Testing of Horn Samplers [1]

Ansuman Banerjee
Indian Statistical Institute, Kolkata

Shayak Chakraborty
Microsoft Corporation

Sourav Chakraborty
Indian Statistical Institute, Kolkata

Kuldeep S. Meel
National University of Singapore

Uddalok Sarkar
Indian Statistical Institute, Kolkata

Sayantan Sen
Indian Statistical Institute, Kolkata

Abstract

Sampling over combinatorial spaces is a fundamental problem in artificial intelligence with a wide variety of applications. Since state-of-the-art techniques heavily rely on heuristics whose rigorous analysis remain beyond the reach of current theoretical tools, the past few years have witnessed interest in the design of techniques to test the quality of samplers. The current state-of-the-art techniques, Barbarik and Barbarik2, focus on the cases where combinatorial spaces are encoded as Conjunctive Normal Form (CNF) formulas. While CNF is a general-purpose form, often techniques rely on exploiting specific representations to achieve speedup. Of particular interest are Horn clauses, which form the basis of the logic programming tools in AI. In this context, a natural question is whether it is possible to design a tester that can determine the correctness of a given Horn sampler. The primary contribution of this paper is an affirmative answer to the above question. We design the first tester, Flash, which tests the correctness of a given Horn sampler: given a specific distribution $\mathcal{I}$ and parameters η, ε, and δ, the tester Flash correctly (with probability at least $1-\delta$) distinguishes whether the underlying distribution of the Horn-sampler is "ε-close" to $\mathcal{I}$ or "η-far" from $\mathcal{I}$ by sampling only $\widetilde{\mathcal{O}}(\mathsf{tilt}^3/(\eta-\varepsilon)^4)$ samples from the Horn-sampler, where the tilt is the ratio of the maximum and the minimum (non-zero) probability masses of $\mathcal{I}$. We also provide a prototype implementation of Flash and test three state-of-the-art samplers on a set of benchmarks.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

1 INTRODUCTION

Sampling from complex combinatorial spaces is a fundamental problem in computer science with a wide variety of applications such as formal verification Chandra and Iyengar (1992), Yuan et al. (2004), Naveh et al. (2006), cryptography Mironov and Zhang (2006), Soos et al. (2009), Morawiecki and Srebrny (2013), Ashur et al. (2017) and various other fields. Often combinatorial spaces are specified as constraints expressed in logical theories where every point in the support corresponds to a solution of the given constraints. The task of designing efficient algorithms for sampling from such complex combinatorial spaces is very difficult in general. While techniques like Markov Chain Monte Carlo (MCMC) as well as those based on universal hashing allow design of sampling algorithms with theoretical guarantees, such algorithms often face scalability challenges. As a result, heuristic techniques are often employed to sample satisfying assignments. Although heuristic techniques often work well in practice and are often devoid of sound theoretical guarantees in general, such techniques may perform well for various scenarios of interest. Consequently, there is a dire need for the design of computationally efficient testers with sound theoretical guarantees that can verify whether a sampler is sampling according to a desired distribution for a given combinatorial space.

Due to the probabilistic nature of the sampling algorithms, designing testers for them is indeed 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 by obtaining samples from it. This requires an exponential number of samples Batu et al. (2001, 2013, 2004), Valiant and Valiant (2011b). Recently, Barbarik and Barbarik2 were proposed as provably correct testers for a set of constraints specified in Conjunctive Normal Form (CNF) Chakraborty and Meel (2019),

[1] Extended version of the paper and the accompanying tool, available open source, can be found at `https://github.com/uddaloksarkar/flash`.

Meel et al. (2020).

CNF is widely used to represent Boolean formulas owing to its expressiveness. In particular, every Boolean formula can be expressed as a CNF with a linear blow-up in size. Such expressiveness, however, comes at the cost of computational complexity: even determining the satisfiability of CNF formulas is NP-hard Cook (1971), Levin (1973). Accordingly, the symbolic reasoning community has explored restricted fragments of Boolean formulas that have tractable complexity. One such fragment is the class of Horn formulas, which play a vital role in logical systems Makowsky (1987), and forms the backbone of many logic programming languages, such as Prolog Lloyd (2012). Horn formulas are also used to model several real-life topological systems, such as power transmission lines, water and gas supply lines, and telecommunication networks Duenas-Osorio et al. (2017).

In this paper, we design efficient tester for testing the correctness of Horn-samplers. When the sampler is supposed to uniformly sample from the satisfying assignments of φ, we refer them as *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 refer them as *Weighted-Horn-sampler*.

It is worth emphasizing that the available testers Barbarik Chakraborty and Meel (2019) and Barbarik2 Meel et al. (2020) for CNF formulas do not provide 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 $\hat{\varphi}$ based on φ and draw samples by running the sampler over $\hat{\varphi}$. The correctness of these testers are shown under suitable assumptions, assuming 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 $\hat{\varphi}$ designed by Barbarik and Barbarik2 might not be a Horn formula, although the original 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: *is it possible to circumvent the above stumbling block and design testers for Horn-samplers?*

The primary contribution of this work is to answer the above question affirmatively. In particular, we design a novel Horn-sampler-tester Flash, that can test general Weighted-Horn-samplers. For input parameters $\varepsilon, \eta, \delta \in (0, 1)$, we prove that if the underlying distribution from which the Horn sampler-under-test draws samples is ε-*close* (in multiplicative ℓ_∞ distance [1]) to a fixed distribution $\mathcal{I}$ (given as input), then our tester Flash will ACCEPT with probability at least $(1 - \delta)$. On the other hand, if the underlying distribution induced by the Horn sampler-under-test is η-*far* (in ℓ_1 distance) from the given distribution $\mathcal{I}$, then our tester Flash will REJECT with probability at least $(1 - \delta)$, assuming the sampler satisfies certain conditions. Flash draws $\widetilde{\mathcal{O}}\left(\mathsf{tilt}^3/(\eta - \varepsilon)^4\right)$ samples from the underlying distribution [2]. Here the fixed distribution is generated from an arbitrary but fixed weight function wt given as input, and tilt denotes the ratio between the maximum and minimum non-zero probability masses among all the elements in $\mathcal{I}$. We would like to point out that there is a lower bound of $\Omega(1/(\eta - \varepsilon)^2)$ samples for Horn sampler-tester problem, which follows from the lower bound of estimating the bias of a coin.

We further provide a prototype implementation of Flash and experimental 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 the baseline approach.

Additionally, when the given distribution $\mathcal{I}$ is uniform over the set of all satisfying assignments of the input Horn formula, we can design a slightly simpler tester, called uFlash. Due to shortage of space and for simplicity of presentation, the details of uFlash along with its formal proof of correctness and experimental results are skipped in the main paper and presented in the supplementary material.

