, hereby declare that, this report entitled **"On Testing of Horn Samplers"** submitted to Indian Statistical Institute Calcutta towards partial requirement of **Master of Technology** in **Computer Science**, is an original work carried out by me under the supervision of **Dr. Sourav Chakraborty** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold the academic ethics and honesty. Whenever an external information or statement or result is used then, that have been duly acknowledged and cited.

Uddalok Sarkar

Calcutta - 700 018 **Uddalok Sarkar**

July 6, 2022

CERTIFICATE

This is to certify that the work contained in this project report entitled **"On Testing of Horn Samplers"** submitted by **Uddalok Sarkar** (**Roll No: CS2028**) to Indian Statistical Institute, Calcutta towards the partial requirement of **Master of Technology** in **Computer Science** has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

[illegible]

Calcutta - 700 018 Dr. Sourav Chakraborty

July 6, 2022 Project Supervisor

ACKNOWLEDGEMENT

I want to extend a sincere and heartfelt obligation towards all the personages without whom the completion of the project was not possible. I express my profound gratitude towards my supervisor, Dr. Sourav Chakraborty, ACMU, Indian Statistical Institute, for his guidance, valuable feedback, and constant encouragement throughout the project. I also thank my colleague Sayantan Sen, whose intuitiveness and insight has been invaluable to the progression of this project, allowing it to mature into the project it is today.

Also I am immensely grateful to Dr. Kuldeep Meel, for his guidance, for providing me access to benchmarks and helping with the access to NSCC Singapore for the computational requirements. Further, I would like to express my gratitude to Yash Pote and Arijit Shaw from MeelGroup, National University of Singapore for their constant supports and cooperative response.

Finally, I would also thank my friends and most importantly my parents. It is because of their encouragement and belief in me that I have been able to come this far.

[illegible]

Calcutta - 700 018 **Uddalok Sarkar**

July 6, 2022

ABSTRACT

Sampling is a fundamental task in machine learning and computer science in general. Of particular interest to computer scientists are the problems of sampling satisfying assignments to a Boolean formula uniformly at random (or according to some specified distribution) from the set of all satisfying assignments of the formula. Despite its importance, due to its inherent difficulty, often heuristic techniques are used to design samplers, which lack theoretical guarantees. Thus, it is crucial to test the correctness of such samplers, that is, whether a sampler is sampling according to the specified distribution. But efficient testers, which are also based on sound theoretical guarantees, are equally hard to obtain. Only recently, Chakraborty and Meel (AAAI 2019) and later Meel, Pote, and Chakraborty (NeurIPS 2020) designed the first efficient and provably correct testers, Barbarik and Barbarik2, for testing the correctness of CNF-samplers. But due to their "grey-box" approach, these testers cannot be used to test the correctness of specialized samplers such as Horn-samplers, which take constraints specified as a Horn formulas a input.

In this thesis work, we design two testers, Flash and wFlash, which test the correctness of a given Horn sampler. The tester Flash correctly (with probability at least $1 - \delta$) tests whether the underlying distribution of a Horn-sampler is "ε-close" to uniform or "η-far" from uniform by sampling only $O(1/(\eta - \varepsilon)^4)$ samples from the Horn-sampler that is being tested. The tester wFlash is a generalization of Flash and tests whether a Horn-sampler is sampling according to a specified distribution (not necessarily uniform). Its sample complexity is $O(tilt^3/(\eta - \varepsilon)^4)$, where the *tilt* is the ratio of the maximum and the minimum (non-zero) probability masses of the specified distribution. We provide prototype implementations of Flash and wFlash and test three state-of-the-art samplers on a set of benchmarks. Thus we present the first such practically usable testers for Horn-samplers, which also have theoretical guarantees.

Keywords: Property Testing, Samplers

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Sampling from complex combinatorial spaces is a fundamental problem in computer science with wide variety of applications. Often combinatorial spaces are specified via constraints expressed in logical theories such that every point in the support corresponds to solutions of the given constraints. The design of efficient algorithms for sampling from complex combinatorial spaces is very difficult. While techniques such as Markov Chain Monte Carlo (MCMC) techniques or those based on universal hashing allow design sampling algorithms that have theoretical guarantees, such algorithms face scalability challenges. As a result, heuristic techniques are often used to sample satisfying assignments. While heuristic techniques often work well in practice and are often devoid of sound theoretical guarantees in general, such techniques may perform well for sceanrios of interest. Consequently, there is a dire need for design of testers that can verify whether a sampler is sampling according to the desired distribution for a given combinatorial space.

Due to the probabilistic nature of the sampling algorithms, designing testers for them is a very difficult task. A tester that uses black-box access to the sampling algorithm would essentially need to test properties of the unknown underlying distribution using samples from the unknown distribution - and this requires an exponential number of samples [5, 3, 6, 31]. Recently, Barbarik was proposed as a provable correct tester when the underlying set of constraints are specified in Conjunctive Normal Form (CNF) [9, 22]. Our main objective is to develop provable correct testers when the underlying set of constraints form a Horn formulae.

1.1 Verification of Samplers

In the field of sampler testing, the main goal is to test whether the distribution induced by the satisfying assignments of the sampler to be tested is *similar* to the distribution of the satisfying assignments of the ideal sampler at hand. It turns out that this problem has been extensively studied in the sub-field of distribution testing, and in property testing in general.

The primary goal is to decide whether two distributions D_1 and D_2 are ε-close or η-far, where ε and η are the closeness and farness parameters respectively, provided as inputs. Generally, the distance measure that is mostly considered is the ℓ_1 distance (or variation distance). This problem is termed as the *tolerant testing* of two distributions. It turns out that $\Theta(\frac{N}{\log N})$ samples are required [30] for this problem in general, where N denotes the support size of the distributions to be tested.

A restricted problem called *equivalence testing*, where the goal is to decide whether two distributions are same or they are η-far also requires $\Theta(N^{\frac{2}{3}})$ many samples [4, 32]. Moreover, the basic problem of testing whether a distribution is uniform takes $\Theta(\sqrt{N})$ samples [24, 20]. However, in case of samplers, as the size N of the support of the distributions over the satisfying assignments of the samplers is extremely large, these algorithms are not practical to use.

In order to remedy this problem, a new model termed as *conditional sampling* model was introduced by Chakraborty et. al [8] and Cannone et. al [7]. In this setting, given a set $T \subseteq [N]$, the sampler obtains a sample $i \in T$ from the distribution, conditioned on the set T. Surprisingly, several interesting problems which are difficult in the normal setting, can be solved very easily in this model, often using poly-logarithmic or even constant number of samples. In fact, the task of testing whether a distribution is uniform can be decided by taking only constant number of samples here, depending only on the input parameters. Moreover, the tolerant testing of the equivalence of two distributions can be done by using samples that is polynomial in $\log N$.

Chakraborty and Meel [9] crucially used the framework of conditional sampling in order to design the first tester to test whether the satisfying assignments produced by a sampler is uniform or not. Their tester $\mathsf{Barbarik}$ is designed to test samplers for CNF formulas, and takes only $\widetilde{\mathcal{O}}(\frac{1}{(\eta-2\varepsilon)^4})$ many samples, where ε and η are the closeness and farness parameters respectively. In fact, they employed a variant of conditional sampling PCOND, defined in Cannone et.al [7], where the size of the conditioning set T is 2, and used the technique of chain-formula to generate conditional samples. However, it turns out the the sample complexity of tolerant testing of closeness of distributions will remain polynomial in $\log N$, even with PCOND model.

In order to bypass this poly-logarithmic dependency on N, the authors of [9] used two different distance measures for the closeness and farness testing. Namely, they used multiplicative ℓ_∞ distance for the closeness measure, where as they employed ℓ_1 distance for the farness case. Later Meel, Pote and Chakraborty [22] used similar techniques in order to design the tester $\mathsf{Barbarik2}$ for testing weighted samplers for CNF formulas. $\mathsf{Barbarik2}$ takes only $\widetilde{\mathcal{O}}(\frac{\mathsf{tilt}^2}{\eta(\eta-6\varepsilon)^3})$ many samples, where tilt refers to the maximum ratio between the weights of any two satisfying assignments of the Horn formula φ. Very recently, Pote and Meel [25] studied the problem of equivalence testing in the context of testing probabilistic circuits.

1.2 Importance of Horn Formulae

CNF is a widely used form to represent Boolean formulas owing to its expressiveness. In particular, every Boolean formula can be expressed in CNF with a linear blow-up in the size. Such expressiveness, however, comes at the cost of computational complexity: even determining satisfiability of CNF formulas is NP-hard. Accordingly, symbolic reasoning community have explored restricted fragments of Boolean formulas that have tractable complexity. Of one such fragment are Horn formulas, which play a vital role in logical systems, and form the backbone of most logic programming languages, such as `Prolog`. Horn formulas have been used for automated theorem proving by using first-order resolution [21], program verification [14], as well as in program analysis for intermediate representation and transformations [19]. Horn formulas are also used to model several real-life topological systems, such as power transmission lines, water and gas supply lines, and telecommunications, for example in [16].

1.3 Horn Sampler and Testing

While the satisfiability of Horn formulas is in P (thanks to the Schaefer's dichotomy theorem [26]), its counting variant is $\#\mathsf{P}$-complete. Since weighted model counting is equivalent to sampling, there has been an ongoing demand for designing efficient Horn samplers that, given a Horn formula φ, can output random satisfying assignments of φ efficiently. As in the case of CNF-samplers, we have some algorithms based on MCMC or hashing methods that have theoretical guarantees but are slow in practice. On the other hand, there are heuristic sampling algorithms that work well in practice, but are devoid of sound theoretical guarantees. In this thesis, we design efficient testers for testing the correctness of Horn-samplers. When the sampler is supposed to uniformly sample from the satisfying assignments of φ, we call them *Uniform-Horn-sampler*. For the general version, when the sampler is supposed to sample according to some specified distribution over the set of satisfying assignments, we will refer them as *Weighted-Horn-sampler*. We design and present two testers Flash and wFlash for testing the correctness of Uniform-Horn-sampler and Weighted-Horn-sampler, respectively.

We would now like to emphasize the fact that the testers Barbarik [9] and Barbarik2 [22] do not provide us testers for Horn samplers. This is due to the inner workings of the testers, both of which are based on the "grey-box" sampling technique of drawing samples from a conditional distribution. To obtain these conditional samples, the testers Barbarik and Barbarik2 create a new formula $\hat{\varphi}$ and draws samples by running the sampler over $\hat{\varphi}$. They prove the correctness of their testers under suitable assumptions. Vaguely speaking, they assume that the behavior of the sampler (under test) would remain somewhat unchanged whether it is given the formula φ or $\hat{\varphi}$ as the input. In the case of Horn-samplers, this assumption is not valid (for their testers) as the formula $\hat{\varphi}$ that they create from original φ as a purpose of conditioning, might not be a Horn formula, although the formula φ was Horn. This is a major stumbling block for using the testers Barbarik and Barbarik2 for testing Horn-samplers. In this context, it is worth asking: *whether it is possible to circumvent the above stumbling block and design testers for Horn-samplers?*

1.4 Contribution of this thesis

The primary contribution of this thesis is to answer the above question affirmatively. In particular, we design two testers, Flash and wFlash , that can test uniform and weighted

Horn samplers. Before this work, *Shayak Chakraborty* established a primary work by developing the algorithm of Flash for Uniform-Horn-sampler-tester in his thesis work in 2019. But there were some gaps in the proofs which we close in this thesis and reduce down the complexity from previous version of Flash developed by Shayak. Also, we extend the work to develop a Horn Sampler Tester for general case, that is, a Weighted-Horn-sampler-tester.

- We prove that if the underlying distribution from which the Horn-sampler under test draws samples is ε-close (in multiplicative ℓ_∞ norm) to the uniform distribution, then our tester Flash will ACCEPT with probability at least $(1-\delta)$. On the other hand, if the underlying distribution is η-far (in the ℓ_1 norm) from the uniform distribution, then our tester Flash will REJECT with probability at least $(1-\delta)$, assuming the sampler satisfies some conditions. Our tester Flash draws $\mathcal{O}(1/(\eta-\varepsilon)^4)$ samples from the underlying distribution.
- We extend our tester Flash , and design another novel tester wFlash for Weighted-Horn-samplers, that given any arbitrary but fixed weight function wt, can test whether the distribution induced by the satisfying assignments produced by the Horn-sampler follows the weight function wt. As in the case of Flash , we also provide a detailed proof of correctness of wFlash .
- We further provide a prototype implementation of Flash and wFlash and the empirical results over three state-of-the-art samplers on a set of benchmarks. Our empirical evaluation shows that we achieve over 10^7-factor speedup over baseline approach.

Organization of the Thesis: In Chapter 2, we formally define the necessary terminology of our work. In Chapter 3, we present our Uniform-Horn-sampler-tester Flash . In this chapter, we have described our prime constitutional subroutines of Flash and wFlash , that is, HornKernel and Encode . And subsequently in the next Chapter 4 we present our Weighted-Horn-sampler-tester wFlash . We show the evaluation results of the prototype implementations of our testers Flash and wFlash with respect to three sate-of-the-art samplers UniGen, QuickSampler and STS in Chapter 5. Finally, we conclude in Chapter 6.

Chapter 2

Definitions and Preliminaries

In this chapter, we present some preliminaries that are relevant to the contributions of this thesis. We begin with a discussion on the general concept of sampling, and move on to describe relevant work on samplers and their verifiers. Further, we present an overview of the conditioning method.

2.1 The Problem of Sampling

In probability theory, we define a sample space to be the set of all possible outcomes of a given experiment. Once a distribution is defined over the sample space, the probability of outcome of a certain sample is fixed by the distribution. A sampler, for us, is a randomized algorithm that emulates a distribution over the sample space and generates or outputs a sample according to its distribution. In this thesis, we use the terms sampler and generator interchangeably.

In our setup, we are dealing with SAT-samplers. For a SAT-sampler, given a formula φ, the sample space is the set of all witnesses R_φ, and in each call the sampler outputs a satisfying assignment that is a witness of φ according to some distribution.

Definition 2.1.1 (Sampler)**.** Given a problem instance E and a distribution defined D_{Ω_E} defined over the sample space Ω_E, a Sampler $\mathcal{G}$ is a randomized algorithm that generates elements from Ω_E according to the distribution D_{Ω_E}, that is,

$$\forall x \in \Omega_E, \mathbb{P}[\mathcal{G}(E) = x] = \mathbb{P}_{D_{\Omega_E}}[x]$$

Simply putting, a sampler is like a blackbox, which on request generates a given number of samples from the sample space according to some distribution.

2.1.1 Horn Sampler

As we are mostly interested in SAT samplers, rather Horn Samplers in particular, we will define some notions related to Horn Samplers and samplers in general, needed further in this thesis.

Definition 2.1.2 (Weight function)**.** Let S be a set of Boolean variables. A weight function $wt : \{0,1\}^{|S|} \rightarrow (0,1)$ assigns weight to each assignment that can be formed using the set S.

Definition 2.1.3 (tilt)**.** Consider a Horn formula φ, and the associated arbitrary but fixed weight function wt. We define tilt of φ with respect to wt as $\mathsf{tilt}(\varphi, wt) = \max_{\sigma_1,\sigma_2 \in R_\varphi} \frac{wt(\sigma_1)}{wt(\sigma_2)}$. We will sometime refer $\mathsf{tilt}(\varphi, wt)$ as tilt when it is clear from the context.

Definition 2.1.4 (Weighted-Horn-sampler)**.** A Weighted-Horn-sampler $\mathcal{G}(\varphi, wt, S, \kappa)$ is a randomized algorithm that takes a Horn formula φ, a set of variables $S \subseteq \mathit{Supp}(\varphi)$, a weight function wt and an integer κ, and outputs κ many independent samples from $R_{\varphi\downarrow S}$, the set of satisfying assignments of φ, projected on the set S.

A weight function helps us to define a distribution in practical scenario. When the weight function is uniform, that is, it assigns the same weight to all the witnesses, the Weighted-Horn-sampler is said to be a Uniform-Horn-sampler.

For brevity, we will often write $\mathcal{G}(\varphi, wt, S, \kappa)$ as $\mathcal{G}(\varphi)$ or $\mathcal{G}$, when it is clear from the context. Also, we will write the distribution induced by the samples obtained from $\mathcal{G}$ as $D_{\mathcal{G}(\varphi)}$.