Organization of the Paper: In Section 2, we formally define the necessary terminology of our work and present a short description of related works. In Section 3, we present our Weighted-Horn-sampler-tester Flash. We show the evaluation results of the prototype implementation of Flash with respect to three state-of-the-art samplers UNIGEN, QUICKSAMPLER and STS in Section 4. The correctness proofs and detailed experimental results are presented in the extended version of the paper which is available at `https://github.com/uddaloksarkar/flash`.

2 PRELIMINARIES

In this work, we follow the same notations as in Chakraborty and Meel Chakraborty and Meel (2019), and Meel, Pote and Chakraborty Meel et al. (2020) in order to maintain the consistency. A literal is a Boolean variable or its negation, and a clause is a collection of disjoint literals, all connected either using the Boolean connective $\vee$, or the Boolean connective $\wedge$. Throughout the paper, we shall be using the terms 'literal' and 'variable' interchangeably when it is clear from the context. A Conjunctive Normal

[1] For two probability distributions p and q over Ω, the multiplicative ℓ_∞ and ℓ_1 distance between p and q are defined as $\max_{i\in\Omega} |p(i) - q(i)| \leq \varepsilon q(i)$, and $\sum_{i\in\Omega} |p(i) - q(i)|$, respectively for some parameter $\varepsilon \in (0, 1)$.

[2] $\widetilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic terms in $1/\eta$, $1/\varepsilon$, and $1/\delta$.

Form (CNF) is a Boolean formula with clause(s) where the variables inside any clause are connected using $\vee$ and different clauses are connected using $\wedge$. A Horn formula is a CNF formula where each clause contains *at most one* positive literal. For any Horn formula φ, the support of φ is denoted as $Supp(\varphi)$ and defined as the set of variables appearing in φ. Moreover, we assume that the variables in $Supp(\varphi)$ are linearly ordered, that is, for a formula φ with two variables, the support set can be written as $Supp(\varphi) = \{x_1, x_2\}$. An assignment $\sigma \in \{0,1\}^{|Supp(\varphi)|}$ is said to be a *witness* or satisfying assignment of φ if φ evaluates to 1 on the assignment σ. For any formula φ, let R_φ denote the set of all satisfying assignments of φ. For any set $S \subseteq Supp(\varphi)$, $\sigma_{\downarrow S}$ denotes the restriction or projection of the assignment σ to the variables of S. Similarly, $R_{\varphi\downarrow S} = \{\sigma_{\downarrow S} \mid \sigma \in R_\varphi\}$ denotes the set of satisfying assignments of φ projected on S. As an example, consider a formula $\varphi := (x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)$ defined over the set of variables $\{x_1, x_2, x_3, x_4, x_5\}$, whose only satisfying assignments are $\sigma_1 = 10100$ and $\sigma_2 = 10101$. Now consider $S \subseteq \{x_1, x_2, x_3, x_4, x_5\}$ such that $S = \{x_1, x_2, x_3\}$. If we project φ onto S, we have $R_{\varphi\downarrow S} = \{101\}$.

Definition 2.1 (Weight function)**.** Let S be a set of Boolean variables. A weight function $wt : \{0,1\}^{|S|} \to (0,1)$ assigns a weight to each assignment formed using S.

A weight function wt is not specific to any formula. Rather it assigns weights to every $|S|$-length Boolean string, irrespective of whether it satisfies a formula or not. We shall define the notion of samplers next. Thenceforth, we will move into defining sampler-tester. Vaguely, a Horn-sampler is a randomized algorithm which, given a Horn formula φ, returns a satisfying assignment of φ.

Definition 2.2 (Weighted-Horn-sampler)**.** A Weighted-Horn-sampler $\mathcal{G}(\varphi, S, wt, \kappa)$ is a randomized algorithm that takes as input a Horn formula φ, a set of variables $S \subseteq Supp(\varphi)$, 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, according to the weight function wt.

For brevity, we will often write $\mathcal{G}(\varphi, S, wt, \kappa)$ 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 $\mathcal{G}$ with φ as the input with $D_{\mathcal{G}(\varphi)}$.

Definition 2.3 (Ideal Weighted-Horn-sampler)**.** Consider a Horn formula φ, a set of variables $S \subseteq Supp(\varphi)$, 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 for every $\sigma \in R_{\varphi\downarrow S}$, $\mathbb{P}\left[\mathcal{I}_\mathcal{W}(\varphi, S, wt) = \sigma\right] = wt(\sigma)/\sum_{\sigma' \in R_{\varphi\downarrow S}} wt(\sigma')$.

Definition 2.4 (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 $\sigma \in R_\varphi$, the following holds [3]:

$$\begin{aligned}(1-\varepsilon)\mathbb{P}\left[\mathcal{I}_\mathcal{W}(\varphi) = \sigma\right] &\leq \mathbb{P}\left[\mathcal{G}(\varphi) = \sigma\right] \\ &\leq (1+\varepsilon)\mathbb{P}\left[\mathcal{I}_\mathcal{W}(\varphi) = \sigma\right].\end{aligned} \tag{1}$$

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

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

Several available samplers Gomes et al. (2006), Ermon et al. (2013), Chakraborty et al. (2013, 2015a) with theoretical guarantees test for ε-closeness (multiplicative approximation), and as a result, in our work, we also employ the same notion for the closeness metric. On the other hand, since there are several samplers available in the real world which do not have strong theoretical guarantees, we design our tester to be more accommodating, in the sense that our tester rejects based on the ℓ_1 distance, which is less stringent than multiplicative guarantees.

Definition 2.5 (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 $\eta > 9\varepsilon$, a confidence parameter δ, and a Horn formula φ, and:

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

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

Definition 2.6 (tilt)**.** For a Horn formula φ, and the associated arbitrary but fixed weight function wt, we define $\mathsf{tilt}(\varphi, wt)$ as

$$\mathsf{tilt}(\varphi, wt) := \max_{\sigma_1, \sigma_2 \in R_\varphi} \frac{wt(\sigma_1)}{wt(\sigma_2)},$$

that is, the maximum possible non-zero mass ratio between two satisfying assignments of a formula φ. We will refer to $\mathsf{tilt}(\varphi, wt)$ as tilt when it is clear from the context.

Definition 2.7 (Chain Formula)**.** The notion of chain formula was first introduced in Chakraborty et al. (2015b). Chain formulas provide a natural way to construct linear sized Boolean formulas with a precise number of satisfying assignments. Formally, a chain formula is defined as follows:

(i) Every literal (a Boolean variable or its negation) is a chain formula.

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

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

(iii) Let $m > 0$ be a natural number and $k < 2^m$ be a positive odd number. Let $c_1 c_2 \dots c_m$ be the m-bit representation of k, where c_m is the Least Significant Bit (LSB) in the representation of m. For every $j \in \{1, \dots, m-1\}$, 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: $\psi_{k,m}(a_1, a_2, .., a_m) = a_1 C_1 (a_2 C_2 (..(a_{m-1} C_{m-1} a_m)..)$, where $a_1, a_2, \dots, a_m$ are variables.

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

Given two natural numbers m and k with $k < 2^m$, the authors in Chakraborty et al. (2015b) constructed a chain CNF formula $\psi_{k,m}$ such that the size of $\psi_{k,m}$ is linear in m and $\psi_{k,m}$ has exactly k satisfying assignments. But the chain formulas constructed in Chakraborty et al. (2015b) are not necessarily Horn - a property that we study in this work. We extend their result to obtain *Horn-chain-formula* which is crucially used in the design of Flash.

Lemma 2.8 (Horn-chain-formula)**.** *Given a natural number $m > 0$ and another integer $k < 2^m$, there exists a Horn-chain-formula $\psi'_{k,m}$ such that the size of $\psi'_{k,m}$ is linear in m and $\psi'_{k,m}$ has exactly k satisfying assignments.*

The exact construction of the Horn-chain-formula and the proof of the above lemma is in the extended version of the paper.

Related works: As noted earlier, the problem of testing correctness of samplers boils down to the problem of testing equivalence between a known distribution $\mathcal{I}$ and an unknown distribution $\mathcal{D}$ (namely the underlying distribution according to which the sampler-under-test samples). This problem of testing equivalence between a known and an unknown distribution has been well studied in the literature of statistics and property testing for several decades. Traditionally in the field of distribution testing, the distribution $\mathcal{D}$ is accessed via obtaining independent and identically distributed samples and the goal is to distinguish whether $\mathcal{D}$ is ε-close or η-far from $\mathcal{I}$ by taking as few samples as possible. Generally, ℓ_1 distance is used as the distance measure. However, it is well known that this requires $\Theta(N/\log N)$ samples, where N is the support size of $\mathcal{D}$ Valiant and Valiant (2011a), Batu et al. (2001). In case of samplers, since the number of satisfying assignments can be exponential in the number of variables of the formula, traditional "*black-box*" approach of sampling is infeasible.

In order to bypass this problem, a new model termed as *conditional sampling* model was introduced by Chakraborty et. al Chakraborty et al. (2016) and Cannone et. al Canonne et al. (2015). In this setting, given a set $X \subseteq [N]$, the sampler obtains a sample $i \in X$ from the distribution, conditioned on the set X. Surprisingly, several interesting problems which have high sample complexity in the standard black-box sampling model, can be solved very easily in this model, often using poly-logarithmic or even constant number of conditional samples Canonne et al. (2014), Falahatgar et al. (2015). Therefore, the challenge of designing practically efficient testers boils down to efficiently obtaining conditional samples from the distribution induced by the sampler.

Chakraborty and Meel Chakraborty and Meel (2019) formulated a sampling technique for obtaining conditional samples from CNF-samplers. Let the sampler-under-test be $\mathcal{G}$ and φ is a CNF-formula, and $X \subset R_\varphi$ be a set of size 2. To draw a conditional sample (conditioned over X) from the distribution $\mathcal{D}_{\mathcal{G}(\varphi)}$, they run the sampler $\mathcal{G}$ over a different and carefully constructed CNF-formula $\hat{\varphi}$. One may call this approach of obtaining conditional samples as "*grey-box*" sampling. The correctness of Chakraborty and Meel (2019) crucially depends on the construction of the new CNF-formula $\hat{\varphi}$.

Using this grey-box sampling technique, Chakraborty and Meel Chakraborty and Meel (2019) designed the first efficient tester Barbarik for testing uniformity of CNF-samplers that takes $\widetilde{\mathcal{O}}(\frac{1}{(\eta-2\varepsilon)^4})$ many samples, where ε and η are the closeness and farness parameters respectively. They not only built the prototype implementation of Barbarik and provided experimental analysis, but also provided strong theoretical guarantee on the correctness of their algorithm. They proved that if the sampler-under-test is ε-close to uniform (in the multiplicative ℓ_∞ distance), then the tester would "certainly" [4] accept the sampler. On the other hand, if the sampler-under-test is η-far from uniform (in the additive ℓ_1 distance) and the sampler satisfies a "*subquery consistency*" assumption, then the tester would reject the sampler [5]. It is important to note that the authors of Chakraborty and Meel (2019) used the notions of multiplicative ℓ_∞ and ℓ_1 distance measures for closeness and farness respectively. They argued that the use of different distance measures is more suitable for practice. Later Meel, Pote and Chakraborty Meel et al. (2020) used similar grey-box techniques to design the tester Barbarik2 for testing weighted samplers for CNF formulas.

It is not immediately clear how the grey-box sampling techniques of Chakraborty and Meel (2019) and Meel et al. (2020) can be employed to obtain conditional samples when the samplers-under-test can only correctly handle Horn formulas. One of the crucial technical challenges here is to design an efficient and practical procedure to obtain conditional samples from the distribution induced by the Horn sampler-under-test.

[4] With probability at least $1 - \delta$ for some parameter $\delta \in (0, 1)$.

[5] Please see Section 3 for more discussion on the subquery consistency assumption.

Algorithm 1: Flash $(\mathcal{G}, \mathcal{I}_{\mathcal{W}}, S, \varepsilon, \eta, \delta, \varphi)$

$t \leftarrow \frac{10}{\eta(\eta-9\varepsilon)} \log_e\left(\frac{1}{\delta}\right)$;
$z \leftarrow \log_e\left(\frac{2t}{\delta}\right)$;
$lo \leftarrow \frac{1+\varepsilon}{1-\varepsilon}$;
$hi \leftarrow 1 + \frac{\eta+9\varepsilon}{4}$;
$\Gamma_1 \leftarrow \mathcal{G}(\varphi, S, t)$;
$\Gamma_2 \leftarrow \mathcal{I}_{\mathcal{W}}(\varphi, S, t)$;
for $i \leftarrow 1$ *to* t **do**
  $\sigma_1 \leftarrow \Gamma_1[i]$; $\sigma_2 \leftarrow \Gamma_2[i]$;
  if $\sigma_1 == \sigma_2$ **then**
    continue
  $\alpha \leftarrow \frac{wt(\sigma_1)}{wt(\sigma_2)}$;
  $L \leftarrow \frac{\alpha \cdot lo}{1+\alpha \cdot lo}$;
  $H \leftarrow \frac{\alpha \cdot hi}{1+\alpha \cdot hi}$;
  $T = \frac{(H+L)}{2}$;
  $N \leftarrow \frac{8z \cdot H}{(H-L)^2}$;
  $X \leftarrow \left(\frac{(1-\varepsilon)(wt(\sigma_1)+wt(\sigma_2))}{(1-\varepsilon)(wt(\sigma_1)+wt(\sigma_2))+(1+\varepsilon)wt(\vec{0})}\right)$;
  $M \leftarrow \left(\frac{\sqrt{z}+\sqrt{z+4NX}}{2X}\right)^2$;
  $\hat{\varphi} \leftarrow \mathsf{HornKernel}(\varphi, \sigma_1, \sigma_2, M)$;
  $\Gamma_3 \leftarrow \mathcal{G}(\hat{\varphi}, S, M)$;
  $\widehat{\Gamma}_3 \leftarrow \mathsf{RemoveZeros}(\Gamma_3, \sigma_1, \sigma_2)$;
  if $\left|\widehat{\Gamma}_3\right| < N$ **then**
    return REJECT
  $Bias \leftarrow \mathsf{Bias}(\sigma_1, \widehat{\Gamma}_3, S)$;
  if $Bias > T$ **then**
    return REJECT
return ACCEPT