Definition 2.1.5 (Ideal Weighted-Horn-sampler and Ideal Uniform-Horn-sampler)**.** Consider a Horn formula φ, a set of variables $S \subseteq \mathit{Supp}(\varphi)$, and a weight function wt. A Horn sampler $\mathcal{I}_{\mathcal{W}}(\varphi, S, wt)$ is said to be an *ideal Weighted-Horn-sampler* with respect to the weight function wt, if

$$\forall \sigma \in R_{\varphi\downarrow S}, \quad \mathbb{P}\left[\mathcal{I}_{\mathcal{W}}(\varphi, S, 1) = \sigma\right] = \frac{wt(\sigma)}{\sum_{\sigma_1 \in R_{\varphi\downarrow S}} wt(\sigma_1)}$$

When the weight function wt is uniform, that is, $wt(\sigma) = \frac{1}{|R_{\varphi\downarrow S}|}$ for all σ, the ideal Weighted-Horn-sampler is said to be an *ideal Uniform-Horn-sampler*, and is denoted as $\mathcal{I}_{\mathcal{U}}$.

Definition 2.1.6 (ε-closeness and η-farness)**.** Consider any Weighted-Horn-sampler $\mathcal{G}$ and an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$. $\mathcal{G}$ is said to be ε-close to $\mathcal{I}_{\mathcal{W}}$, if for all Horn-formula φ and $\sigma \in R_{\varphi}$ [1],

$$(1-\varepsilon)\mathbb{P}\left[\mathcal{I}_{\mathcal{W}}(\varphi) = \sigma\right] \leq \mathbb{P}\left[\mathcal{G}(\varphi) = \sigma\right] \leq (1+\varepsilon)\mathbb{P}\left[\mathcal{I}_{\mathcal{W}}(\varphi) = \sigma\right]. \qquad (2.1)$$

[1] The definition is given with $S = \mathit{Supp}(\varphi)$. However, similar definition can be defined for any arbitrary S.

If $\mathcal{G}$ is ε-close to the ideal Uniform-Horn-sampler, then $\mathcal{G}$ is called an ε-Additive Almost Uniform-Horn-sampler (AAU).

On the other hand, $\mathcal{G}$ is said to be η-far from $\mathcal{I}_{\mathcal{W}}$ with respect to some Horn-formula φ if

$$\sum_{\sigma \in R_\varphi} \left| \mathbb{P}\left[\mathcal{G}(\varphi) = \sigma\right] - \mathbb{P}\left[\mathcal{I}_{\mathcal{W}}(\varphi) = \sigma\right] \right| \geq \eta. \tag{2.2}$$

Example 2.1.7. Let $\varphi := (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_3)$ be a Horn formula. Then the *witness* set of φ is

$$R_\varphi = \{000, 001, 010, 011, 110\}$$

now, we can define a weight function wt on the witness space R_φ as follows:

$$wt(000) = 0.25$$
$$wt(001) = 0.25$$
$$wt(010) = 0.5$$
$$wt(011) = 0.75$$
$$wt(110) = 0.75$$

Hence, probability that a Ideal Weighted-Horn-sampler sample $\sigma = 000$ is $\frac{0.25}{2.5} = 0.1$. Now, if we incorporate a 0.2-close Weighted-Horn-sampler$\mathcal{G}$ then $\mathbb{P}[\mathcal{G} = 000] \in [0.08, 0.12]$.

2.2 Horn Sampler Verification

Since most of the times the witness space R_φ is too large for enumeration, samplers use many heuristics to generate samples, quality of which do not often have theoretical guarantees. Thus it is necessary to verify the quality of the sample.

Definition 2.2.1 (Horn-sampler-tester)**.** A Horn-sampler-tester takes as input a Weighted-Horn-sampler $\mathcal{G}$, an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, a tolerance parameter $\varepsilon \in (0, 1/3]$, an intolerance parameter $\eta \in (0, 2]$ with $\eta > 3\varepsilon$, a confidence parameter δ, and a Horn formula φ, and:

(1) If $\mathcal{G}$ is ε-close to the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then the tester outputs ACCEPT with probability at least $1 - \delta$.

(2) If $\mathcal{G}$ is η-far from an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$ with respect to φ, then it outputs REJECT with probability at least $1 - \delta$.

Since the Horn-sampler-tester will be have black-box or grey-box access to the Weighted-Horn-sampler $\mathcal{G}$, the tester would start by studying the output of $\mathcal{G}$ on the input φ. In practice, we are not provided with the exact probability mass function the sampler $\mathcal{G}$ following to retrieve a sample. In such situations we need to estimate the probabilities with a certain guarantee, using *maximum likelihood estimation.* In order to determine the number of samples needed to estimate the unknown probability with a certain probabilistic guarantee, we use *Chernoff bounds.* In the next section we explain the notion of Chernoff bounds and in the following subsection we explain how parameter estimation is done using Maximum Likelihood estimation.

2.3 Important Probabilistic Methods

2.3.1 Chernoff Bound

In our work, we use the following four concentration inequalities, see [15].

Lemma 2.3.1 (Chernoff-Hoeffding bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum\limits_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$, the following holds for all $0 \le \delta \le 1$*

$$\mathbb{P}\left(|X - \mu| \ge \delta\mu\right) \le 2\exp\left(\frac{-\mu\delta^2}{3}\right).$$

Lemma 2.3.2 (Chernoff-Hoeffding bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum\limits_{i=1}^{n} X_i$ and $\mu_l \le \mathbb{E}[X] \le \mu_h$, the followings hold for any $\delta > 0$.*

(i) $\mathbb{P}\left(X \ge \mu_h + \delta\right) \le \exp\left(\frac{-2\delta^2}{n}\right)$.

(ii) $\mathbb{P}\left(X \le \mu_l - \delta\right) \le \exp\left(\frac{-2\delta^2}{n}\right)$.

Lemma 2.3.3 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $a_i \le X_i \le b_i$ and $X = \sum\limits_{i=1}^{n} X_i$. Then, for all $\delta > 0$,*

$$\mathbb{P}\left(|X - \mathbb{E}[X]| \ge \delta\right) \le 2\exp\left(\frac{-2\delta^2}{\sum\limits_{i=1}^{n}(b_i - a_i)^2}\right)$$

Lemma 2.3.4. *Let $Z_1, \ldots, Z_n$ be n independent and identically distributed $0-1$ random variable. Then, the following hold:*

(i) If $\mathbb{E}[Z_i] \ge \theta \ge 0$, then for any $t \le \theta$, we have

$$\mathbb{P}\left[\sum_{j\in[n]} \frac{Z_j}{n} \le t\right] \le \exp\left(-\frac{(\theta - t)^2 n}{2\theta}\right)$$

(ii) If $\mathbb{E}[Z_i] \le \theta$, then for any $t \ge \theta$, we have

$$\mathbb{P}\left[\sum_{j\in[n]} \frac{Z_j}{n} \ge t\right] \le \exp\left(-\frac{(t - \theta)^2 n}{2\theta}\right)$$

2.3.2 Maximum Likelihood Estimation

Suppose we are given a probability distribution $\mathcal{D}(\theta)$ with some unknown parameter θ along with a sample configuration $\{x_1, x_2, ..., x_n\}$ drawn according to $\mathcal{D}(\theta)$. Then the likelihood function $L(\theta)$ is defined as,

$$L(\theta) = \mathbb{P}_{\mathcal{D}(\theta)}(x_1, x_2, ..., x_n; \theta)$$

Maximum likelihood estimation tells us to pick the value of θ for which the likelihood function $L(\theta)$ is maximized.

For example, suppose we are to determine the bias of a coin which is tossed n times and n_H many heads are occured. By MLE then we infer that the bias of the coin is n_H/n. Now, surely there will be some error in the estimated bias. We can use some concentration bounds to infer how many times we should toss the coin to bound the error to an additive constant. If the bias of the coin is p then with probablity $e^{\mathcal{O}(\gamma^2 n)}$, the estimate n_H/n is within additive error of γ,

$$\mathbb{P}\left[\left|p - \frac{n_H}{n}\right| \leq \gamma\right] \geq e^{\gamma^2 n/2}$$

We define $M(\gamma)$ to be the number of times we need to toss the coin to estimated the bias upto an additive error of γ with probability $1/2$. Note that, if the coin is tossed $k \times M(\gamma)$ times then the probability of the estimated bias being in the range $[p - \gamma,\, p + \gamma]$ is given by $(1 - (1/2)^k)$.

2.4 Chain Formulas

The notion of chain formula was first introduced in [11]. Chain formulas provide a natural way to construct linear sized Boolean formulas with a precise number of the satisfying assignments. The structure of the chain formula has been exploited after its inception.

Definition 2.4.1 (Chain Formula)**.** A chain formula is defined as follows:

(1) Every literal (a boolean variable or its negation) is a chain formula.

(2) If l is a literal and φ be a chain formula such that neither l nor $\neg l$ appear in φ, then $(l \vee \varphi)$ as well as $(l \wedge \varphi)$ are two chain formulas.

(3) Let $m > 0$ be a natural number and $k < 2^m$ be a positive odd number. Let $c_1 c_2 \ldots c_m$ be the m-bit representation of k, where c_m is the Least Significant Bit (LSB) in the representation of m. For every $j \in \{1, \ldots, m-1\}$, if $c_j = 1$ then C_j is "$\vee$', else if $c_j = 0$, then C_j is "$\wedge$". The chain formula $\psi_{k,m}$ is defined as:

$$\psi_{k,m}(a_1, a_2, \ldots, a_m) = a_1 C_1 (a_2 C_2 (\ldots (a_{m-1} C_{m-1} a_m) \ldots)$$

where $a_1, a_2, \ldots, a_m$ are variables.

Example 2.4.2. For $k = 7$ and $m = 5$, as the binary representation of 7 in 5 bits is 00111, the corresponding chain formula would be $\psi_{k,m}(a_1, a_2, a_3, a_4, a_5) = (a_1 \wedge (a_2 \wedge (a_3 \vee (a_4 \vee a_5))))$. We will omit the variables when it is clear from the context.

Lemma 2.4.3 ([11])**.** *Given a natural number $m > 0$ and $k < 2^m$, there exists CNF-formulas $\psi_{k,m}$ such that the size of $\psi_{k,m}$ is linear in m and $\psi_{k,m}$ has exactly k satisfying assignments.*

Due to this property of chain formulas, they have been used in many contexts since their inception. But the chain formulas constructed in Lemma 2.4.3 are not necessarily Horn - a property that we consider in this work. So, we extend the above lemma to obtain *Horn-chain-formulas.*

2.4.1 Horn Chain Formula

We define a special class of Chain formula as designed in [11]. Given a Chain formula $\psi_{k,m}(a_1, ..., a_n)$, we convert it to corresponding Horn Chain formula $\psi'_{k,m}$ by replacing all a_i's by it's complement $\tilde{a_i}$. Loosely speaking, a Horn chain formula is a chain formula where every literal is a negative literal.

Lemma 2.4.4 (Horn-chain-formula)**.** *Given a natural number $m > 0$ and $k < 2^m$, any chain formula $\psi_{k,m}$ can be transformed into a Horn-chain-formula $\psi'_{k,m}$ with the following properties:*

(i) *The support and structure of $\psi_{k,m}$ is preserved in $\psi'_{k,m}$.*

(ii) *$|\psi'_{k,m}|$ is linear in m and $\psi'_{k,m}$ has exactly k satisfying assignments.*

(iii) *$\psi'_{k,m}$ can be converted to an equivalent Horn formula with m variables and at most m clauses.*

Proof. Let $m > 0$ be a natural number and $k < 2^m$ and $\psi_{k,m}$ be the corresponding chain formula as defined above. Now we proceed to prove the properties below:

(i) Let $\psi'_{k,m} = \psi_{k,m}$. Now, for each $j \in \{1, \ldots, m\}$, we replace a_j by $\widetilde{a_j}$ in $\psi'_{k,m}$. Since none of the variables have been changed, $Supp(\psi_{k,m}) = Supp(\psi'_{k,m})$. Also, we have not changed any of the connectives, that is, for each j, C_j remains same. Thus, the structure of $\psi_{k,m}$ is preserved in $\psi'_{k,m}$. This completes the proof of the first part of the lemma.

(ii) First, we will prove that $\psi'_{k,m}$ has exactly k satisfying assignments. We prove this by using induction over m. Let us assume that the statement holds for all Horn-chain-formula upto m variables. Now we will prove this for Horn-chain-formula with $(m + 1)$ variables.

Here we will represent k in $m+1$ bits. Let k_1 denote the integer if we ignore the MSB of k in $(m+1)$-bit representation. Now we consider the following two cases:

Case (a): If the Most Significant Bit (MSB) of k in $(m+1)$-bit representation is 0, then we can write the above formula as $\widetilde{a_1} \wedge \psi'_{k_1,m}$. The only way to satisfy this formula is by satisfying both $\widetilde{a_1}$ and $\psi'_{k_1,m}$. Thus the number of solutions of this formula is k_1. Since the MSB is 0, $k = k_1$, and the statement holds true for this case.

Case (b): Now consider the other case when MSB of k in $(m+1)$-bit representation is 1. Then we can write the above formula as $\widetilde{a_1} \vee \psi'_{k_1,m}$. To satisfy this, we could either satisfy $\tilde{a_1}$, which can be done in 2^m ways by setting a_1 as False and rest of the variables can be assigned any value, or by setting a_1 as True and satisfying the formula $\psi'_{k_1,m}$ which has k_1 solutions. Thus the total number of solutions for this case is $2^m + k_1$. Also, note that $k = 2^m + k_1$.

Thus, $\psi'_{k,m}$ has exactly k satisfying assignments.

A similar inductive argument on m can be used to prove that $|\psi'_{k,m}|$ is linear in m, where two lists are sufficient to store the Horn-chain-formula $\psi'_{k,m}$. We use one list to store the m-bit binary representation of k, and another list to store the m literals of $\psi'_{k,m}$. This completes the proof of the second part of the lemma.

(iii) We use induction on the number of variables to prove the statement. Let us assume that the statement is true for all Horn-chain-formula upto $(m-1)$ many variables, and consider a Horn-chain-formula of m variables.

Let us first keep aside the variable a_1 associated with the MSB of m. Then we write the formula as $\psi'_{k,m} = \widetilde{a_1} C_1(\psi'_{k_1,m-1})$. Now $\psi'_{k_1,m-1} = \psi'_1 \wedge \psi'_2 \ldots \wedge \psi'_j$, where for each $i \in \{1, ..., j\}$, ψ'_i is a Horn clause with at most $m-1$ variables and $j \leq m-1$, following the induction hypothesis. Now, consider the two following cases:

Case (a): If C_1 is $\wedge$, then $\widetilde{a_1}$ becomes a unit negative clause which is a Horn clause itself. Thus, $\psi'_{k,m}$ is a Horn formula with m variables and at most m clauses.

Case (b): If C_1 is $\vee$, then $\psi'_{k,m} = \widetilde{a_1} \vee (\psi'_1 \wedge \ldots \wedge \psi'_j)$. By the distributive property, we can expand it into $\psi'_{k,m} = (\widetilde{a_1} \vee \psi'_1) \wedge (\widetilde{a_1} \vee \psi'_2) \ldots \wedge (\widetilde{a_1} \vee \psi'_j)$.

As the addition of the negative literal $\widetilde{a_1}$ does not change the nature of the Horn formulas ψ'_i, with $i \in \{1, \ldots, j\}$, $\psi'_{k,m}$ now has $m-1$ clauses each with at most m variables.

Therefore, $\psi'_{k,m}$ can be converted to an equivalent Horn formula with m variables and at most m clauses. This proves the third part of the lemma.

□

Example 2.4.5. Let's consider the Chain formula ψ taken in the Example 2.4.2, that is,

$$\psi_{7,5}(a_1, a_2, a_3, a_4, a_5) = (a_1 \wedge (a_2 \wedge (a_3 \vee (a_4 \vee a_5))))$$

We can convert it to the following Horn Chain formula by replacing all the literals a_i's by $\tilde{a_i}$,

$$\psi'_{7,5}(a_1, a_2, a_3, a_4, a_5) = (\tilde{a_1} \wedge (\tilde{a_2} \wedge (\tilde{a_3} \vee (\tilde{a_4} \vee \tilde{a_5}))))$$

Note that, $\psi'_{7,5}$ has exactly 7 satisfying assignments. We have to make a_1, a_2 as false and the rest can be satisfied in 7 ways. So the witness set is

$$R_{\psi'_{7,5}} = \{00000, 00001, 00010, 00011, 00100, 00101, 00110\}$$

and $\left| R_{\psi'_{7,5}} \right| = 7$. Now again note that, we can expand $\psi'_{7,5}$ as follows

$$\psi'_{7,5} = \tilde{a_1} \wedge \tilde{a_2} \wedge (\tilde{a_3} \vee \tilde{a_4} \vee \tilde{a_5})$$

where each clause is a Horn Clause and hence it is a Horn formula.