3 TESTER FOR HORN SAMPLERS

As stated in Section 1, any black-box sampling technique for testing Horn samplers requires exponential number of samples. To bypass this bottleneck, Flash employs a grey-box sampling technique by drawing samples from a conditional distribution.

The primary technical challenge of Flash is to draw samples from a conditional distribution obtained by conditioning over the distribution $\mathcal{D}_{\mathcal{G}(\varphi)}$ on two distinct satisfying assignments of φ, namely, σ_1 and σ_2. The plan is to construct a new Horn formula $\hat{\varphi}$ from φ such that the only satisfying assignments of $\hat{\varphi}$ are σ_1 and σ_2. One easy way is to come up with a CNF formula $\hat{\varphi}$ by conjuncting $(\sigma_1 \vee \sigma_2)$ to φ and obtaining the new formula $\hat{\varphi} := \varphi \wedge (\sigma_1 \vee \sigma_2)$.

However, this brings us to the second challenge. The newly constructed formula $\hat{\varphi}$ may not be a Horn formula $\hat{\varphi}$, and thus cannot be handled by the Horn sampler $\mathcal{G}$. Flash makes use of the subroutine HornKernel to avoid this inconsistency. Interestingly, the newly constructed Horn formula $\hat{\varphi}$ may contain another satisfying assignment aside σ_1

Algorithm 2: HornKernel $(\varphi, \sigma_1, \sigma_2, \tau)$

$\mathcal{L} \leftarrow \mathsf{Encode}(\sigma_1, \sigma_2)$;
$\hat{\varphi} \leftarrow \varphi \wedge \mathcal{L}$;
$Lits_1 \leftarrow (\sigma_1 \backslash \sigma_2)$;
$Lits_2 \leftarrow (\sigma_2 \backslash \sigma_1)$;
$n \leftarrow \min(|Lits_1 \cup Lits_2|, 4)$;
$k \leftarrow \lceil \tau^{1/n} \rceil$;
$m \leftarrow \lceil \log(k) \rceil$;
$V \leftarrow \mathsf{NewVars}(\varphi, m, n)$;
$ix \leftarrow 0$;
for $i \in [n]$ **do**
  $l \sim Lits_1 \cup Lits_2$;
  $\hat{\varphi} \leftarrow \hat{\varphi} \wedge (l \rightarrow \psi'_{k,m}(V[ix : ix + m]))$;
  $\hat{\varphi} \leftarrow \hat{\varphi} \wedge (\neg l \rightarrow \psi'_{k,m}(V[ix : ix + m]))$;
  $ix \leftarrow ix + m$;
return $\hat{\varphi}$;

Algorithm 3: Encode (σ_1, σ_2)

$\Sigma \leftarrow [\sigma_1, \sigma_2]$;
$\mathcal{L} \leftarrow True$;
$TrueLits \leftarrow \mathsf{cmmTrueLits}(\sigma_1, \sigma_2)$;
$FalseLits \leftarrow \mathsf{cmmFalseLits}(\sigma_1, \sigma_2)$;
$diffLits \leftarrow \mathsf{unCmmLits}(\sigma_1, \sigma_2)$;
$tLit \sim TrueLits$;
$fLit \sim FalseLits$;
for *each* $i \in TrueLits \setminus \{tLit\}$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} \wedge (x_i \iff x_{tLit})$;
$\mathcal{L} \leftarrow \mathcal{L} \wedge x_{tLit}$;
for *each* $i \in FalseLits \setminus \{fLit\}$ **do**
  $\mathcal{L} \leftarrow \mathcal{L} \wedge (x_i \iff x_{fLit})$;
$\mathcal{L} \leftarrow \mathcal{L} \wedge \neg x_{fLit}$;
$diff_1 \leftarrow \mathsf{NULL}$;
$diff_2 \leftarrow \mathsf{NULL}$;
$(diff_1, diff_2) \leftarrow \mathsf{findSplitVars}(\sigma_1, \sigma_2)$;
for *each* $i \in diffLits$ **do**
  if $val(x_i, \sigma_1) == 1$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} \wedge (x_i \iff x_{diff_1})$;
  else
    $\mathcal{L} \leftarrow \mathcal{L} \wedge (x_i \iff x_{diff_2})$;
if $diff_1 \neq \mathsf{NULL}$ & $diff_2 \neq \mathsf{NULL}$ **then**
  $\mathcal{L} \leftarrow \mathcal{L} \wedge (x_{diff_1} \implies \neg x_{diff_2})$;
return $\mathcal{L}$;

Algorithm 4: NewVars (φ, m, n)

$\mathcal{R} \leftarrow$ set of all variables;
$S \leftarrow Supp(\varphi)$;
$V \leftarrow \emptyset$;
for $i \in [n]$ *and* $j \in [m]$ **do**
  $l \sim \mathcal{R} \setminus S$;
  $V \leftarrow V \cup l$;
return V;

Algorithm 5: RemoveZeros $(\Gamma, \sigma_1, \sigma_2)$

$\widehat{\Gamma} \leftarrow \emptyset$;
for $\gamma \in \Gamma$ **do**
  if $\gamma \in \{\sigma_1, \sigma_2\}$ **then**
    $\widehat{\Gamma} \leftarrow \widehat{\Gamma} \cup \gamma$;
return $\widehat{\Gamma}$

Algorithm 6: Bias (σ, L, S)

$count \leftarrow 0$;
for $\sigma' \in L$ **do**
  if $\sigma'_{\downarrow S} == \sigma$ **then**
    $count \leftarrow count + 1$;
return $\frac{count}{|L|}$;

and σ_2, which we will denote as $\widetilde{0}$ [6]. For correctness, we try to estimate the weight ratio of σ_1 and σ_2 with respect to the weight function wt. For estimation purposes, Flash draws enough samples of σ_1 and σ_2 using $\mathcal{G}$ from $\hat{\varphi}$. Meanwhile, a significant amount of $\widetilde{0}$ may appear in the sampled set which is taken care of properly by Flash.

In Section 3.1, we give a high level overview of various subroutines, while in Section 3.2, we discuss the correctness of Flash.

3.1 High Level Description of Our Subroutines

While the basic framework of Flash is similar to that of Barbarik2, there are significant differences in the subroutines used. In the following, we start by briefly describing the subroutines HornKernel, Encode, Bias, and RemoveZeros.

HornKernel: The primary goal of HornKernel (Algorithm 2) is to employ the conditioning step. It takes as input a Horn formula φ, two distinct satisfying assignments $\sigma_1, \sigma_2 \in R_\varphi$, and an integer τ, and returns a Horn formula $\hat{\varphi}$ such that φ and $\hat{\varphi}$ have "similar" structures, and the following conditions hold:

(1) $|R_{\hat{\varphi}}| = 2\tau$.

(2) $Supp(\varphi) \subseteq Supp(\hat{\varphi})$.

(3) Let $\widetilde{0}$ denote the assignment whose only True literals are the common True literals of σ_1 and σ_2 and $S \subseteq Supp(\varphi)$ be a set of variables.

(a) If $\widetilde{0} \notin R_\varphi$, then $R_{\hat{\varphi}\downarrow S}$ has only two distinct elements σ_1 and σ_2 and

$$|\{\sigma \in R_{\hat{\varphi}} \mid \sigma_{\downarrow S} = \sigma_1\}| = |\{\sigma \in R_{\hat{\varphi}} \mid \sigma_{\downarrow S} = \sigma_2\}|$$

(b) If $\widetilde{0} \in R_\varphi$ then $R_{\hat{\varphi}\downarrow S}$ has only three distinct elements σ_1, σ_2 and $\widetilde{0}$ and

$$|\{\sigma \in R_{\hat{\varphi}} \mid \sigma_{\downarrow S} = \sigma_1\}| = |\{\sigma \in R_{\hat{\varphi}} \mid \sigma_{\downarrow S} = \sigma_2\}| = \left|\{\sigma \in R_{\hat{\varphi}} \mid \sigma_{\downarrow S} = \widetilde{0}\}\right|.$$

To achieve these properties, the subroutine HornKernel, in Line 1, constructs a Horn formula $\mathcal{L}$ from σ_1, σ_2 using the subroutine Encode (discussed below) such that σ_1, σ_2 and $\widetilde{0}$ are the only satisfying assignments of $\mathcal{L}$. 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 $\widetilde{0}$. After HornKernel constructs a Horn formula $\mathcal{L}$ from σ_1, σ_2 using Encode (in Line 1), it constructs a new Horn formula $\hat{\varphi}$ by conjuncting the original Horn formula φ with $\mathcal{L}$ (which in effect will generate the conditioned distribution) [7]. Next HornKernel tries to blow-up the sample space of $\hat{\varphi}$ from 3 to $\mathcal{O}(\tau)$ so that an adversarial sampler could not fool Flash. To do this, HornKernel makes use of Horn-chain-formula. In Line 3, 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 that are true in exactly one of σ_1 and σ_2. In Line 5, it sets the variable n to $min\{|Lits_1 \cup Lits_2|, 4\}$, which indicates the number of Horn-chain-formulas it will generate further to achieve the blow-up. Then in Line 7, it calculates the values of k and m which are the parameters needed to produce Horn-chain formula $\psi'_{k,m}$ (from Lemma 2.8). As noted earlier, the construction of Horn-chain-formulas in Lemma 2.8, is another important technical contribution of this paper. $\psi'_{k,m}$ has exactly k satisfying assignments. Then, in Line 8, it generates a list V of $n \times m$ new variables that are not present in $Supp(\varphi)$ using a subroutine NewVars (Algorithm 4) which is required to construct the Horn-chain-formula. Finally, in the loop of Line 10, HornKernel constructs a new Horn formula $\hat{\varphi}$ by adding n Horn-clauses of the form $(l \rightarrow \psi'_{k,m})$ and $(\neg l \rightarrow \psi'_{k,m})$ over the set V of newly generated variables, where l is a literal chosen from $Lits_1 \cup Lits_2$, and $\psi'_{k,m}$ is a Horn-chain-formula defined on variables in V, with $R_{\hat{\varphi}}$ of size $\mathcal{O}(\tau)$.

Encode: The subroutine Encode (Algorithm 3) is another technical contribution of this work. We first define some terminologies used to describe Encode. As discussed in Section 2, the set of literals $Supp(\varphi)$ of φ is a linearly ordered set, where the k-th element literal is denoted as x_k. Given two witnesses σ_1 and σ_2, $TrueLits$ (resp. $FalseLits$) denotes the index-set of literals which are True (resp. False) in both σ_1 and σ_2. $tLit$ and $fLit$ are indices chosen from $TrueLits$ and $FalseLits$ respectively.

[6] The structure of $\widetilde{0}$ depends upon σ_1 and σ_2.

[7] $\mathcal{L}$ itself cannot represent the conditioning, as the structure of φ and $\mathcal{L}$ can be completely different. Hence the distributions on R_φ and $R_\mathcal{L}$ may be completely different.

$diffLits$ denotes the index-set of literals which have different assignments in σ_1 and σ_2. Finally $diff1$ and $diff2$ are chosen from $diffLits$, such that, $x_{diff1} = 0$ and $x_{diff2} = 1$ in σ_1, but $x_{diff1} = 1$ and $x_{diff2} = 0$ in σ_2. For example, given $\sigma_1 := 1100$ and $\sigma_2 := 1010$, we have $TrueLits = \{0\}$, $FalseLits = \{3\}$, $tLit = 0$, $fLit = 3$, $diff1 = 2$, $diff2 = 1$.

The idea of Encode is to partition the set of variables appearing in σ_1 and σ_2 into four equivalence classes with respect to the relation ' $\iff$ ': (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 index-set of the common True and False literals of σ_1 and σ_2 by means of cmmTrueLits and cmmFalseLits in Line 3 and Line 4, respectively. It also finds the index-set of literals $diffLits$ that have different values using unCmmLits in Line 5. 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 $\mathcal{L}$ by adding 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 $\mathcal{L}$ 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.