2.5 Log-Linear Distributions and Inverse Transform Sampling

Log-linear distributions have myriad of applications in machine learning, such as in graphical models, skip-gram models, and so on. See [23] for an exhaustive list of references. For any $\sigma \in \{0,1\}^n$ and any parameter θ, log-linear distribution is formally defined as:

$$\mathbb{P}[\sigma \mid \theta] \propto e^{\theta \cdot \sigma}$$

For our purpose, we use *literal weighted functions*, a notion defined by Chavira and Darwiche [12], which is equivalent to log-linear models.

Definition 2.5.1. For any Horn formula φ, and a set $S \subseteq Supp(\varphi)$, a weight function $wt : \{0,1\}^{|S|} \to (0,1)$ is said to be a literal weighted function, if there exists another function $W : S \to (0,1)$, such that for any satisfying assignment $\sigma \in R_{\varphi\downarrow S}$, $wt(\sigma)$ is defined as follows:

$$wt(\sigma) = \prod_{x \in \sigma} \begin{cases} W(x), & x = 1 \\ 1 - W(x), & x = 0 \end{cases}$$

wt is said to be literal weighted function with respect to the function W.

In order to construct the new Horn formula $\widehat{\varphi}$ from the input formula φ and the weight function wt, we apply the method of *inverse transform sampling.* The proof follows in similar line as that of Meel, Pote and Chakraborty [22]. However, instead of chain formulas, we use the notion of Horn-chain-formulas, introduced in Lemma 2.4.4.

Lemma 2.5.2. *Given any ε-Almost Additive Uniform-Horn-sampler $\mathcal{G}$, a Horn formula φ, along with a set $S = Supp(\varphi)$, a literal weighted function $wt : \{0,1\}^{|S|} \to (0,1)$, a new Horn formula $\widehat{\varphi}$ can be constructed such that the following holds:*

$$\forall \sigma \in R_{\varphi} : \ \frac{(1-\varepsilon)wt(\sigma)}{\sum\limits_{\sigma_1 \in R_{\varphi}} wt(\sigma_1)} \leq \mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma) \leq \frac{(1+\varepsilon)wt(\sigma)}{\sum\limits_{\sigma_1 \in R_{\varphi}} wt(\sigma_1)}$$

Proof. In order to construct the new Horn formula $\widehat{\varphi}$, for each $y_i \in S$, we will use a set of m_i many fresh variables $S_i = \{y_i^1, \ldots, y_i^{m_i}\}$ that have not been used before. Once we have the new variable set S_i, we will construct a Horn-chain-formula $\psi'_{k_i,m_i}(y_i^1, \ldots, y_i^{m_i})$ for some positive odd integer $k_i < 2^{m_i}$, as defined in Lemma 2.4.4. We will write $\psi'_{k_i,m_i}(y_i^1, \ldots, y_i^{m_i})$ as ψ'_{k_i,m_i} when it is clear from the context. We add the new clause $(y_i \iff \psi'_{k_i,m_i})$.

For each variable $y_i \in S$, if $y_i = 1$, then $W(y_i) = \frac{k_i}{2^{m_i}}$, and if $y_i = 0$, then $W(y_i) = 1 - \frac{k_i}{2^{m_i}}$, and $(y_i \iff \psi'_{k_i,m_i})$ is the corresponding clause. Thus, the Horn formula $\widehat{\varphi}$ is defined as follows:

$$\widehat{\varphi} = \varphi \bigwedge (\bigwedge_{i \in S} (y_i \iff \psi'_{k_i,m_i}))$$

The size of the set of satisfying assignments of $\widehat{\varphi}$ is as follows:

$$\left|R_{\widehat{\varphi}}\right| = \sum_{\sigma \in R_{\widehat{\varphi}}} 1 = \sum_{\sigma \in R_{\varphi}} \sum_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1 \downarrow S} = \sigma} 1$$

For any assignment σ, let σ_0 denote the set of variables that are assigned the False value in σ. Similarly, σ_1 corresponds to the set of variables that are assigned the True value in σ. Thus, we can say that $\sum\limits_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1 \downarrow S} = \sigma} 1 = \prod\limits_{i \in \sigma_0} (2^{m_i} - k_i) \prod\limits_{i \in \sigma_1} k_i$

Now consider the Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$:

$$\begin{aligned}
\mathbb{P}_{\mathcal{I}_{\mathcal{U}}}(\widehat{\varphi}, S, \sigma) &= \sum_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1\downarrow S} = \sigma} \mathbb{P}_{\mathcal{I}_{\mathcal{U}}}(\widehat{\varphi}, S, \sigma_1) \\
&= \sum_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1\downarrow S} = \sigma} \frac{1}{|R_{\widehat{\varphi}}|} \\
&= \frac{\sum\limits_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1\downarrow S} = \sigma} 1}{\sum\limits_{\sigma \in R_{\varphi}} \sum\limits_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1\downarrow S} = \sigma} 1} \\
&= \frac{\prod\limits_{i \in \sigma_0} (2^{m_i} - k_i) \prod\limits_{i \in \sigma_1} k_i}{\sum\limits_{\sigma \in R_{\varphi}} \prod\limits_{i \in \sigma_0} (2^{m_i} - k_i) \prod\limits_{i \in \sigma_1} k_i} \\
&= \frac{\prod\limits_{i \in \sigma_0} (2^{m_i} - k_i) \prod\limits_{i \in \sigma_1} k_i}{\prod\limits_{i \in S} 2^{m_i}} \cdot \frac{\prod\limits_{i \in S} 2^{m_i}}{\sum\limits_{\sigma \in R_{\varphi}} \prod\limits_{i \in \sigma_0} (2^{m_i} - k_i) \prod\limits_{i \in \sigma_1} k_i} \\
&= \frac{\prod\limits_{i \in S} W(\sigma_{\downarrow y_i})}{\sum\limits_{\sigma \in R_{\varphi}} \prod\limits_{i \in S} W(\sigma_{\downarrow y_i})} \\
&= \frac{wt(\sigma_1)}{\sum\limits_{\sigma \in R_{\varphi}} wt(\sigma)} \qquad (2.3)
\end{aligned}$$

As $\mathcal{G}$ is ε-close to the ideal Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$, we can say that

$$(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{U}}}(\varphi, S, \sigma) \leq \mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma) \leq (1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{U}}}(\varphi, S, \sigma)$$

Following Equation (2.3), we can say that

$$\forall \sigma \in R_{\varphi} : \ \frac{(1-\varepsilon) wt(\sigma)}{\sum\limits_{\sigma_1 \in R_{\varphi}} wt(\sigma_1)} \leq \mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma) \leq \frac{(1+\varepsilon) wt(\sigma)}{\sum\limits_{\sigma_1 \in R_{\varphi}} wt(\sigma_1)}$$

□

As it was also mentioned in [22], we would like to emphasize the fact that the above lemma (Lemma 2.5.2) holds only for ε- AAU Horn samplers. The analogous statement does not hold for η-far Horn samplers. As a result, we can not directly apply Lemma 2.5.2 to test closeness to ideal Uniform-Horn-sampler.

Chapter 3

Tester for Uniform Horn Sampler

In this chapter, we shall describe our tester Flash for Uniform-Horn-samplers. We shall describe first the methodology of Flash in section 3.1 and then will prove the correctness of Flash and determine the sample complexity of our approach in section 3.2.

3.1 Methodology of Flash

Our Uniform-Horn-sampler-tester Flash , takes as input a black-box Horn sampler $\mathcal{G}$, a Horn formula φ, three parameters $\varepsilon, \eta, \delta$, such that $\varepsilon \in (0, \frac{1}{3}]$, $\eta > 9\varepsilon$, $\delta > 0$, and outputs ACCEPT with probability at least $1 - \delta$, if $\mathcal{G}$ is an ε-AAU-Horn-sampler, and if the distribution of the satisfying assignments corresponding to $\mathcal{G}$, that is, $D_{\mathcal{G}(\varphi)}$, is η-far in ℓ_1 distance from the uniform distribution, it outputs REJECT with probability at least $1 - \delta$.

While the basic framework of our tester Flash is similar to that of Barbarik , there are significant and crucial differences particularly in the subroutines used. So, we start by describing the subroutines.

3.1.1 HornKernel

The job of the subroutine HornKernel is to take as input a Horn formula φ, two satisfying assignments $\sigma_1, \sigma_2 \in R_\varphi$, and an integer κ, and returns a Horn formula φ' such that φ and φ' have "similar" structures, and the following properties are satisfied:

(1) $\left|R_{\varphi'}\right| = 2\tau$. (2) $Supp(\varphi) \subseteq Supp(\varphi')$.

(3) Let $\tilde{0}$ denote the assignment whose only True literals are the common true literals of σ_1 and σ_2.

- If $\tilde{0} \notin R_\varphi$, then $R_{\varphi'\downarrow S}$ has only two elements σ_1 and σ_2 and

$$\left|\{x \mid x \in R_{\varphi'} \ \&\ x_{\downarrow S} = \sigma_1\}\right| = \left|\{x \mid x \in R_{\varphi'} \ \&\ x_{\downarrow S} = \sigma_2\}\right|.$$

- If $\tilde{0} \in R_\varphi$ then $R_{\varphi'\downarrow S}$ has only three elements σ_1, σ_2 and $\tilde{0}$ and

$$\begin{aligned}\left|\{x \mid x \in R_{\varphi'} \ \&\ x_{\downarrow S} = \sigma_1\}\right| &= \left|\{x \mid x \in R_{\varphi'} \ \&\ x_{\downarrow S} = \sigma_2\}\right| \\ &= \left|\{x \mid x \in R_{\varphi'} \ \&\ x_{\downarrow S} = \tilde{0}\}\right|.\end{aligned}$$

In order to achieve these properties, the subroutine HornKernel , in Line 1, constructs a Horn formula $\mathcal{T}$ from σ_1, σ_2 using the subroutine Encode , such that σ_1, σ_2 and $\tilde{0}$ are the only satisfying assignments of $\mathcal{T}$. We note that, it is in fact not possible to construct a Horn formula with only two satisfying assignments σ_1 and σ_2, and due to this reason, we have to deal with the third satisfying assignment, namely $\tilde{0}$. The subroutine Encode (Algorithm 3) is one of the technical contributions of this work.

```
Algorithm 1: HornKernel $(\varphi, \sigma_1, \sigma_2, \tau)$
1  $\mathcal{T} \leftarrow$ Encode $(\sigma_1, \sigma_2)$;
2  $\hat{\varphi} \leftarrow \varphi \wedge \mathcal{T}$;
3  $Lits_1 \leftarrow (\sigma_1 \backslash \sigma_2)$;
4  $Lits_2 \leftarrow (\sigma_2 \backslash \sigma_1)$;
5  $n \leftarrow \min(|Lits_1 \cup Lits_2|, 4)$;
6  $k \leftarrow \lceil \tau^{1/n} \rceil$, $m \leftarrow \lceil \log(k) \rceil$;
7  $V \leftarrow$ NewVars $(\varphi, m, n)$;
8  $ix \leftarrow 0$;
9  for $i \in [n]$ do
10     $l \sim Lits_1 \cup Lits_2$;
11     $\hat{\varphi} \leftarrow \hat{\varphi} \wedge (l \rightarrow \psi'_{k,m}(V[ix : ix + m]))$;
12     $\hat{\varphi} \leftarrow \hat{\varphi} \wedge (\neg l \rightarrow \psi'_{k,m}(V[ix : ix + m]))$;
13     $ix \leftarrow ix + m$;
14 return $\hat{\varphi}$;
```

```
Algorithm 2: NewVars $(\varphi, m, n)$
1  $\mathcal{R} \leftarrow$ set of all variables;
2  $S \leftarrow supp(\varphi)$;
3  $V \leftarrow \emptyset$;
4  for $i \in [n]$ and $j \in [m]$ do
5      $l \sim \mathcal{R} \setminus S$;
6      $V \leftarrow V \cup l$;
7  return $V$;
```

We shall describe the algorithm HornKernel assuming the subroutine Encode which we shall be describing in the next subsection. After HornKernel constructs a Horn formula $\mathcal{T}$ from σ_1, σ_2 using the subroutine Encode (in Line 1), such that σ_1, σ_2 and $\tilde{0}$ are the only satisfying assignments of $\mathcal{T}$, in Line 2, it constructs a new Horn formula $\widehat{\varphi}$ by conjuncting the original Horn formula φ with $\mathcal{T}$. In Line 3 and Line 4, HornKernel constructs the symmetric difference of σ_1 and σ_2 by generating two sets $Lits_1$ and $Lits_2$, where the symmetric difference corresponds to a set of literals which are true in exactly one of σ_1 and σ_2. In Line 5, it sets the variable n to minimum of $|Lits_1 \cup Lits_2|$ and 4, which indicates the number of Horn-chain formulae it will generate further. Then in Line 6, it calculates the values of k and m which are the parameters needed to produce Horn-chain formula $\psi'_{k,m}$ (from Lemma 2.4.4). As noted earlier, the construction of Horn-chain formulas in Lemma 2.4.4, is another important technical contribution of this thesis. Note that, $\psi'_{k,m}$ has exactly k many satisfying assignments. Then, in Line 7, it generates a list V of $n \times m$ many new variables that are not present in φ using a subroutine NewVars . Finally in the Loop of Line 9, HornKernel constructs a new Horn formula $\hat{\varphi}$ by adding n many Horn-clauses of the form $(l \rightarrow \psi'_{k,m})$ and $(\neg l \rightarrow \psi'_{k,m})$ over the set of newly generated variables V, where l is a literal sampled from $Lits_1 \cup Lits_2$, and $\psi'_{k,m}$ is a Horn-chain formula defined over the variables of V.

3.1.2 Encode

As we have already stated, this subroutine is on eof the key contribution of this work. Given the two witnesses σ_1 and σ_2 as input, the overall idea of Encode is to partition the set of variables appearing in σ_1 and σ_2 into four equivalence classes [1]: (i) equivalence class $[x_{tLit}]$ containing all the common True literals of σ_1 and σ_2, (ii) equivalence class $[x_{fLit}]$ containing all the common False literals of σ_1 and σ_2, (iii) equivalence class $[x_{diff1}]$ containing all the literals which are True in σ_1, but False in σ_2, and (iv) equivalence class $[x_{diff2}]$ containing all the literals which are False in σ_1, but True in σ_2. Thus Encode first finds the set of common True and False literals of σ_1 and σ_2 by means of $cmmTrueLits$ and $cmmFalseLits$ respectively. It also finds the set of literals that have different values in Line 5 using $UnCmmLits$.

In order to consider the first equivalence class $[x_{tLit}]$ containing only the common True literals of σ_1 and σ_2, in the for loop at Line 8, it constructs a formula $\mathcal{T}$ by adding

[1]Equivalence classes with respect to the relation $\iff$.

equivalence between x_{tLit} and the common True literals obtained from Line 3. Moreover, to ensure that all these variables are assigned the True value, it further conjuncts the literal x_{tLit} with $\mathcal{T}$ in Line 10. Similarly, for the second equivalence class $[x_{fLit}]$ of the common False literals of σ_1 and σ_2, it runs the for loop in Line 11, and conjuncts the literal $\neg x_{fLit}$ in Line 13. In order to take care of the last two equivalence classes, it first finds two variables x_{diff1} and x_{diff2} (using $findSplittingVars$) such that, $x_{diff1} = 0$ and $x_{diff2} = 1$ in the witness σ_1, but $x_{diff1} = 1$ and $x_{diff2} = 0$ in the witness σ_2 [2]. Using these two variables x_{diff1} and x_{diff2}, Encode constructs two formulas in the for loop starting from Line 17. Finally, to ensure the different values of the two equivalence classes, Encode adds the formula $(x_{diff1} \implies \neg x_{diff2})$ in Line 23. It is interesting to note that we might have added the formula $(x_{diff1} \iff \neg x_{diff2})$ in Line 23. However, it turns out that $(\neg x_{diff2} \implies x_{diff1})$ is not a Horn formula [3]. This causes 3 witnesses of $\mathcal{T}$: σ_1, σ_2, and $\tilde{0}$, instead of only σ_1 and σ_2.

Example 3.1.1. Let's consider two satisfying assignments of a Horn formula φ defined on eight literals $x_1, x_2, .., x_8$,

$$\sigma_1 = 11001100$$

$$\sigma_2 = 11110000$$

Then a formula $\mathcal{T}$ returned by Encode can be as follows:

$$\mathcal{T} := x_1 \neg x_7 (x_1 \iff x_2)(x_7 \iff x_8)(x_3 \iff x_4)(x_5 \iff x_6)(x_3 \implies \neg x_5)$$

Note that, the satisfying assignments of $\mathcal{T}$ are σ_1, σ_2 and $\tilde{0} = 11000000$.