To take care of the last two equivalence classes, it first finds two variables x_{diff1} and x_{diff2} using findSplitVars such that, $x_{diff1} = 0$ and $x_{diff2} = 1$ in σ_1, but $x_{diff1} = 1$ and $x_{diff2} = 0$ in σ_2 (since $\sigma_1 \neq \sigma_2$, the existence of at least one of the variables x_{diff1} and x_{diff2} is guaranteed). Using 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 last two equivalence classes, Encode adds the clause $(x_{diff1} \implies \neg x_{diff2})$ in Line 23. Although Encode might add the clause $(x_{diff1} \iff \neg x_{diff2})$ in Line 23 (which would produce $\mathcal{L} \equiv \sigma_1 \vee \sigma_2$), it turns out that $(\neg x_{diff2} \implies x_{diff1})$ is not a Horn formula [8]. Such incapability causes 3 witnesses of $\mathcal{L}$: σ_1, σ_2, and $\widetilde{0}$, instead of only σ_1 and σ_2.

Flash: Flash (Algorithm 1) first draws t samples from the Horn sampler $\mathcal{G}$ to be tested, as well as from the ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$ and stores them in Γ_1 and Γ_2 in Line 5. Thus, Γ_1 is a set of samples obtained from the distribution $D_{\mathcal{G}(\varphi)}$, while Γ_2 is a set of samples drawn according to the fixed distribution with respect to wt over R_φ, the witness space of φ. Then in the for loop of Line 7, it first takes a sample σ_1 from Γ_1, and another sample σ_2 from Γ_2, and calculates the parameters T, M, N from the weights of σ_1, σ_2 and $\widetilde{0}$. Flash then calls the subroutine HornKernel with σ_1, σ_2 and φ in Line 18, and HornKernel returns a Horn formula $\hat{\varphi}$ that employs the conditioning. Next Flash obtains M satisfying assignments of $\hat{\varphi}$ in Line 19, and calls the subroutine RemoveZeros (Algorithm 5) in Line 20 to check if there are at least N witnesses of σ_1 and σ_2 out of the M witnesses obtained in Line 19. If the number of occurrences 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 23 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 7, 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 fixed distribution. It may be noted that unlike Barbarik2 [Meel et al. (2020)], Flash has to carefully handle the fact that the formula $\hat{\varphi}$ on which the sampler $\mathcal{G}$ is run must be a Horn formula. At the same time, we need to take care of the fact that the formula $\hat{\varphi}$ returned by the subroutine HornKernel may have three satisfying assignments (after projecting onto the support of φ), instead of exactly two - namely σ_1 and σ_2, which was crucial for the correctness of Barbarik2.

3.2 Correctness of Flash

For an arbitrary but fixed weight function wt and parameters $\varepsilon \in (0, 1/3), \eta \in (0, 2), \delta > 0$ with $\eta > 9\varepsilon$, our Horn-sampler-tester Flash has the following three guarantees:

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

(ii) if Flash outputs REJECT for $\mathcal{G}$, then it provides a certificate of rejection.

(iii) if $\mathcal{G}$ is η-far from $\mathcal{I}_{\mathcal{W}}$, and $\mathcal{G}$ is subquery consistent as mentioned in the introduction, Flash outputs REJECT with probability at least $(1 - \delta)$.

Now we formally describe the notion of subquery consistency:

Subquery Consistency of Sampler $\mathcal{G}$: Consider any Horn formula φ. For every $S \subseteq Supp(\varphi)$, $\sigma_1, \sigma_2 \in R_{\varphi\downarrow S}$, let $\hat{\varphi}$ be the Horn formula obtained from the subroutine ENCODE. A Horn sampler $\mathcal{G}$ is said to be *subquery consistent*, if the output of $\mathcal{G}(\hat{\varphi}, wt, S, \kappa)$ is κ 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. The justification behind this assumption follows from the fact that φ and $\hat{\varphi}$ have similar structures. Hence the distribution induced by $\mathcal{G}$ for $\hat{\varphi}$, that is, $\mathcal{D}_{\mathcal{G}(\hat{\varphi})}$ and $\mathcal{D}_{\mathcal{G}(\varphi)}|X$ has to be similar.

[8] $(\neg a \implies b) \equiv (b \vee a)$ contains 2 positive literals.

Before going into our main theorem, we formally state here the lemma that supports our Encode subroutine.

Lemma 3.1. *(i) The formula Γ generated by* Encode *is a Horn formula.*

(ii) There are only three satisfying assignments of Γ, that is, $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$.

The formal statement of correctness of Flash is presented below, while its proof is in the extended version of the paper.

Theorem 3.2. *Given a Weighted-Horn-sampler $\mathcal{G}$, an ideal Weighted-Horn-sampler $\mathcal{I}_\mathcal{W}$ with respect to an arbitrary but fixed weight function wt, a tolerance parameter $\varepsilon \in (0, 1/3)$, an intolerance parameter $\eta \in (0, 2]$, with $\eta > 9\varepsilon$, and a confidence parameter $\delta > 0$, our Weighted-Horn-sampler-tester* Flash *takes $\widetilde{\mathcal{O}}(\frac{\mathsf{tilt}(wt,\varphi)^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2})$ samples, and:*

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

(ii) If $\mathcal{G}$ is η-far from $\mathcal{I}_\mathcal{W}$, and $\mathcal{G}$ is subquery consistent, Flash *outputs REJECT with probability at least $(1-\delta)$.*

4 EVALUATION RESULTS

To evaluate the practical effectiveness of Flash, we implemented the prototype of Flash in Python 3.8.3. 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 4 GB 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 code are in the extended version.

The primary objective of our empirical evaluation was to answer the following questions:

RQ1 Can Flash test whether off-the-shelf samplers are ε-close or η-far from ideal samplers?

RQ2 What kind of improvements are possible over the baseline?

Samplers Tested: We follow the similar setup as in Barbarik and Barbarik2. We employ the following three state-of-the-art samplers: UNIGEN3 Soos et al. (2020), QUICKSAMPLER Dutra et al. (2018), STS Ermon et al. (2012) [9] and augment these samplers with an inverse sampling module [10]. We shall term the newly generated samplers as WUNIGEN, WQUICKSAMPLER, WSTS respectively in Table 2. Furthermore, while implementing Flash, our algorithm requires the access of a known ideal Weighted-Horn-sampler $\mathcal{I}_\mathcal{W}$ beforehand. We use SPUR Achlioptas et al. (2018) as the corresponding ideal Uniform-Horn-sampler, and augment it by inverse sampling to achieve ideal Weighted-Horn-sampler needed for Flash.

Test Parameters: For our experiments with Flash, 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 Flash outputs ACCEPT when the sampler $\mathcal{G}$ to be tested is ε-close to the ideal Weighted-Horn-sampler with probability at least $(1-\delta)$. Similarly, with probability at least $(1-\delta)$, Flash outputs REJECT when $\mathcal{G}$ is η-far from ideal Weighted-Horn-sampler.

Benchmarks: Our benchmark suite consists of formulas arising from the reliability computation of power transmission networks in US cities Duenas-Osorio et al. (2017). For the evaluation of Flash, we only consider log-linear distributions which play a crucial role in several machine learning algorithms. A formal discussion on log-linear distribution is presented in the extended version. In particular, the weight functions for log-linear distributions can be specified using weights on literals. For every benchmark instance of Horn formulas, we designed two sets of weight functions for evaluation purposes and thus we procured two sets of benchmarks (Benchmark-I and Benchmark-II) as depicted in Table 1. Column 3 (Column 4) in Table 1 shows the tilt with respect to the weight function wt in Benchmark-I (Benchmark-II). While designing such weight functions, we sample a set of literals from the support set and assign random nontrivial weights [11]. Then we assign weight 0.5 to the rest of the literals. For Benchmark-I, we pick each literal with probability $1/3$, while for Benchmark-II, 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.

Benchmark	Model Count	tilt (Benchmark - I)	tilt (Benchmark - II)
Net6_count_91	2.19×10^{32}	20.40	12.34
Net8_count_96	3.2×10^{36}	26.23	5.80
Net12_count_106	6.34×10^{43}	20.40	5.80
Net22_count_116	9.49×10^{50}	26.23	7.46
Net27_count_118	8.05×10^{53}	43.36	7.46
Net29_count_164	4.51×10^{63}	92.17	7.46
Net39_count_240	2.46×10^{91}	14043.96	9.60
Net43_count_243	8.41×10^{100}	1137.74	4.51
Net46_count_322	3.22×10^{129}	23215.53	5.80
Net52_count_362	2.64×10^{147}	286565.21	2.73
Net53_count_339	4.05×10^{143}	38376.70	7.46

Table 1: Benchmark Details: Model Count and tilt

Description of the Tables: Table 2 depicts our experiments with Flash. In the 2nd, 3rd and 4th columns of Table 2, we present the experimental results of Flash on

[9] We use the default parameters for QUICKSAMPLER, STS and UNIGEN, which were employed in the previous studies Chakraborty and Meel (2019), Meel et al. (2020) to maintain the consistency of the experiments.

[10] Inverse sampling converts (φ, wt) to a formula $\hat{\varphi}$ preserving the satisfying assignments' distribution.

[11] 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.

wUniGen, wQuickSampler and wSTS, respectively. In each of these cells, A and R indicate whether the output of Flash were ACCEPT or REJECT, respectively, and the 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 Flash 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 the input formula. But TLE indicates a combined failure of both the sampler-under-test and our verification algorithm.

		wUniGen		wQuickSampler		wSTS	
	Benchmark	o/p	#Samples	o/p	#Samples	o/p	#Samples
Benchmark-I	N6_c_91_w1	TLE	-	R	106910	R	15626
	N8_c_96_w1	TLE	-	R	22716	R	39944
	N12_c_106_w1	TLE	-	R	27428	R	41334
	N22_c_116_w1	DNS	-	R	98629	R	9217
	N27_c_118_w1	DNS	-	R	49654	R	25296
	N29_c_164_w1	DNS	-	R	123202	R	12322
	N39_c_240_w1	DNS	-	R	7745	R	7922
	N43_c_243_w1	DNS	-	R	209062	R	22351
	N46_c_322_w1	DNS	-	R	23105	R	7922
	N52_c_362_w1	DNS	-	R	6085	R	8650
	N53_c_339_w1	DNS	-	R	38417	R	23105
Benchmark-II	N6_c_91_w2	A	274175	R	17667	R	26995
	N8_c_96_w2	A	397169	A	388885	R	16385
	N12_c_106_w2	A	197713	R	6085	R	5930
	N22_c_116_w2	A	302546	R	22947	R	24561
	N27_c_118_w2	TLE	-	R	10405	R	26245
	N29_c_164_w2	A	238673	R	7226	R	17706
	N39_c_240_w2	A	282138	R	13690	R	14885
	N43_c_243_w2	TLE	-	R	238260	R	9217
	N46_c_322_w2	A	437529	R	135368	R	30819
	N52_c_362_w2	TLE	-	R	210925	R	23127
	N53_c_339_w2	A	191806	R	8650	R	9605

[a] Benchmark NetX_count_Y is abbreviated as NX_c_Y

Table 2: Evaluation results of Flash

Detailed Results

RQ1: From our experiments, we find that among the 22 instances, Flash outputs REJECT in all the instances of wSTS. When run with wQuickSampler, Flash outputs REJECT in 21 instances, and outputs ACCEPT in 1 instance. When Flash is run with wUniGen, Flash outputs ACCEPT for 8 instances, while there are 6 cases of TLE and 8 instances of DNS. For these instances, Flash 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 Batu et al. (2013), 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 wUniGen, wSTS and wQuickSampler. It is worth highlighting that Flash terminates within 24 hours for all the instances for most of the samplers - hence the massive (over 10^7) speed up in the runtime compared to the baseline for all the benchmark instances.

Conclusion: We designed and implemented the first Horn-sampler-tester Flash with sound theoretical guarantees. It also works well in practice, as described by the evaluation results with respect to wUniGen, wQuickSampler, and wSTS.

Limitations of our work: The primary limitation of our approach is the sample complexity of Flash, which depends on the cubic power of tilt. Thus, when the value of tilt is large, the sample complexity of Flash becomes quite high. This phenomenon is also supported from the evaluation results of Flash presented in Table 2, where for some instances of the benchmark data, we have exceeded the time limit (TLE), particularly while dealing with wUniGen.

Acknowledgement

This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0004], Ministry of Education Singapore Tier 2 grant MOE-T2EP20121-0011, and Ministry of Education Singapore Tier 1 Grant [R-252-000-B59-114]. The computational works of this article were performed on the resources of the National Supercomputing Centre, Singapore https://www.nscc.sg. We also thank Yash Pote for helpful discussions regarding the work and Arijit Shaw for technical inputs during the implementation.

References

Achlioptas, D., Hammoudeh, Z. S., and Theodoropoulos, P. (2018). Fast sampling of perfectly uniform satisfying assignments. In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, pages 135–147.

Ashur, T., De Witte, G., and Liu, Y. (2017). An automated tool for rotational-xor cryptanalysis of arx-based primitives. In *SITB*.

Batu, T., Fischer, E., Fortnow, L., Kumar, R., Rubinfeld, R., and White, P. (2001). Testing random variables for independence and identity. In *Proc. of FOCS*, pages 442–451.

Batu, T., Fortnow, L., Rubinfeld, R., Smith, W. D., and White, P. (2000). Testing that distributions are close. In *Proc. of FOCS*, pages 259–269.

Batu, T., Fortnow, L., Rubinfeld, R., Smith, W. D., and White, P. (2013). Testing closeness of discrete distributions. *J. ACM*, 60(1):1–25.