[2]Since $\sigma_1 \neq \sigma_2$, the existence of at least one of the variables x_{diff1} and x_{diff2} is guaranteed.
[3]$(\neg x_{diff2} \implies x_{diff1}) \equiv (x_{diff2} \vee x_{diff1})$ contains 2 positive literals.

Algorithm 3: Encode (σ_1, σ_2)

```
1  $\Sigma \leftarrow [\sigma_1, \sigma_2]$;
2  $\mathcal{T} \leftarrow True$;
3  $TrueLits \leftarrow cmmTrueLits(\sigma_1, \sigma_2)$;
4  $FalseLits \leftarrow cmmFalseLits(\sigma_1, \sigma_2)$;
5  $UnCmmLits \leftarrow unCmmLits(\sigma_1, \sigma_2)$;
6  $tLit \leftarrow TrueLits[0]$;
7  $fLit \leftarrow FalseLits[0]$;
8  for each $i \in TrueLits \setminus \{tLit\}$ do
9      $\mathcal{T} \leftarrow \mathcal{T} \wedge (x_i \iff x_{tLit})$;
10 $\mathcal{T} \leftarrow \mathcal{T} \wedge x_{tLit}$;
11 for each $i \in FalseLits \setminus \{fLit\}$ do
12     $\mathcal{T} \leftarrow \mathcal{T} \wedge (x_i \iff x_{fLit})$;
13 $\mathcal{T} \leftarrow \mathcal{T} \wedge \neg x_{fLit}$;
14 $diff_1 \leftarrow$ NULL;
15 $diff_2 \leftarrow$ NULL;
16 $(diff_1, diff_2) \leftarrow findSplitVars(\sigma_1, \sigma_2)$;
17 for each $i \in unCmmLits$ do
18     if $val(x_i, \sigma_1) == 1$ then
19         $\mathcal{T} \leftarrow \mathcal{T} \wedge (x_i \iff x_{diff_1})$;
20     else
21         $\mathcal{T} \leftarrow \mathcal{T} \wedge (x_i \iff x_{diff_2})$;
22 if $diff_1 \neq$ NULL & $diff_2 \neq$ NULL then
23     $\mathcal{T} \leftarrow \mathcal{T} \wedge (x_{diff_1} \implies \neg x_{diff_2})$;
24 return $\mathcal{T}$;
```

3.1.3 Flash

Finally, we present the main algorithm Flash (Algorithm 4). It first draws t many samples from the sampler $\mathcal{G}$ to be tested, as well as from the ideal Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$ and stores them in Γ_1 and Γ_2 in Line 9 and Line 10 respectively. Thus, Γ_1 is a set of samples from the distribution $D_{\mathcal{G}(\varphi)}$, while Γ_2 is a set of samples drawn according to the uniform distribution over R_φ, the witness space of φ. Then in the for loop of Line 11, it first takes a sample σ_1 from Γ_1, and another sample σ_2 from Γ_2, and calls the subroutine HornKernel with σ_1, σ_2 and φ in Line 16. Now Flash obtains M many satisfying assignments of $\hat{\varphi}$ in Line 17, and calls the subroutine RemoveZeros in Line 18 to check if there are at least N many witnesses of σ_1 and σ_2 out of the M witnesses obtained in Line 17. If the number of witnesses of σ_1 and σ_2 is less than N, Flash outputs REJECT and terminates the algorithm. Otherwise, it employs the subroutine Bias in Line 21 in order to determine the fraction of witnesses obtained from RemoveZeros that are same as σ_1 when projected on S. If this fraction is more than T, then it outputs REJECT . If Flash does not output REJECT in any of the t iterations of the for loop of Line 11, it finally outputs ACCEPT , and declares that the distribution induced by the satisfying assignments produced by the Horn sampler $\mathcal{G}$ is ε-close to the uniform distribution.

Flash has similar flavor to that of Barbarik2 (of [22]). But unlike their algorithm, we have to carefully handle the fact that the formula φ' on which the sampling algorithm is run must be a Horn formula. At the same time we have to handle the extra complication of the fact that the formula φ' returned by the the subroutine HornKernel may have three satisfying assignments (after projecting on to the support size of φ), instead of exactly two - namely σ_1 and σ_2 which was crucially used in the correctness proof of Barbarik2 .

3.2 Theoretical Analysis of Flash

The algorithm Flash is also theoretical sound. If the sampler to be tested is ε-additive almost uniform (AAU) sampler, then Flash outputs ACCEPT with probability at least $(1-\delta)$. But to prove that Flash outputs REJECT if the sampler that is η-far from the ideal Uniform-Horn-sampler, we would need an extra assumption on the sampler that is being tested. We will term this (using the same terminology from [22]) *Subquery Consistency* of Sampler.

Definition 3.2.1 (Subquery Consistency of Sampler)**.** Consider any Horn formula φ. For

Algorithm 4: Flash ($\mathcal{G}$, $\mathcal{U}$, S, ε, η, δ, φ)

```
1  $t \leftarrow \frac{10}{\eta(\eta-9\varepsilon)} \log_e\left(\frac{1}{\delta}\right)$;
2  $n \leftarrow \log_e\left(\frac{2t}{\delta}\right)$;
3  $L \leftarrow \frac{1+\varepsilon}{2}$;
4  $H \leftarrow \frac{1+\frac{\eta+9\varepsilon}{4}}{2+\frac{\eta+9\varepsilon}{4}}$;
5  $T = \frac{(H+L)}{2}$;
6  $N \leftarrow \frac{8n \cdot H}{(H-L)^2}$;
7  $X \leftarrow 2\left(\frac{1-\varepsilon}{3-\varepsilon}\right)$;
8  $M \leftarrow \left(\frac{\sqrt{n}+\sqrt{n+4NX}}{2X}\right)^2$;
9  $\Gamma_1 \leftarrow \mathcal{G}(\varphi, S, t)$;
10 $\Gamma_2 \leftarrow \mathcal{I}_{\mathcal{U}}(\varphi, S, t)$;
11 **for** $i \leftarrow 1$ *to* $t$ **do**
12     $\sigma_1 \leftarrow \Gamma_1[i]$;
13     $\sigma_2 \leftarrow \Gamma_2[i]$;
14     **if** $\sigma_1 == \sigma_2$ **then**
15         **continue**
16     $\hat{\varphi} \leftarrow$ HornKernel $(\varphi, \sigma_1, \sigma_2)$;
17     $\Gamma_3 \leftarrow \mathcal{G}(\hat{\varphi}, S, M)$;
18     $\widehat{\Gamma}_3 \leftarrow$ RemoveZeros $(\Gamma_3)$;
19     **if** $\left|\widehat{\Gamma}_3\right| < N$ **then**
20         **return** REJECT
21     $Bias \leftarrow$ Bias $(\sigma_1, \widehat{\Gamma}_3, S)$;
22     **if** $Bias > T$ **then**
23         **return** REJECT
24 **return** ACCEPT
```

all $S \subseteq Supp(\varphi)$, $\sigma_1, \sigma_2 \in R_{\varphi \downarrow S}$, let $\widehat{\varphi}$ be the Horn formula obtained from the subroutine ENCODE. A Horn sampler $\mathcal{G}$ is said to be *subquery consistent*, if the output of $\mathcal{G}(\widehat{\varphi}, wt, S, \kappa)$ is κ many independent samples from the distribution $\mathcal{D}_{\mathcal{G}(\varphi)|X}$, that is, the distribution $\mathcal{D}_{\mathcal{G}(\varphi)}$ conditioned on the set X, where either $X = \{\sigma_1, \sigma_2\}$ or $X = \{\sigma_1, \sigma_2, \tilde{0}\}$ depending on whether $\tilde{0} \in R_\varphi$ or not.

Now we move towards proving the formal statement of correctness of Flash . The formal statement of correctness of Flash is stated below.

Theorem 3.2.2 (Correctness of Flash). *Given a Horn sampler $\mathcal{G}$, a tolerance parameter $\varepsilon \in (0, \frac{1}{3}]$, an intolerance parameter $\eta \in (0, 2]$, with $\eta > 9\varepsilon$ and a confidence parameter $\delta > 0$, our Uniform-Horn-sampler-tester* Flash *takes $\widetilde{\mathcal{O}}(\frac{1}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2})$ many samples, and decides the following:*

(i) If $\mathcal{G}$ is an ε-additive almost uniform (AAU) Horn sampler, Flash *outputs ACCEPT with probability at least $1 - \delta$.*

(ii) If $\mathcal{G}$ is η-far from being the ideal Uniform-Horn-sampler and $\mathcal{G}$ is subquery consistent, Flash *outputs REJECT with probability at least $1 - \delta$.*

where, $\widetilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic factors in $\frac{1}{\eta-3\varepsilon}, \frac{1}{\eta-9\varepsilon}, \frac{1}{\eta}, \frac{1}{\delta}$.

Completeness Theorem

Theorem 3.2.3 (Completeness). *If the Horn sampler $\mathcal{G}$ is an ε-additive almost-uniform (AAU) Horn sampler, then* Flash *outputs ACCEPT with probability at least $(1 - \delta)$.*

In order to prove this theorem, we will use the following two lemmas.

Lemma 3.2.4. *If $\mathcal{G}$ is ε-AAU Horn sampler, then when we draw M samples from $\mathcal{G}(\hat{\varphi}, .)$, the probability that $\widetilde{0}$ appears at least $(M - N)$ many times among M samples is at most $\frac{\delta}{2t}$.*

Lemma 3.2.5. *Given that $\tilde{0}$ has appeared less than $(M - N)$ times out of the M samples, the probability that* Flash *outputs REJECT in each iteration is at most $\frac{\delta}{2t}$.*

Assuming Lemma 3.2.4 and Lemma 3.2.5 hold, now we proceed to prove Theorem 3.2.3.

Proof. Note that Flash (Algorithm 4) outputs REJECT when either number of times $\tilde{0}$ appears at least $(M - N)$ times among M samples, or when the $Bias$ computed is more than T. Let us now divide them into two cases as follows:

Case (i): $\widetilde{0}$ appears at least $(M - N)$ times among M samples from $\mathcal{G}$.

Case (ii): $Bias > T$ as determined in Line 21.

From Lemma 3.2.4, we know that the probability of Case (i) is at most $\frac{\delta}{2t}$. Also, by Lemma 3.2.5, we know the probability of Case (ii) is at most $\frac{\delta}{2t}$ as well. So, combining both Case (i) and Case (ii), we can say that with probability at most $\frac{\delta}{t}$, Flash outputs REJECT in any iteration. Since there are t many iterations of Flash , the probability that Flash outputs ACCEPT is at least $(1 - \frac{\delta}{t})^t \geq 1 - \delta$. This completes the proof of Theorem 3.2.3.

□

Now we proceed to prove Lemma 3.2.4.

Proof of Lemma 3.2.4. Let us define the following binary random variable:

$$Y_j = \begin{cases} 1 & \text{if } \Gamma_3[j] = \tilde{0} \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathcal{G}$ is ε-AAU, from Definition 2.1.6 we can say that

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)} \leq \frac{1+\varepsilon}{1-\varepsilon} \tag{3.1}$$

Similarly, we have

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \leq \frac{1+\varepsilon}{1-\varepsilon} \tag{3.2}$$

Thus, noting the fact that, $\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0}) = 1$,

$$\begin{aligned}
\mathbb{E}[Y_j] &= \mathbb{P}_{\mathcal{G}}(Y_j = 1) \\
&= \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0}) \\
&= \frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})} \\
&\leq \frac{1+\varepsilon}{3-\varepsilon} \qquad (3.3)
\end{aligned}$$

Where the last inequality follows from the fact that if $\frac{a}{c} < \frac{l}{m}$ and $\frac{a}{c} < \frac{l}{n}$, then $\frac{a}{b+c} < \frac{l}{m+n}$ and $\frac{a}{a+b+c} < \frac{l}{l+m+n}$, along with Equation (3.1) and Equation (3.2).

Now consider the random variable Y defined as $Y = \sum_{j=1}^{M} Y_j$. Following Equation (3.3), we can say that, $\mathbb{E}[Y] \leq \frac{1+\varepsilon}{3-\varepsilon} M$.

Applying Chernoff bound 2.3.2, we can say that the probability that Y is more than $(M - N)$ is $\mathbb{P}(Y > M - N) \leq \frac{\delta}{2t}$. Thus, with probability at least $\left(1 - \frac{\delta}{t}\right)$, $\widetilde{0}$ appears less than $(M - N)$ many times among M samples obtained from $\mathcal{G}$ in Line 17 of Flash .

□

Proof of Lemma 3.2.5. Consider the case when the count of $\widetilde{0}$ appears less than $(M - N)$ times out of the M samples obtained from the sampler $\mathcal{G}$ in Line 17 of the algorithm Flash . Now, recall the set $\widehat{\Gamma}_3$ from Flash that contains only σ_1 and σ_2 after removing all $\tilde{0}$ from Γ. Let us define the following binary random variable Z_i for each sample j of $\widehat{\Gamma}_3$.

$$Z_j = \begin{cases} 1 & \text{if } \widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_1 \\ 0 & \text{if } \widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_2 \end{cases}$$

So, the expected value of Z_j is given by

$$\mathbb{E}[Z_j] = \frac{\mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma_2)}$$

As $\mathcal{G}$ is ε-AAU, we can say that

$$\frac{\mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\widehat{\varphi}, S, \sigma_2)} \leq \frac{1+\varepsilon}{1-\varepsilon}$$

Thus, $\mathbb{E}[Z_j] \leq \frac{1+\varepsilon}{2}$. We set $L = \frac{1+\varepsilon}{2}$ in the algorithm Flash .

Note that Bias is computed as follows in Line 21 of Flash :

$$Bias = \sum_{j \in [|\widehat{\Gamma}_3|]} \frac{\mathbb{1}(\widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_1)}{\mathbb{1}(\widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_1) + \mathbb{1}(\widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_2)} = \sum_{j \in |\widehat{\Gamma}_3|} \frac{Z_j}{\left|\widehat{\Gamma_3}\right|}$$

So, the probability that $Bias > T$ is given by:

$$\begin{aligned} \mathbb{P}(Bias > T \mid \left|\widehat{\Gamma}_3\right| \geq N) &= \mathbb{P}(\sum_{j \in |\widehat{\Gamma}_3|} \frac{Z_j}{\left|\widehat{\Gamma_3}\right|} > T \mid \left|\widehat{\Gamma}_3\right| \geq N) \\ &\leq \exp\left(-\frac{(H-L)^2 N}{8H}\right) \\ &\leq \frac{\delta}{t} \end{aligned}$$

□

Soundness Theorem

Theorem 3.2.6 (Soundness). *If $\mathcal{G}$ is η-far from being the ideal Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$ and $\mathcal{G}$ is subquery consistent,* Flash *outputs REJECT with probability at least $1-\delta$.*

Before proceeding to prove Theorem 3.2.6, let us first partition the set R_φ, the set of satisfying assignments of φ into the following subsets:

- $W_{-1} = \{x \in R_\varphi : \mathbb{P}_\mathcal{G}(\varphi, x) \leq \frac{1}{N}\}$
- $W_0 = \{x \in R_\varphi : \frac{1}{N} < \mathbb{P}_\mathcal{G}(\varphi, x) < \left(1 + \frac{\eta+9\varepsilon}{4}\right)\frac{1}{N}\}$
- $W_1 = \{x \in R_\varphi : \left(1 + \frac{\eta+9\varepsilon}{4}\right)\frac{1}{N} \leq \mathbb{P}_\mathcal{G}(\varphi, x)\}$

Lemma 3.2.7. *If $\mathcal{G}$ is η-far from the ideal Uniform-Horn-sampler $\mathcal{I}_\mathcal{U}$, then*

$$\mathbb{P}\left(Bias > T \mid (\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1})\right) \geq \frac{4}{5}$$

Lemma 3.2.8. *If the sampler $\mathcal{G}$ is η-far from the ideal Uniform-Horn-sampler $\mathcal{I}_\mathcal{U}$, then*

$$\mathbb{P}\left(\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1}\right) \geq \frac{\eta(\eta - 9\varepsilon)}{8}$$

Assuming Lemma 3.2.7 and Lemma 3.2.8 hold, we are now ready to prove Theorem 3.2.6.

Proof of Theorem 3.2.6. Let us first define the following events:

$$\mathcal{E}_1 := \sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1}$$

$$\mathcal{E}_2 := Bias > T \text{ in an iteration}$$

So, following Lemma 3.2.7 and Lemma 3.2.8, we can say that:

$$\mathbb{P}(\mathcal{E}_2) = \mathbb{P}(\mathcal{E}_2 \mid \mathcal{E}_1)\mathbb{P}(\mathcal{E}_1) \geq \frac{4}{5}\frac{\eta(\eta - 9\varepsilon)}{8}$$

Thus, the probability that Flash returns REJECT is:

$$\mathbb{P}\left(\mathsf{Flash} \text{ returns } \text{ REJECT }\right) \geq 1 - \left(1 - \frac{\eta(\eta - 9\varepsilon)}{10}\right)^t$$

As $t = \frac{10}{\eta(\eta-9\varepsilon)} \log_e \frac{1}{\delta}$, when $\mathcal{G}$ is η-far from the Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$, Flash rejects $\mathcal{G}$ with probability at least $1 - \delta$. This completes the proof of Theorem 3.2.6.

□

Proof of Lemma 3.2.7. Consider the case when $\sigma_1 \in W_1$ and $\sigma_2 \in W_{-1}$. From the definition of W_1, we know that $\mathbb{P}_{\mathcal{G}}(\varphi, \sigma_1) \geq \left(1 + \frac{\eta+9\varepsilon}{4}\right)\frac{1}{N}$. Also, from the definition of W, we can say that $\mathbb{P}_{\mathcal{G}}(\varphi, \sigma_2) < \frac{1}{N}$. So, we can say that $\mathbb{P}_{\mathcal{G}}(\varphi, \sigma_1) \geq \left(1 + \frac{\eta+9\varepsilon}{4}\right)\mathbb{P}_{\mathcal{G}}(\varphi, \sigma_2)$.

Assuming the subquery consistency property of $\mathcal{G}$, along with the fact that $\widehat{\Gamma}_3$ contains only σ_1 and σ_2, we can say that

$$\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) = \frac{\mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_2)} \geq \frac{1 + \frac{\eta+9\varepsilon}{4}}{2 + \frac{\eta+9\varepsilon}{4}}$$

As $H = \frac{1+\frac{\eta+9\varepsilon}{4}}{2+\frac{\eta+9\varepsilon}{4}}$, applying Chernoff bound, we can say that,

$$\mathbb{P}(Bias \leq T \mid \sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1}) \leq \frac{1}{5}$$

So, the proof of the lemma follows.

□

Proof of Lemma 3.2.8. Since $\mathcal{G}$ is η-far from the Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$, we can say that

$$\sum_{x \in W} \left(\mathbb{P}_{\mathcal{G}}(\varphi, x) - \frac{1}{N} \right) = \sum_{x \in W_{-1}} \left(\frac{1}{N} - \mathbb{P}_{\mathcal{G}}(\varphi, x) \right) \geq \frac{\eta}{2} \tag{3.4}$$

Therefore from Equation (3.4),

$$\frac{|W_{-1}|}{N} \geq \frac{\eta}{2} \tag{3.5}$$

Now, $\sum\limits_{x \in W_0} \left(\mathbb{P}_{\mathcal{G}}(\varphi, x) - \frac{1}{N} \right) \leq \sum\limits_{x \in W_0} \frac{\eta+9\varepsilon}{4} \frac{1}{N} < \frac{\eta+9\varepsilon}{4}$

Thus,

$$\sum_{x \in W_1} \mathbb{P}_{\mathcal{G}}(\varphi, x) \geq \frac{\eta}{2} - \frac{\eta + 9\varepsilon}{4} \geq \frac{\eta - 9\varepsilon}{4} \tag{3.6}$$

Hence, from the independence of σ_1 and σ_2 we have proven the Lemma from Equation (3.5) and Equation (3.6). □

Sample Complexity of Flash

Theorem 3.2.9. *The sample complexity of* Flash *is* $\widetilde{\mathcal{O}}(\frac{1}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2})$*, where* $\widetilde{\mathcal{O}}(\cdot)$ *hides poly-logarithmic factor in* $\frac{1}{\eta}, \frac{1}{\eta-3\varepsilon}, \frac{1}{\eta-9\varepsilon}$*, and* $\frac{1}{\delta}$.

Proof. We first note that Flash takes t samples from $\mathcal{G}$ and the ideal Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$ in Line 9 and Line 10 respectively. Thereafter, in each iteration of the for loop starting from Line 11, it takes M samples from $\mathcal{G}$ in Line 17. Since the loop runs for t iterations, the sample complexity is $2t + Mt$, which is at most $2Mt$. Below, we will bound the value of $2Mt$.

First, we see that $N = \frac{8nH}{(H-L)^2}$. As $H = \frac{1+\frac{\eta+9\varepsilon}{4}}{2+\frac{\eta+9\varepsilon}{4}}$, and $L = \frac{1+\varepsilon}{2}$, we can say that $N \leq 8\frac{\log \frac{2t}{\delta}}{(\eta-3\varepsilon)^2}$.

As $M = \left(\frac{\sqrt{n}+\sqrt{n+4NX}}{2X}\right)^2 \leq \frac{n+4NX}{X^2}$, $X = 2\left(\frac{1-\varepsilon}{3-\varepsilon}\right)$, we can say that

$$\begin{aligned}
2Mt &\leq 2t\left(\log\frac{2t}{\delta}\frac{1}{X^2} + \frac{4N}{X}\right) \\
&\leq 2t\left(\log\frac{2t}{\delta}\cdot 6 + 32\log\frac{2t}{\delta}\frac{1}{(\eta-3\varepsilon)^2}\right) \\
&= \widetilde{\mathcal{O}}\left(\frac{1}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2}\right)
\end{aligned}$$

□

Chapter 4

Tester for General Horn Sampler

In this chapter, we shall describe our tester wFlash for Weighted-Horn-samplers. Following the organisation of previous chapter, in this chapter again we shall describe first the methodology of wFlash in section 4.1 and then will provide the correctness proof of wFlash and determine the sample complexity in section 4.2.

4.1 Methodology of wFlash

In this section, we describe the algorithmic framework of wFlash , to test the ideality of a black-box Weighted-Horn-sampler$\mathcal{G}$. wFlash takes as input a black-box Horn sampler $\mathcal{G}$ (which is to be tested), access to an ideal Horn sampler $\mathcal{I}_{\mathcal{W}}$, a Boolean formula φ along with corresponding weight function wt, and three parameters $\varepsilon, \eta, \delta$, such that $\varepsilon \in (0, \frac{1}{3}], \eta \geq 9\varepsilon$, and $\delta > 0$, and if $\mathcal{G}$ is η-far from $\mathcal{I}_{\mathcal{W}}$, then with probability at least $(1 - \delta)$, it outputs REJECT and if $\mathcal{G}$ is ε-close to $\mathcal{I}_{\mathcal{W}}$, it outputs ACCEPT with probability at least $(1 - \delta)$. Our main result here is stated as follows:

Theorem 4.1.1 (Correctness of wFlash). *Given a Weighted-Horn-sampler $\mathcal{G}$, an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, a tolerance parameter $\varepsilon \in (0, \frac{1}{3}]$, an intolerance parameter $\eta > 0$, with $\eta > 9\varepsilon$ and a confidence parameter $\delta > 0$, and an arbitrary but fixed weight function wt, our Weighted-Horn-sampler-tester* wFlash *takes $\widetilde{\mathcal{O}}(\frac{\mathsf{tilt}(wt,\varphi)^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2})$ many samples, and decides the following:*

(i) If $\mathcal{G}$ is ε-close to the Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, wFlash *outputs ACCEPT with probability at least $1 - \delta$.*

(ii) If $\mathcal{G}$ is η-far from the Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, wFlash *outputs REJECT with probability at least $1 - \delta$.*

where $\mathsf{tilt}(wt, \varphi)$ denotes the maximum ratio between any two satisfying assignments of φ with respect to the weight function wt, and $\widetilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic factors in $\frac{1}{\eta}, \frac{1}{\eta-3\varepsilon}, \frac{1}{\eta-9\varepsilon}$, and $\frac{1}{\delta}$.

Let us assume that the distribution generated by $\mathcal{G}(\varphi)$ is denoted by $\mathcal{D}_{\mathcal{G}(\varphi)}$. In order to test whether $\mathcal{G}$ is ε-close or η-far from $\mathcal{I}_{\mathcal{W}}$, one might need to draw exponentially many samples from $\mathcal{I}_{\mathcal{W}}$ and $\mathcal{G}$. However, wFlash uses the conditional sampling paradigm to handle this situation. In each iteration, it takes a sample σ_1 drawn from $\mathcal{G}$ and another sample σ_2 drawn from $\mathcal{I}_{\mathcal{W}}$ and infers about the similarity between σ_1 and σ_2 in an effective way. To achieve this, it draws samples from the conditional distribution $\mathcal{D}_{\mathcal{G}(\varphi)|\{\sigma_1,\sigma_2\}}$. However, note that we do not actually have any direct access to such distribution. Here we use the HornKernel subroutine, that takes as input a formula φ, two satisfying assignments σ_1 and σ_2 and returns a formula $\hat{\varphi}$ such that, either $\mathcal{D}_{\mathcal{G}(\hat{\varphi})} \approx \mathcal{D}_{\mathcal{G}(\varphi)|\{\sigma_1,\sigma_2\}}$ or $\mathcal{D}_{\mathcal{G}(\hat{\varphi})} \approx \mathcal{D}_{\mathcal{G}(\varphi)|\{\sigma_1,\sigma_2,\tilde{0}\}}$ (this depends on whether $\tilde{0}$ is a satisfying assignment of φ).

The main idea of wFlash is that, if $\mathcal{G}$ is η-far, then with high probability we receive σ_1 and σ_2 in an iteration such that they are far apart according to the distribution $\mathcal{D}_{\mathcal{G}(\varphi)}$

(that is, the distance between $\mathbb{P}_{\mathcal{G}}(\varphi, \sigma_1)$ and $\mathbb{P}_{\mathcal{G}}(\varphi, \sigma_2)$ from $\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, \sigma_1)$ and $\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, \sigma_2)$ is significantly large and thus can be differentiated by wFlash). To estimate this difference between σ_1 and σ_2, we draw enough samples (N many) from $\mathcal{D}_{\mathcal{G}(\hat{\varphi})}$. However, while drawing samples from $\mathcal{D}_{\mathcal{G}(\hat{\varphi})}$, it might be the case that many of the N drawn samples are $\tilde{0}$, which lowers the success probability of wFlash to infer about ideality of $\mathcal{G}$. So, we keep on drawing samples from $\mathcal{D}_{\mathcal{G}(\hat{\varphi})}$, until we get N many σ_1 and σ_2, upto M samples. Here, we set M in such a way that, if $\mathcal{G}$ is ε-close to ideal sampler, then with high probability we will get at least N many σ_1 and σ_2 from the M drawn samples. So, if wFlash receives more than $M - N$ many $\tilde{0}$, it returns REJECT without any further processing. Otherwise, it moves forward to check the similarity of σ_1 and σ_2 by applying the Bias subroutine. Finally, the algorithm returns REJECT if the Bias returned by the Bias subroutine is more than T, where T is a carefully chosen parameter.

Remark 4.1.2. One should note that, unlike Flash , in wFlash we cannot fix N and M beforehand, rather one has to reconsider the values of N and M in every iteration of wFlash . This is due to the fact that the weights of each assignment σ_1 and σ_2 changes as σ_1 and σ_2 alters, and consecutively, weight of $\tilde{0}$ alters in each of the t-many iterations. Thus we need to carefully determine the number of samples N needed to determine the bias properly and the number of total samples M needed to rule out enough many $\tilde{0}$. Also note that, both of these values N, M depends upon the weights of σ_1 and σ_2.

4.2 Theoretical Analysis of wFlash

Let us start by restating the theorem corresponding to our Weighted-Horn-sampler-tester wFlash .

Theorem 4.1.1 (Correctness of wFlash). *Given a Weighted-Horn-sampler $\mathcal{G}$, an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, a tolerance parameter $\varepsilon \in (0, \frac{1}{3}]$, an intolerance parameter $\eta >$*

```
Algorithm 5: wFlash ($\mathcal{G}$, $\mathcal{I}_{\mathcal{W}}$, $S$, $\varepsilon$, $\eta$, $\delta$, $\varphi$)
1  $t \leftarrow \frac{10}{\eta(\eta-9\varepsilon)} \log_e\left(\frac{1}{\delta}\right)$;
2  $n \leftarrow \log_e\left(\frac{2t}{\delta}\right)$;
3  $lo \leftarrow \frac{1+\varepsilon}{1-\varepsilon}$, $hi \leftarrow 1 + \frac{\eta+9\varepsilon}{4}$;
4  $\Gamma_1 \leftarrow \mathcal{G}(\varphi, S, t)$;
5  $\Gamma_2 \leftarrow \mathcal{I}_{\mathcal{W}}(\varphi, S, t)$;
6  for $i \leftarrow 1$ to $t$ do
7      $\sigma_1 \leftarrow \Gamma_1[i]$, $\sigma_2 \leftarrow \Gamma_2[i]$;
8      if $\sigma_1 == \sigma_2$ then
9          continue
10     $\alpha \leftarrow \frac{wt(\sigma_1)}{wt(\sigma_2)}$;
11     $L \leftarrow \frac{\alpha \cdot lo}{1+\alpha \cdot lo}$, $H \leftarrow \frac{\alpha \cdot hi}{1+\alpha \cdot hi}$;
12     $T = \frac{(H+L)}{2}$;
13     $N \leftarrow \frac{8n \cdot H}{(H-L)^2}$;
14     $X \leftarrow \left(\frac{(1-\varepsilon)(wt(\sigma_1)+wt(\sigma_2))}{(1-\varepsilon)(wt(\sigma_1)+wt(\sigma_2))+(1+\varepsilon)wt(\tilde{0})}\right)$;
15     $M \leftarrow \left(\frac{\sqrt{n}+\sqrt{n+4NX}}{2X}\right)^2$;
16     $\hat{\varphi} \leftarrow$ HornKernel $(\varphi, \sigma_1, \sigma_2)$;
17     $\Gamma_3 \leftarrow \mathcal{G}(\hat{\varphi}, S, M)$;
18     $\widehat{\Gamma}_3 \leftarrow$ RemoveZeros $(\Gamma_3)$;
19     if $\left|\widehat{\Gamma}_3\right| < N$ then
20         return REJECT
21     $Bias \leftarrow$ Bias $(\sigma_1, \widehat{\Gamma}_3, S)$;
22     if $Bias > T$ then
23         return REJECT
24 return ACCEPT
```

0, with $\eta > 9\varepsilon$ and a confidence parameter $\delta > 0$, and an arbitrary but fixed weight function wt, our Weighted-Horn-sampler-tester wFlash *takes $\widetilde{\mathcal{O}}(\frac{\mathsf{tilt}(wt,\varphi)^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2})$ many samples, and decides the following:*

(i) If $\mathcal{G}$ is ε-close to the Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, wFlash *outputs ACCEPT with probability at least $1-\delta$.*

(ii) If $\mathcal{G}$ is η-far from the Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, wFlash *outputs REJECT with probability at least $1-\delta$.*

where $\mathsf{tilt}(wt, \varphi)$ *denotes the maximum ratio between any two satisfying assignments of φ with respect to the weight function wt, and $\widetilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic factors in $\frac{1}{\eta}, \frac{1}{\eta-3\varepsilon}, \frac{1}{\eta-9\varepsilon}$, and $\frac{1}{\delta}$.*

Completeness Property of wFlash

Theorem 4.2.1 (Completeness Theorem)**.** *If the Horn sampler $\mathcal{G}$ is ε-close to an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then with probability at least $(1-\delta)$,* wFlash *will output ACCEPT .*

Proof of Theorem 4.2.1. To begin with, first note that, the algorithm of wFlash runs the for loop for t many times. To prove the Theorem 4.2.1, we will first show that if we have a Horn sampler $\mathcal{G}$ which is ε-close to an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then wFlash outputs REJECT in some i-th iteration of t loops with probability at most δ/t. Using the union bound, the proof follows. We divide the proof of the completeness theorem into Lemma 4.2.2 and Lemma 4.2.3.

The proof of Theorem 4.2.1 follows by first proving that if $\mathcal{G}$ is ε-close to an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then if we draw M samples from $\mathcal{G}(\hat{\varphi}, .)$, with probability at least $(1-\delta/2t)$, we will receive at least N samples from the set $\{\sigma_1, \sigma_2\}$. Then we have to show that if we receive N many samples from $\{\sigma_1, \sigma_2\}$, then the probability that $\mathcal{G}$ outputs REJECT in each iteration of wFlash is at most $\delta/2t$.

Lemma 4.2.2. *If the sampler $\mathcal{G}$ is ε-close to the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then when we draw M samples from $\mathcal{G}(\hat{\varphi}, .)$, the probability that $\widetilde{0}$ appears at least $(M-N)$ many times among M samples is at most $\frac{\delta}{2t}$.*

Lemma 4.2.3. *Given that $\tilde{0}$ has appeared less than $(M - N)$ times out of the M samples, the probability that* wFlash *outputs REJECT in any iteration is at most $\frac{\delta}{2t}$.*

Assuming Lemma 4.2.2 and Lemma 4.2.3 hold, we now prove Theorem 4.2.1 as follows:

First note that $\hat{\varphi}$ has three satisfying assignments: $\{\sigma_1, \sigma_2, \tilde{0}\}$ when projected onto the support set S. In each iteration, we are drawing M samples from $\mathcal{G}(\hat{\varphi}, .)$. We now define an index set $\mathcal{I}$ on Γ_3 as follows:

$$\mathcal{I} = \left\{ j \mid \Gamma_3[j]_{\downarrow S} \neq \tilde{0} \right\}$$

Note that there are two possibilities when wFlash outputs REJECT . They are the following:

Case (i) When we draw M samples in Line 17 of wFlash (Algorithm 5), we obtain more than $(M - N)$ many $\tilde{0}$, that is, $|\mathcal{I}| < N$. In this case, wFlash outputs REJECT .

Case (ii) When we draw M samples and get more than N samples from the set $\{\sigma_1, \sigma_2\}$ (that is, $|\mathcal{I}| \geq N$), but the *Bias* estimated by the Bias subroutine exceeds the threshold T.

So, the probability that wFlash outputs REJECT in an iteration, is given by,

$$\begin{aligned}
&\mathbb{P}\left(|\mathcal{I}| < N\right) + \mathbb{P}\left(Bias > T \mid |\mathcal{I}| \geq N\right) \cdot \mathbb{P}\left(|\mathcal{I}| \geq N\right) \\
&\leq \frac{\delta}{2t} + \frac{\delta}{2t} \cdot 1 \\
&= \frac{\delta}{t}
\end{aligned}$$

The first term in the first line corresponds to Case (i), while the second term corresponds to Case (ii) discussed above. The first inequality follows from Lemma 4.2.2 and Lemma 4.2.3.

Since the probability that wFlash outputs REJECT in some i-th iteration is at most $\frac{\delta}{t}$, then the probability that wFlash does not REJECT in any of the t iterations is at least $(1-\frac{\delta}{t})^t \geq 1-\delta$. So, when the sampler $\mathcal{G}$ is ε-close to the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, the algorithm wFlash outputs ACCEPT with probability at least $1-\delta$.

□

Proof of Lemma 4.2.2. Recall from the Algorithm 5 in Line 17, we sample M many samples from $\hat{\varphi}$, which are contained in Γ_3. So Γ_3 consists of witnesses from the set $\{\sigma_1, \sigma_2, \tilde{0}\}$. Let us first define the following binary random variable,

$$Z_j = \begin{cases} 1 & \text{if } \Gamma_3[j] = \tilde{0} \\ 0 & \text{otherwise} \end{cases}$$

Now, since the sampler $\mathcal{G}$ is ε-close to an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$ by our assumption, so we have the following inequalities,

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \tilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)} \leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \tilde{0})}{(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1)} \tag{4.1}$$

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \tilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \tilde{0})}{(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2)} \tag{4.2}$$

As $\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \tilde{0}) = 1$, we can say that:

$$\begin{aligned}
\mathbb{E}[Z_j] &= \mathbb{P}\left(Z_j = 1\right) \\
&= \mathbb{P}_{\mathcal{G}}\left(\hat{\varphi}, S, \tilde{0}\right) \\
&= \frac{\mathbb{P}_{\mathcal{G}}\left(\hat{\varphi}, S, \tilde{0}\right)}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \tilde{0})} \\
&\leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\hat{\varphi}, S, \tilde{0}\right)}{(1-\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\hat{\varphi}, S, \sigma_1\right) + (1-\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\hat{\varphi}, S, \sigma_2\right) + (1+\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\hat{\varphi}, S, \tilde{0}\right)} \\
&= \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \tilde{0}\right)}{(1-\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \sigma_1\right) + (1-\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \sigma_2\right) + (1+\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \tilde{0}\right)}
\end{aligned}$$

The inequality in fourth line follows due to the fact that if $\frac{a}{c} < \frac{l}{m}$ and $\frac{a}{c} < \frac{l}{n}$, then $\frac{a}{b+c} < \frac{l}{m+n}$ and $\frac{a}{a+b+c} < \frac{l}{l+m+n}$, along with Equation (4.1) and Equation (4.2).

Now we define the random variable Z as $Z = \sum_{j=1}^{|M|} Z_j$. Following the expression of $\mathbb{E}[Z_j]$, we can say the following:

$$\begin{aligned}
\mathbb{E}[Z] &\leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \tilde{0}\right) M}{(1-\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \sigma_1\right) + (1-\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \sigma_2\right) + (1+\varepsilon)\mathbb{P}_{\mathcal{I_W}}\left(\varphi, S, \tilde{0}\right)} \\
&= \frac{(1+\varepsilon)wt\left(\tilde{0}\right) M}{(1-\varepsilon)wt\left(\sigma_1\right) + (1-\varepsilon)wt\left(\sigma_2\right) + (1+\varepsilon)wt\left(\tilde{0}\right)}
\end{aligned}$$

Let us define τ as $\tau = (1-\varepsilon)wt\left(\sigma_1\right) + (1-\varepsilon)wt\left(\sigma_2\right) + (1+\varepsilon)wt\left(\tilde{0}\right)$. As $M = \left(\frac{\sqrt{n}+\sqrt{n+4NX}}{2X}\right)^2$, where $n = \log\frac{2t}{\delta}$, and $X = \left(\frac{(1-\varepsilon)wt(\sigma_1)+(1-\varepsilon)wt(\sigma_2)}{\tau}\right)$, we have $M = N + \mathbb{E}[Z] + \sqrt{Mn}$. Applying Lemma 2.3.2, we can say that $\mathbb{P}\left(|\mathcal{I}| < N\right) = \mathbb{P}\left(Z > M - N\right) = \mathbb{P}\left(Z > \mathbb{E}[Z] + \sqrt{Mn}\right) \leq \frac{\delta}{2t}$, and the lemma follows.

$\square$

Proof of Lemma 4.2.3. When M samples drawn from the sampler $\mathcal{G}$ in Line 17, if at least N many samples belong to the set $\{\sigma_1, \sigma_2\}$, then wFlash outputs REJECT if and only if $Bias$ is more then T. Let us now consider an iteration j among the t iterations of wFlash . Recall the set $\mathcal{I}$ on Γ_3 defined as $\mathcal{I} = \left\{j \mid \Gamma_3[j]_{\downarrow S} \neq \tilde{0}\right\}$. Consider the following binary random variable Y_j with $j \in \mathcal{I}$ defined as follows:

$$Y_j = \begin{cases} 1 & \text{if } \widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_1 \\ 0 & \text{if } \widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_2 \end{cases}$$

Then, we have the expected value of Y_j given by,

$$\mathbb{E}[Y_j] = \frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \tag{4.3}$$

Since $\mathcal{G}$ is ε-close to ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, from Definition 2.1.6, we can say that:

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1)}{(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2)} \tag{4.4}$$

Following the facts that $\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1) = \frac{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\sigma_1)}{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\sigma_1)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\sigma_2)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\tilde{0})}$ and $\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2) = \frac{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\sigma_2)}{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\sigma_1)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\sigma_2)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,S,\tilde{0})}$, from Equation (4.3) and Equation (4.4), we can say the following:

$$\begin{aligned}
\mathbb{E}[Y_j] &\leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1)}{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1) + (1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2)} \\
&= \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_1)}{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_1) + (1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_2)} \\
&= \frac{(1+\varepsilon)wt(\sigma_1)}{(1+\varepsilon)wt(\sigma_1) + (1-\varepsilon)wt(\sigma_2)} = L
\end{aligned}$$

The first inequality follows due to the fact that if $\frac{a}{b} < \frac{l}{m}$, then $\frac{a}{a+b} \leq \frac{l}{l+m}$, where $a, b, l, m \in \mathbb{R}$.

Now, $Bias$ is calculated in wFlash as follows:

$$Bias = \sum_{j \in [M]} \frac{\mathbb{I}(\widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_1)}{\mathbb{I}(\widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_1) + \mathbb{I}(\widehat{\Gamma}_3[j]_{\downarrow S} = \sigma_2)} = \sum_{j \in \mathcal{I}} \frac{Y_j}{|\mathcal{I}|}$$

Now given that $|\mathcal{I}| \geq N$, we can apply the Chernoff bound (Lemma 2.3.4) as follows:

$$\begin{aligned}
\mathbb{P}(Bias > T \mid |\mathcal{I}| \geq N) &= \mathbb{P}\left(\sum_{j \in \mathcal{I}} \frac{Y_j}{|\mathcal{I}|} > T \,\middle|\, |\mathcal{I}| \geq N \right) \\
&< \exp\left(-\frac{(T-L)^2 N}{2L} \right) = \exp\left(-\frac{(H-L)^2 N}{8L} \right) \\
&\leq \exp\left(-\frac{(H-L)^2 N}{8H} \right) \leq \frac{\delta}{2t}
\end{aligned}$$

□

Soundness Property of wFlash

Theorem 4.2.4 (Soundness Theorem)**.** *If the Horn sampler $\mathcal{G}$ is subquery consistent with respect to* HornKernel *and is η-far from an ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then with probability at least $(1-\delta)$,* wFlash *will output REJECT .*

Proof. Let us first partition the set of witnesses of φ into the following three disjoint sets as follows:

- $W_{-1} = \{x \in R_\varphi : \mathbb{P}_{\mathcal{G}}(\varphi, x) \leq \mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, x)\}$
- $W_0 = \{x \in R_\varphi : \mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, x) < \mathbb{P}_{\mathcal{G}}(\varphi, x) < \left(1 + \frac{\eta + 9\varepsilon}{4}\right) \mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, x)\}$
- $W_1 = \{x \in R_\varphi : \left(1 + \frac{\eta + 9\varepsilon}{4}\right) \mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, x) \leq \mathbb{P}_{\mathcal{G}}(\varphi, x)\}$

Now, we will show that if we receive σ_1 from W_1 and σ_2 from W_{-1}, then wFlash will output REJECT with high probability. In order to prove this, we need the following two lemmas.

Lemma 4.2.5. *If the sampler $\mathcal{G}$ is η-far from the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then*

$$\mathbb{P}\left(Bias > T \mid (\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1})\right) \geq \frac{4}{5}$$

Lemma 4.2.6. *If the sampler $\mathcal{G}$ is η-far from the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then*

$$\mathbb{P}\left(\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1}\right) \geq \frac{\eta(\eta - 9\varepsilon)}{8}$$

So, from Lemma 4.2.5 and Lemma 4.2.6, we can estimate the probability of rejection of wFlash in an iteration as follows:

$$\begin{aligned}
&\mathbb{P}\left(Bias > T\right) \\
&= \mathbb{P}\left(Bias > T \mid (\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1})\right) \cdot \mathbb{P}\left(\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1}\right) \\
&\geq \left(\frac{4}{5}\right) \frac{\eta(\eta - 9\varepsilon)}{8}
\end{aligned}$$

Thus, the probability that wFlash returns REJECT for is given by:

$$\begin{aligned}
1 - \prod_{i \in [t]} \mathbb{P}(Bias < T \text{ in } i\text{-th iteration}) &\geq 1 - \prod_{i \in [t]} \left(1 - \frac{\eta(\eta - 9\varepsilon)}{10}\right) \\
&= 1 - \left(1 - \frac{\eta(\eta - 9\varepsilon)}{10}\right)^t \\
&\geq 1 - \delta
\end{aligned}$$

□

Proof of Lemma 4.2.5. Assuming, $\sigma_1 \in W_1$ and $\sigma_2 \in W_{-1}$, we can obtain the following inequality

$$\begin{aligned}\frac{\mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_2)} &\geq \left(1 + \frac{\eta + 9\varepsilon}{4}\right) \cdot \frac{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_1)}{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_2)} \\ &= \left(1 + \frac{\eta + 9\varepsilon}{4}\right) \cdot \frac{wt(\sigma_1)}{wt(\sigma_2)}\end{aligned}$$

Thus, assuming the subquery consistent property of sampler $\mathcal{G}$, along with the fact that $\widehat{\Gamma}_3$ contains only σ_1 and σ_2, we can say the following:

$$\begin{aligned}\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) &= \frac{\mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\varphi, S, \sigma_2)} \\ &\geq \left(1 + \frac{\eta + 9\varepsilon}{4}\right) \cdot \frac{wt(\sigma_1)}{wt(\sigma_2)} \cdot \left(1 + \left(1 + \frac{\eta + 9\varepsilon}{4}\right) \cdot \frac{wt(\sigma_1)}{wt(\sigma_2)}\right)^{-1} \\ &= H\end{aligned}$$

Thus,

$$\begin{aligned}&\mathbb{P}\left(Bias \leq T \text{ in } i\text{-th iteration } \mid (\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1})\right) \\ &\leq \exp\left(-\frac{(H-L)^2 N}{8H}\right) \\ &\leq \frac{\delta}{2t} && \text{[applying Chernoff bound 2.3.4]} \\ &\leq \frac{1}{5} && [\text{as } \delta < 0.5 \text{ and } t \geq 2]\end{aligned}$$

So, the probability that wFlash returns REJECT when $\sigma_1 \in W_1$ and $\sigma_2 \in W_{-1}$ is,

$$\mathbb{P}\left(Bias > T \text{ in } i\text{-th iteration } \mid (\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1})\right) \geq \frac{4}{5}$$

$\square$

Proof of Lemma 4.2.6. Since the sampler $\mathcal{G}$ is η-far from the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, then for input φ, the ℓ_1 distance between $\mathcal{D}_{\mathcal{G}}(\varphi)$ and $\mathcal{D}_{\mathcal{I}_{\mathcal{W}}}(\varphi)$ is at least η. Thus,

$$\sum_{x\in W_0\cup W_1}(\mathbb{P}_{\mathcal{G}}(\varphi,x)-\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x))=\sum_{x\in W_{-1}}(\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x)-\mathbb{P}_{\mathcal{G}}(\varphi,x))\geq\frac{\eta}{2}\tag{4.5}$$

Therefore from Equation (4.5),

$$\sum_{x\in W_{-1}}\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x)\geq\frac{\eta}{2}\tag{4.6}$$

Now, from the definition of W_0, we have,

$$\sum_{x\in W_0}(\mathbb{P}_{\mathcal{G}}(\varphi,x)-\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x))<\frac{\eta+9\varepsilon}{4}\sum_{x\in W_0}\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x)<\frac{\eta+9\varepsilon}{4}$$

Hence, we have:

$$\sum_{x\in W_1}(\mathbb{P}_{\mathcal{G}}(\varphi,x)-\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x))\geq\frac{\eta}{2}-\frac{\eta+9\varepsilon}{4}=\frac{\eta-9\varepsilon}{4}$$

Therefore,

$$\sum_{x\in W_1}\mathbb{P}_{\mathcal{G}}(\varphi,x)\geq\frac{\eta}{2}-\frac{\eta+9\varepsilon}{4}=\frac{\eta-9\varepsilon}{4}\tag{4.7}$$

Since the events $\sigma_1\in W_1$ and $\sigma_2\in W_{-1}$ are independent, from Equation 4.6 and Equation 4.7, we can say that

$$\mathbb{P}\left(\sigma_1\in W_1\wedge\sigma_2\in W_{-1}\right)\geq\frac{\eta(\eta-9\varepsilon)}{8}$$

□

Sample Complexity of wFlash

Theorem 4.2.7. *The sample complexity of* wFlash *is* $\widetilde{\mathcal{O}}(\frac{\mathsf{tilt}(\varphi,wt)^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2})$*, where* tilt *denotes the maximum weight between any two satisfying assignments, where* $\widetilde{\mathcal{O}}(\cdot)$ *hides polylogarithmic factor in* $\frac{1}{\eta}$, $\frac{1}{\eta-9\varepsilon}$, $\frac{1}{\eta-3\varepsilon}$, *and* $\frac{1}{\delta}$.

Proof. For ease of presentation, we will represent $\mathsf{tilt}(\varphi, wt)$ as tilt. Note that wFlash (Algorithm 5) takes t samples in the for loop in Line 4. Also, it takes M samples in Line 5 in each iteration of the for loop. So, wFlash takes $2t + Mt$ samples from $\mathcal{G}$, which can be at most $2Mt$. Now, we find an upper bound of $2Mt$ below.

Note that

$$\begin{aligned}
\frac{1}{X} &= 1 + \frac{(1+\varepsilon)wt(\widetilde{0})}{(1-\varepsilon)(wt(\sigma_1)+wt(\sigma_2))} \\
&= 1 + \frac{1+\varepsilon}{1-\varepsilon}\frac{1}{\frac{wt(\sigma_1)}{wt(\widetilde{0})} + \frac{wt(\sigma_2)}{wt(\widetilde{0})}} \\
&\leq 1 + \frac{1+\varepsilon}{1-\varepsilon}\frac{\mathsf{tilt}}{2} \\
&\leq 1 + \mathsf{tilt} && (\because \varepsilon \leq \frac{1}{3}) \\
&\leq 2\ \mathsf{tilt} && (\because 1 \leq \mathsf{tilt})
\end{aligned}$$

Note that $N = 8\log\frac{2t}{\delta}\frac{H}{(H-L)^2} \leq 8\log\frac{2t}{\delta}\left(\frac{\mathsf{tilt}}{(\eta-3\varepsilon)}\right)^2$. Thus, we can say that

$$\begin{aligned}
2Mt &= 2t\left(\log\frac{2t}{\delta}\frac{1}{X^2} + \frac{4N}{X}\right) \\
&\leq 2t\left(\log\frac{2t}{\delta}\cdot 4\ \mathsf{tilt}^2 + 32\log\frac{2t}{\delta}\frac{\mathsf{tilt}^3}{(\eta-3\varepsilon)^2}\right) \\
&\leq \widetilde{\mathcal{O}}\left(t\cdot\frac{\mathsf{tilt}^3}{(\eta-3\varepsilon)^2}\right) \\
&\leq \widetilde{\mathcal{O}}\left(\frac{\mathsf{tilt}^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2}\right)
\end{aligned}$$

So, the total sample complexity of wFlash is $\widetilde{\mathcal{O}}\left(\frac{\mathsf{tilt}^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2}\right)$. This completes the proof of Theorem 4.2.7.

□

Chapter 5

Evaluation Results

The main objective of our work is to evaluate the two following questions:

RQ1 Can Flash and wFlash check if of-the-shelf samplers are ε-close or η-far from ideal samplers?

RQ2 What kind of improvements are possible over the baseline?

To evaluate the practical effectiveness of Flash and wFlash , we implemented the prototypes of Flash and wFlash in Python 3.8.3. All the experiments are carried out on a high-performance computer cluster, where each node consists of E5-2690 v3 @2.60GHz CPU with 24 cores and 4GB memory per core. For each benchmark-sampler pair, one single core is being employed with a maximum time limit of 23 hrs 50 minutes. The detailed logs and the runtime code employed to run the experiments are given in the supplementary material.

5.1 Samplers Tested

To remain consistent with prior works, we follow the setup described in Barbarik and Barbarik2 . We employ the following state of the art samplers: UniGen3 [27], QuickSampler [17], STS [18] [1]. For experiments with Weighted-Horn-sampler-tester, we augment these samplers with an inverse sampling module [2]. We shall term the newly generated samplers as wUniGen, wQuickSampler, wSTS respectively in Table 5.2. Furthermore, while implementing Flash (resp. wFlash), our algorithm requires the access of a known ideal Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$ (resp. ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$) beforehand. We use SPUR [1] as the corresponding ideal Uniform-Horn-sampler, and augment it by inverse sampling to achieve ideal Weighted-Horn-sampler needed for wFlash .

5.2 Test Parameters

For both of our experiments with Flash and wFlash , the tolerance parameter ε, intolerance parameter η, and confidence parameter δ are set to be 0.1, 1.6, and 0.1 respectively. This implies that both Flash and wFlash outputs ACCEPT when the sampler $\mathcal{G}$ to be tested is ε-close to the ideal Uniform-Horn-sampler and ideal Weighted-Horn-sampler respectively with probability at least $1-\delta$. Similarly, with probability at least $1-\delta$, Flash and wFlash outputs REJECT when $\mathcal{G}$ is η-far from ideal Uniform-Horn-sampler and ideal Weighted-Horn-sampler respectively.

[1]We use the default parameters for Quicksampler, STS and UniGen, which were employed in the previous studies[9][22] to maintain the consistency of the experiments.

[2]Inverse sampling, basically, converts the (φ, wt) pair to another formula $\hat{\varphi}$ which preserves the distribution of the satisfying assignments.

5.3 Benchmarks

Our benchmark suites consist of formulas arising from the reliability computation of power transmission networks in US cities [16]. For weighted sampler testing, we only consider log-linear distributions. They plays a crucial role in several machine learning algorithms. A formal discussion on log-linear distribution is presented in the supplementary material. In particular, the weight functions for log-linear distributions can be specified using weights on literals. For each of the benchmark instances of unweighted Horn formulas, we designed two sets of weight functions and thus, we procured two sets of benchmarks as depicted in Table 5.2. In both these benchmarks, we sample a set of literals from the support set and have random nontrivial weights [3] for the sampled literals and assign weight 0.5 to rest of the literals. For the first set of benchmarks, (presented in the top half of Table 5.2), we pick each literal with probability 1/3, while for the second set of benchmarks, we uniformly sample a constant (12) number of literals. Note that assigning the weights of all the literals as 0.5 is equivalent to uniform sampling.

5.4 Description of the Tables

Table 5.1 and Table 5.2 depict our experiments of Flash and wFlash respectively. The 2nd and 3rd columns of the Table 5.1 indicate the number of variables and clauses in the benchmark instances. In the 4th, 5th and 6th columns of Table 5.1 , we present the experimental results of Flash on UniGen, QuickSampler and STS respectively. In the 2nd, 3rd and 4th columns of Table 5.2, we present the experimental results of wFlash on wUniGen, wQuickSampler and wSTS respectively. In each of these cells, A and R indicate whether the output of Flash was ACCEPT or REJECT respectively, and the

[3]Non-trivial weights are of the form $k/2^m$. We have chosen $m = 4$ and k is set randomly to either 7 or 9.

number on the right indicates the number of samples drawn by the tester in that instance. DNS denotes the situation where the sampler-under-test has failed to sample any sample during the period of run-time, and TLE denotes the situation where wFlash is unable to complete the test within the time limit of the experiment. It is worth noting that DNS is caused due to the failure of the sampler-under-test to draw satisfying assignments from a formula. But TLE indicates a combined failure of both the sampler-under-test and our verification algorithm.

	UniGen		QuickSampler		STS	
Benchmark	o/p	#Samples	o/p	#Samples	o/p	#Samples
`Net6_count_91`	A	218505	R	52025	R	20810
`Net8_count_96`	A	218505	R	166480	R	31215
`Net12_count_106`	A	218505	R	72835	R	52025
`Net22_count_116`	A	218505	R	72835	R	41620
`Net27_count_118`	A	218505	R	72835	R	10405
`Net29_count_164`	A	218505	R	114455	R	20810
`Net39_count_240`	A	218505	R	114455	R	114455
`Net43_count_243`	A	218505	R	93645	R	114455
`Net46_count_322`	A	218505	R	10405	R	10405
`Net52_count_362`	A	218505	R	10405	R	20810
`Net53_count_339`	A	218505	R	31215	R	72835

Table 5.1: Evaluation results of Flash

Detailed Results

RQ1 From our experiments, we find that in all the 11 benchmark instances, Flash outputs REJECT for STS and QuickSampler. On the other hand, Flash outputs ACCEPT for UniGen in all 11 instances. In this context, it is worth highlighting that the samplers STS and QuickSampler are designed based on heuristic techniques, while Unigen has sound theoretical guarantees.

Out of the 22 instances, wFlash outputs REJECT in all the instances of wSTS. When run with wQuickSampler, wFlash outputs REJECT in 21 instances, while it outputs

	wUniGen		wQuickSampler		wSTS	
Benchmark	o/p	#Samples	o/p	#Samples	o/p	#Samples
Net6_count_91_w1	TLE	-	R	106910	R	15626
Net8_count_96_w1	TLE	-	R	22716	R	39944
Net12_count_106_w1	TLE	-	R	27428	R	41334
Net22_count_116_w1	DNS	-	R	98629	R	9217
Net27_count_118_w1	DNS	-	R	49654	R	25296
Net29_count_164_w1	DNS	-	R	123202	R	12322
Net39_count_240_w1	DNS	-	R	7745	R	7922
Net43_count_243_w1	DNS	-	R	209062	R	22351
Net46_count_322_w1	DNS	-	R	23105	R	7922
Net52_count_362_w1	DNS	-	R	6085	R	8650
Net53_count_339_w1	DNS	-	R	38417	R	23105
Net6_count_91_w2	A	274175	R	17667	R	26995
Net8_count_96_w2	A	397169	A	388885	R	16385
Net12_count_106_w2	A	197713	R	6085	R	5930
Net22_count_116_w2	A	302546	R	22947	R	24561
Net27_count_118_w2	TLE	-	R	10405	R	26245
Net29_count_164_w2	A	238673	R	7226	R	17706
Net39_count_240_w2	A	282138	R	13690	R	14885
Net43_count_243_w2	TLE	-	R	238260	R	9217
Net46_count_322_w2	A	437529	R	135368	R	30819
Net52_count_362_w2	TLE	-	R	210925	R	23127
Net53_count_339_w2	A	191806	R	8650	R	9605

Table 5.2: Evaluation results of wFlash

ACCEPT in 1 instance. When wFlash is run with wUniGen, wFlash outputs ACCEPT for 8 instances, while there are 6 cases of TLE and 8 instances of DNS. For these instances wFlash had demanded a very high volume of samples that wUniGen failed to provide within the given time limit.

RQ2 The number of samples required by the baseline approach, following the work of Batu et.al [5], is extremely high. We estimate the average time taken by a sampler for particular instances of our benchmarks. Using our estimate, we observe that the time taken by our baseline would be over 10^{12} seconds for all 11 benchmarks for UniGen, STS and Quicksampler. In this context, it is worth highlighting that Flash terminates within 24 hours for all the instances for all the samplers - hence the have massive (over 10^7) speed up in the runtime compared to the baseline for all the instances of the benchmark.

5.5 Extended Experimental Results

In this section, we describe the extended experimental results of our Uniform-Horn-sampler-tester Flash and Weighted-Horn-sampler-tester wFlash . As mentioned previously, we will use DNS to denote the situation where sampler-under-test has failed to sample during the time period, and TLE to denote when the tester has not been able to complete the test within the time limit of the experiment. For each benchmark-sampler pair, one single core is being employed with a maximum time limit of 23 hrs 50 minutes, where the experiments have been carried out on a high-performance computer cluster, where each node consists of E5-2690 v3 @2.60GHz CPU with 24 cores and 4GB memory per core.

We also computed the model count, that is, the number of satisfying assignments of the benchmark Horn formulas using the tool sharpSAT [29]. We would like to point out that as we are using the same set of benchmarks for the Horn samplers UniGen, QuickSampler, and STS, the model count remains same in all the tables corresponding to the Uniform-Horn-sampler-tester Flash . Moreover, in the evaluation of our Weighted-Horn-sampler-tester wFlash , as we are considering the same 11 benchmark instances with respect to two different weight functions, model count of the benchmark instances remain same there as well. For completeness purpose, we are presenting them in each table.

5.5.1 Results of our Uniform-Horn-sampler-tester Flash

Here we present the extended results of our Uniform-Horn-sampler-tester Flash . For baseline, we employed the tester from [3, 24]. For any Horn formula φ, it takes $\mathcal{O}(\sqrt{n}(\eta - \varepsilon)^{-2}\log(\frac{n}{\delta}))$ many samples, where n denotes the model count of φ, and ε, η and δ are the closeness, farness and confidence parameters respectively. Due to the extremely large sample complexity, it is not feasible to compute the exact time for the baseline approach. Thus, we have estimated the average time taken for any benchmark instance.

In Table 5.3, we compare our Uniform-Horn-sampler-tester Flash with respect to the sampler UniGen to the baseline over the 11 benchmark instances. 1st column of Table 5.3 represents the name of the benchmark instance and 2nd column corresponds to the model count of the Horn formula corresponding to that instance. In 3rd and 4th columns, we present the number of samples and time required by the baseline tester, whereas 5th, 6th and 7th columns represent the number of samples and the time required by Flash , and the output of Flash with respect to that particular benchmark instance. We find that Flash outputs ACCEPT in all 11 benchmark instances. As it is evident from the entries of the table, Flash vastly outperforms the baseline approach, both in terms of the number of samples required, as well as the time required for testing.

Similarly, in Table 5.4, we compare our Uniform-Horn-sampler-tester Flash when run with the sampler QuickSampler along with the baseline tester over the 11 benchmark instances. Similar to Table 5.3, 1st and 2nd columns represent the benchmark instance name and the model count, whereas 3rd and 4th columns correspond to the number of samples and time required by the baseline tester, and 5th, 6th and 7th columns represent the number of samples and total time required by Flash , and the output of Flash on that instance. It turns out that Flash outputs REJECT in all the instances with respect to QuickSampler. As in the case of UniGen, the number of samples required, and the total time taken by Flash is much smaller compared to the baseline approach.

Finally, in Table 5.5, we present our results when we run our Uniform-Horn-sampler-tester Flash for the sampler STS. The organization of this table follows in similar fashion as Table 5.3 and Table 5.4. As in the case of QuickSampler, Flash outputs REJECT in all 11 instances here as well. Similar to the case of UniGen and QuickSampler, it is clear that Flash outperforms the baseline approach by several orders of magnitude both in the context of number of samples required, as well as the time taken to complete the experiment.

		Baseline		Flash		
Benchmark	Model Count	#Samples	Time (s)	#Samples	Time (s)	o/p
Net6_count_91	2.19×10^{32}	7.72×10^{18}	3.41×10^{17}	218505	643	A
Net8_count_96	3.2×10^{36}	9.9×10^{19}	4.7×10^{18}	218505	664	A
Net12_count_106	6.34×10^{43}	5.27×10^{23}	3.15×10^{22}	218505	709	A
Net22_count_116	9.49×10^{50}	2.36×10^{27}	1.73×10^{26}	218505	756	A
Net27_count_118	8.05×10^{53}	7.27×10^{28}	5.46×10^{27}	218505	537	A
Net29_count_164	4.51×10^{63}	6.41×10^{33}	1.02×10^{33}	218505	671	A
Net39_count_240	2.46×10^{91}	6.77×10^{47}	2.61×10^{47}	218505	1002	A
Net43_count_243	8.41×10^{100}	4.36×10^{52}	1.75×10^{52}	218505	1045	A
Net46_count_322	3.22×10^{129}	1.09×10^{67}	1.12×10^{67}	218505	1491	A
Net52_count_362	2.64×10^{147}	1.12×10^{76}	1.24×10^{76}	218505	1793	A
Net53_count_339	4.05×10^{143}	1.36×10^{74}	1.17×10^{74}	218505	1622	A

Table 5.3: Evaluation results of Flash with UniGen

		Baseline		Flash		
Benchmark	Model Count	#Samples	Time (s)	#Samples	Time (s)	o/p
Net6_count_91	2.19×10^{32}	7.72×10^{18}	1.79×10^{16}	52025	54	R
Net8_count_96	3.2×10^{36}	9.91×10^{19}	2.16×10^{17}	166480	182	R
Net12_count_106	6.34×10^{43}	5.27×10^{23}	1.19×10^{21}	72835	88	R
Net22_count_116	9.49×10^{50}	2.36×10^{27}	6.06×10^{24}	72835	94	R
Net27_count_118	8.05×10^{53}	7.27×10^{28}	1.86×10^{26}	72835	97	R
Net29_count_164	4.51×10^{63}	6.41×10^{3}	2.08×10^{31}	114455	210	R
Net39_count_240	2.46×10^{91}	6.76×10^{47}	3.23×10^{45}	166480	477	R
Net43_count_243	8.41×10^{100}	4.36×10^{52}	2.04×10^{50}	93645	278	R
Net46_count_322	3.22×10^{129}	1.09×10^{67}	6.68×10^{64}	10405	44	R
Net52_count_362	2.64×10^{147}	1.12×10^{76}	7.7×10^{73}	10405	56	R
Net53_count_339	4.05×10^{143}	1.36×10^{74}	8.74×10^{71}	31215	145	R

Table 5.4: Evaluation results of Flash with QuickSampler

5.5.2 Results of our Weighted-Horn-sampler-tester wFlash

In this section, we present the extended results of the experiments of our Weighted-Horn-sampler-tester wFlash to test the samplers wUniGen, wQuickSampler and wSTS. For each of the 11 benchmark instances of unweighted Horn formulas, we designed two sets of weight functions and thus, we procured two sets of benchmarks, and we run our experiments over these 22 benchmark instances.

For baseline, we employed the tester of Batu et.al [4]. For any Horn formula φ, the

		Baseline		Flash		
Benchmark	Model Count	#Samples	Time (s)	#Samples	Time (s)	o/p
Net6_count_91	2.19×10^{32}	7.72×10^{18}	1.68×10^{16}	20810	16	R
Net8_count_96	3.2×10^{36}	9.91×10^{19}	2.45×10^{17}	31215	26	R
Net12_count_106	6.34×10^{43}	5.27×10^{23}	1.7×10^{21}	52025	51	R
Net22_count_116	9.49×10^{50}	2.36×10^{27}	9.17×10^{24}	41620	43	R
Net27_count_118	8.05×10^{53}	7.27×10^{28}	3.12×10^{26}	10405	12	R
Net29_count_164	4.51×10^{63}	6.41×10^{33}	4.05×10^{31}	20810	30	R
Net39_count_240	2.46×10^{91}	6.76×10^{47}	8.94×10^{45}	114455	310	R
Net43_count_243	8.41×10^{100}	4.36×10^{52}	6.49×10^{50}	114455	273	R
Net46_count_322	3.22×10^{129}	1.09×10^{67}	2.65×10^{65}	10405	35	R
Net52_count_362	2.64×10^{147}	1.12×10^{76}	3.33×10^{74}	20810	97	R
Net53_count_339	4.05×10^{143}	1.36×10^{74}	3.68×10^{72}	72835	267	R

Table 5.5: Evaluation results of Flash with STS

tester of [4] takes $\mathcal{O}(n^{\frac{2}{3}}(\eta-\varepsilon)^{-\frac{8}{3}}\log(\frac{n}{\delta}))$ many samples, where n denotes the model count of φ, and ε, η and δ are the closeness, farness and confidence parameters respectively. Similar to the case of Flash , as the sample complexity of [4] is very large, we have estimated the average time taken for any benchmark instance by the baseline tester.

In Table 5.6, we present the results of our Weighted-Horn-sampler-tester wFlash to test the sampler wUniGen. 1st column of Table 5.6 denotes the name of the benchmark instance, 2nd column corresponds to the model count of the Horn formula corresponding to the benchmark instance, and 3rd column represents the tilt of the formula corresponding to the instance, where tilt denotes the maximum ratio between the weights of any two satisfying assignments. 4th and 5th columns represent the number of samples and time required by the baseline tester respectively. 6th, 7th and 8th columns corresponds to the number of samples and time required by wFlash and its output on that benchmark instance respectively. Among the 22 benchmark instances, in 8 instances, wFlash outputs ACCEPT , whereas there are 6 instances of TLE and 8 instances of DNS. For the instances where wFlash does not output TLE or DNS, it is clear that wFlash outperforms the baseline tester by a large order of magnitude.

In Table 5.7, we present our results of wFlash with respect to wQuickSampler. The

			Baseline		wUniGen		
Benchmark	Model Count	tilt	#Samples	Time (s)	#Samples	Time (s)	o/p
Net6_count_91_w1	2.19×10^{32}	20.40	3.12×10^{24}	TLE	-	TLE	TLE
Net8_count_96_w1	3.2×10^{36}	26.23	9.18×10^{25}	TLE	-	TLE	TLE
Net12_count_106_w1	6.34×10^{43}	20.40	8.03×10^{30}	TLE	-	TLE	TLE
Net22_count_116_w1	9.49×10^{50}	26.23	5.65×10^{35}	TLE	-	DNS	DNS
Net27_count_118_w1	8.05×10^{53}	43.36	5.35×10^{37}	TLE	-	DNS	DNS
Net29_count_164_w1	4.51×10^{63}	92.17	1.98×10^{44}	TLE	-	DNS	DNS
Net39_count_240_w1	2.46×10^{91}	14043.96	8.81×10^{62}	TLE	-	DNS	DNS
Net43_count_243_w1	8.41×10^{100}	1137.74	2.20×10^{69}	TLE	-	DNS	DNS
Net46_count_322_w1	3.22×10^{129}	23215.53	3.81×10^{88}	TLE	-	DNS	DNS
Net52_count_362_w1	2.64×10^{147}	286565.21	3.19×10^{100}	TLE	-	DNS	DNS
Net53_count_339_w1	4.05×10^{143}	38376.70	8.92×10^{97}	TLE	-	DNS	DNS
Net6_count_91_w2	2.19×10^{32}	12.34	3.12×10^{24}	4.57×10^{23}	274175	3921	A
Net8_count_96_w2	3.2×10^{36}	5.80	9.18×10^{25}	1.35×10^{25}	397169	5837	A
Net12_count_106_w2	6.34×10^{43}	5.80	8.03×10^{30}	1.42×10^{30}	197713	3022	A
Net22_count_116_w2	9.49×10^{50}	7.46	5.65×10^{35}	1.04×10^{35}	302546	4551	A
Net27_count_118_w2	8.05×10^{53}	7.46	5.35×10^{37}	TLE	-	TLE	TLE
Net29_count_164_w2	4.51×10^{63}	7.46	1.98×10^{44}	6.36×10^{43}	238673	3986	A
Net39_count_240_w2	2.46×10^{91}	9.60	8.81×10^{62}	5.83×10^{62}	282138	5909	A
Net43_count_243_w2	8.41×10^{100}	4.51	2.20×10^{69}	TLE	-	TLE	TLE
Net46_count_322_w2	3.22×10^{129}	5.80	3.21×10^{88}	3.81×10^{88}	437529	5038	A
Net52_count_362_w2	2.64×10^{147}	2.73	3.19×10^{100}	TLE	-	TLE	TLE
Net53_count_339_w2	4.05×10^{143}	7.46	8.92×10^{97}	1.12×10^{98}	191806	2933	A

Table 5.6: Evaluation results of wFlash with wUniGen

organization of this table is similar to that of Table 5.6. Our Weighted-Horn-sampler-tester wFlash outputs REJECT in 21 instances out of the 22 instances, and outputs ACCEPT in 1 benchmark instance. Moreover, as in the case of wUniGen, the sample complexity and total time taken by wFlash is much better compared to the baseline approach.

Finally, in Table 5.8, we show the results of wFlash in order to test the sampler STS. The organization of the table follows in similar line to the preceding tables. It turns out that wFlash outputs REJECT in all of the 22 benchmark instances here. Also, similar to wUniGen and wQuickSampler, wFlash outperforms the baseline tester by a large magnitude.

			Baseline		wQuickSampler		
Benchmark	Model Count	tilt	#Samples	Time (s)	#Samples	Time (s)	o/p
Net6_count_91_w1	2.19×10^{32}	20.40	3.11×10^{24}	1.33×10^{22}	106910	158	R
Net8_count_96_w1	3.2×10^{36}	26.23	9.18×10^{25}	3.94×10^{23}	22716	37	R
Net12_count_106_w1	6.34×10^{43}	20.40	8.03×10^{30}	3.22×10^{28}	27428	48	R
Net22_count_116_w1	9.49×10^{50}	26.23	5.66×10^{35}	2.73×10^{33}	98629	182	R
Net27_count_118_w1	8.05×10^{53}	43.36	5.35×10^{37}	2.54×10^{35}	49654	94	R
Net29_count_164_w1	4.51×10^{63}	92.17	1.98×10^{44}	1.53×10^{42}	123202	337	R
Net39_count_240_w1	2.46×10^{91}	14043.96	8.81×10^{62}	1.04×10^{61}	7745	54	R
Net43_count_243_w1	8.41×10^{100}	1137.74	2.20×10^{69}	2.37×10^{67}	209062	934	R
Net46_count_322_w1	3.22×10^{129}	23215.53	3.21×10^{88}	3.89×10^{86}	23105	174	R
Net52_count_362_w1	2.64×10^{147}	286565.21	3.19×10^{100}	4.65×10^{98}	6085	99	R
Net53_count_339_w1	4.05×10^{143}	38376.70	8.92×10^{97}	1.13×10^{96}	38417	331	R
Net6_count_91_w2	2.19×10^{32}	12.34	3.12×10^{24}	9.1×10^{21}	17667	23	R
Net8_count_96_w2	3.2×10^{36}	5.80	9.18×10^{25}	2.83×10^{23}	388885	486	A
Net12_count_106_w2	6.34×10^{43}	5.80	8.02×10^{30}	2.98×10^{28}	6085	10	R
Net22_count_116_w2	9.49×10^{50}	7.46	5.66×10^{35}	2.08×10^{33}	22947	36	R
Net27_count_118_w2	8.05×10^{53}	7.46	5.35×10^{37}	1.89×10^{35}	10405	16	R
Net29_count_164_w2	4.51×10^{63}	7.46	1.98×10^{44}	8.94×10^{41}	7226	17	R
Net39_count_240_w2	2.46×10^{91}	9.60	8.81×10^{62}	5.96×10^{60}	13690	43	R
Net43_count_243_w2	8.41×10^{100}	4.51	2.20×10^{69}	1.45×10^{67}	238260	765	R
Net46_count_322_w2	3.22×10^{129}	5.80	3.21×10^{88}	2.56×10^{86}	135368	592	R
Net52_count_362_w2	2.64×10^{147}	2.73	3.19×10^{100}	2.98×10^{98}	210925	1138	R
Net53_count_339_w2	4.05×10^{143}	7.46	8.92×10^{97}	7.23×10^{95}	8650	43	R

Table 5.7: Evaluation results of wFlash with wQuickSampler

			Baseline		wSTS		
Benchmark	Model Count	tilt	#Samples	Time (s)	#Samples	Time (s)	o/p
Net6_count_91_w1	2.19×10^{32}	20.40	3.12×10^{24}	2.15×10^{22}	15626	26	R
Net8_count_96_w1	3.2×10^{36}	26.23	9.18×10^{25}	6.27×10^{23}	39944	73	R
Net12_count_106_w1	6.34×10^{43}	20.40	8.03×10^{30}	5.92×10^{28}	41334	82	R
Net22_count_116_w1	9.49×10^{50}	26.23	5.66×10^{35}	5.32×10^{33}	9217	22	R
Net27_count_118_w1	8.05×10^{53}	43.36	5.35×10^{37}	5.2×10^{35}	25296	64	R
Net29_count_164_w1	4.51×10^{63}	92.17	1.98×10^{44}	3.2×10^{42}	12322	41	R
Net39_count_240_w1	2.46×10^{91}	14043.96	8.81×10^{62}	3.6×10^{61}	7922	77	R
Net43_count_243_w1	8.41×10^{100}	1137.74	2.2×10^{69}	8.04×10^{67}	22351	165	R
Net46_count_322_w1	3.22×10^{129}	23215.53	3.21×10^{88}	2.04×10^{87}	7922	91	R
Net52_count_362_w1	2.64×10^{147}	286565.21	3.19×10^{100}	2.63×10^{99}	8650	153	R
Net53_count_339_w1	4.05×10^{143}	38376.70	8.92×10^{97}	6.88×10^{96}	23105	331	R
Net6_count_91_w2	2.19×10^{32}	12.34	3.12×10^{24}	1.36×10^{22}	26995	30	R
Net8_count_96_w2	3.2×10^{36}	5.80	9.18×10^{25}	4.07×10^{23}	16385	21	R
Net12_count_106_w2	6.34×10^{43}	5.80	8.03×10^{30}	4.7×10^{28}	5930	8	R
Net22_count_116_w2	9.49×10^{50}	7.46	5.66×10^{35}	3.83×10^{33}	24561	36	R
Net27_count_118_w2	8.05×10^{53}	7.46	5.35×10^{37}	3.91×10^{35}	26245	37	R
Net29_count_164_w2	4.51×10^{63}	7.46	1.98×10^{44}	2.02×10^{42}	17706	33	R
Net39_count_240_w2	2.46×10^{91}	9.60	8.81×10^{62}	1.62×10^{61}	14885	35	R
Net43_count_243_w2	8.41×10^{100}	4.51	2.2×10^{69}	4.53×10^{67}	9217	26	R
Net46_count_322_w2	3.22×10^{129}	5.80	3.21×10^{88}	1.04×10^{87}	30819	98	R
Net52_count_362_w2	2.64×10^{147}	2.73	3.19×10^{100}	1.28×10^{99}	23127	100	R
Net53_count_339_w2	4.05×10^{143}	7.46	8.92×10^{97}	3.04×10^{96}	9605	38	R

Table 5.8: Evaluation results of wFlash with wSTS

Chapter 6

Conclusion

We designed and implemented the first Horn-sampler-testers Flash and wFlash that have sound theoretical guarantees. They also work well in practice, as described by the evaluation results with respect to three state-of-the-art samplers UniGen, QuickSampler, and STS, along with their weighted counterparts wUniGen, wQuickSampler, and wSTS. To best of our knowledge Flash and wFlash are the first testing frameworks for checking reliability of the Uniform-Horn-sampler and Weighted-Horn-samplerȦpart from Horn, the other classes of CNF like 2-SAT, Dual-Horn and some non-CNF classes like XOR-CNF are of keen interest in various fields. Thus coming up with testing frameworks exclusively for such classes could give a new direction to this research.

Bibliography

[1] Dimitris Achlioptas, Zayd S Hammoudeh, and Panos Theodoropoulos. Fast sampling of perfectly uniform satisfying assignments. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 135–147. Springer, 2018.

[2] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM Journal on Computing*, 35(1):132–150, 2005.

[3] Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 442–451. IEEE, 2001.

[4] Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE, 2000.

[5] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 60(1):1–25, 2013.

[6] Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 381–390, 2004.

[7] Clément L Canonne, Dana Ron, and Rocco A Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015.

[8] Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM Journal on Computing*, 45(4):1261–1296, 2016.

[9] Sourav Chakraborty and Kuldeep S Meel. On testing of uniform samplers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7777–7784, 2019.

[10] Supratik Chakraborty, Daniel J Fremont, Kuldeep S Meel, Sanjit A Seshia, and Moshe Y Vardi. On parallel scalable uniform sat witness generation. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 304–319. Springer, 2015.

[11] Supratik Chakraborty, Dror Fried, Kuldeep S Meel, and Moshe Y Vardi. From weighted to unweighted model counting. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[12] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

[13] Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Information and computation*, 125(1):1–12, 1996.

[14] Emanuele De Angelis, Fabio Fioravanti, Alberto Pettorossi, and Maurizio Proietti. Verifying relational program properties by transforming constrained horn clauses. In *CILC*, pages 69–85. Citeseer, 2016.

[15] Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[16] Leonardo Duenas-Osorio, Kuldeep Meel, Roger Paredes, and Moshe Vardi. Counting-based reliability estimation for power-transmission grids. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[17] Rafael Dutra, Kevin Laeufer, Jonathan Bachrach, and Koushik Sen. Efficient sampling of sat solutions for testing. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 549–559. IEEE, 2018.

[18] Stefano Ermon, Carla P Gomes, and Bart Selman. Uniform solution sampling using a constraint solver as an oracle. *arXiv preprint arXiv:1210.4861*, 2012.

[19] Graeme Gange, Jorge A Navas, Peter Schachte, Harald Søndergaard, and Peter J Stuckey. Horn clauses as an intermediate representation for program analysis and transformation. *Theory and Practice of Logic Programming*, 15(4-5):526–542, 2015.

[20] Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer, 2011.

[21] Johann A. Makowsky. Why horn formulas matter in computer science: Initial structures and generic examples. *Journal of Computer and System Sciences*, 34(2-3):266–292, 1987.

[22] Kuldeep S Meel, Yash Pralhad Pote, and Sourav Chakraborty. On testing of samplers. *Advances in Neural Information Processing Systems*, 33:5753–5763, 2020.

[23] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[24] Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

[25] Yash Pralhad Pote and Kuldeep S Meel. Testing probabilistic circuits. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.

[27] Mate Soos, Stephan Gocht, and Kuldeep S Meel. Tinted, detached, and lazy cnf-xor solving and its applications to counting and sampling. In *International Conference on Computer Aided Verification*, pages 463–484. Springer, 2020.

[28] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.

[29] Marc Thurley. sharpsat–counting models with advanced component caching and implicit bcp. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 424–429. Springer, 2006.

[30] Gregory Valiant and Paul Valiant. Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new clts. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 685–694, 2011.

[31] Gregory Valiant and Paul Valiant. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 403–412. IEEE, 2011.

[32] Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.