Batu, T., Kumar, R., and Rubinfeld, R. (2004). Sublinear algorithms for testing monotone and unimodal distributions. In *Proc. of STOC*, pages 381–390.

Canonne, C., Ron, D., and Servedio, R. A. (2014). Testing equivalence between distributions using conditional samples. In *Proc. of SODA*, pages 1174–1192.

Canonne, C. L., Ron, D., and Servedio, R. A. (2015). Testing probability distributions using conditional samples. *SIAM J. Comput.*, 44:540–616.

Chakraborty, S., Fischer, E., Goldhirsh, Y., and Matsliah, A. (2016). On the power of conditional samples in distribution testing. *SIAM J. Comput.*, 45:1261–1296.

Chakraborty, S., Fremont, D. J., Meel, K. S., Seshia, S. A., and Vardi, M. Y. (2015a). On parallel scalable uniform SAT witness generation. In *TACAS*, volume 9035, pages 304–319.

Chakraborty, S., Fried, D., Meel, K. S., and Vardi, M. Y. (2015b). From weighted to unweighted model counting. In *Proc. of IJCAI*, pages 689–695.

Chakraborty, S. and Meel, K. S. (2019). On testing of uniform samplers. In *Proc. of AAAI*, pages 7777–7784.

Chakraborty, S., Meel, K. S., and Vardi, M. Y. (2013). A scalable and nearly uniform generator of SAT witnesses. In *In Proc. of CAV*, volume 8044, pages 608–623.

Chandra, A. K. and Iyengar, V. S. (1992). Constraint solving for test case generation: a technique for high-level design verification. In *Proc. of ICCD*, pages 245–246.

Chavira, M. and Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artif. Intell.*, 172:772–799.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proc. of STOC*, pages 151–158.

Dubhashi, D. P. and Panconesi, A. (2009). *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.

Duenas-Osorio, L., Meel, K., Paredes, R., and Vardi, M. (2017). Counting-based reliability estimation for power-transmission grids. In *Proc. of AAAI*, volume 31.

Dutra, R., Laeufer, K., Bachrach, J., and Sen, K. (2018). Efficient sampling of sat solutions for testing. In *Proc. of ICSE*, pages 549–559.

Ermon, S., Gomes, C. P., Sabharwal, A., and Selman, B. (2013). Embed and project: Discrete sampling with universal hashing. In *Proc. of NeurIPS*, pages 2085–2093.

Ermon, S., Gomes, C. P., and Selman, B. (2012). Uniform solution sampling using a constraint solver as an oracle. In *Proc. of UAI*, pages 255–264.

Falahatgar, M., Jafarpour, A., Orlitsky, A., Pichapati, V., and Suresh, A. T. (2015). Faster algorithms for testing under conditional sampling. In *COLT*, volume 40, pages 607–636.

Goldreich, O. and Ron, D. (2011). On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography*, volume 6650, pages 68–75. Springer.

Gomes, C. P., Sabharwal, A., and Selman, B. (2006). Near-uniform sampling of combinatorial spaces using XOR constraints. In *Proc. of NeurIPS*, pages 481–488.

Levin, L. A. (1973). Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116.

Lloyd, J. W. (2012). *Foundations of logic programming*. Springer Science & Business Media.

Makowsky, J. A. (1987). Why horn formulas matter in computer science: Initial structures and generic examples. *J. Comput. Syst. Sci.*, 34:266–292.

Meel, K. S., Pote, Y. P., and Chakraborty, S. (2020). On testing of samplers. In *Proc. of NeurIPS*, volume 33, pages 5753–5763.

Mironov, I. and Zhang, L. (2006). Applications of sat solvers to cryptanalysis of hash functions. In *SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115.

Morawiecki, P. and Srebrny, M. (2013). A sat-based preimage analysis of reduced keccak hash functions. *Inf. Process. Lett.*, 113(10-11):392–397.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press.

Naveh, Y., Rimon, M., Jaeger, I., Katz, Y., Vinov, M., s Marcu, E., and Shurek, G. (2006). Constraint-based random stimuli generation for hardware verification. *Proc. of AAAI*, pages 1720–1727.

Paninski, L. (2008). A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Trans. Inf. Theory*, 54:4750–4755.

Pote, Y. P. and Meel, K. S. (2021). Testing probabilistic circuits. In *Proc. of NeurIPS*, volume 34.

Soos, M., Gocht, S., and Meel, K. S. (2020). Tinted, detached, and lazy cnf-xor solving and its applications to counting and sampling. In *CAV (1)*, volume 12224, pages 463–484.

Soos, M., Nohl, K., and Castelluccia, C. (2009). Extending sat solvers to cryptographic problems. In *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257.

Thurley, M. (2006). sharpsat–counting models with advanced component caching and implicit bcp. In *SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 424–429.

Valiant, G. and Valiant, P. (2011a). Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new clts. In *Proc. of STOC*, pages 685–694.

Valiant, G. and Valiant, P. (2011b). The power of linear estimators. In *Proc. of FOCS*, pages 403–412.

Valiant, P. (2011). Testing symmetric properties of distributions. *SIAM J. Comput.*, 40:1927–1968.

Yuan, J., Aziz, A., Pixley, C., and Albin, K. (2004). Simplifying boolean constraint solving for random simulation-vector generation. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 23(3):412–420.

A EXTENDED PRELIMINARY

A.1 Useful Concentration Bounds

In our work, we use the following four concentration inequalities, see Dubhashi and Panconesi (2009).

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

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

Lemma A.2 (Chernoff-Hoeffding bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum_{i=1}^{n} X_i$ and $\mu_l \leq \mathbb{E}[X] \leq \mu_h$, the followings hold for any $\delta > 0$.*

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

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

Lemma A.3 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $a_i \leq X_i \leq b_i$ and $X = \sum_{i=1}^{n} X_i$. Then, for all $\delta > 0$,*

$$\mathbb{P}\left(|X - \mathbb{E}[X]| \geq \delta\right) \leq 2\exp\left(\frac{-2\delta^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

Lemma A.4. *Let $Z_1, \ldots, Z_n$ be n independent and identically distributed $0-1$ random variable. Then, the following hold:*

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

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

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

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

A.2 Chain Formulas

As mentioned briefly in the Preliminaries section (Section 2), we now formally define the notion of Chain formula as follows:

Chain Formula: The notion of chain formula was first studied in Chakraborty et al. (2015b). The structure of the chain formula has been exploited after its inception. Formally, a chain formula is defined as:

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

- Let $m > 0$ be a natural number and $k < 2^m$ be a positive odd number. Let $c_1 c_2 \dots c_m$ be the m-bit representation of k, where c_m is the Least Significant Bit (LSB) in the representation of m. For every $j \in \{1, \dots, m-1\}$, 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:

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

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

For example, for $k = 7$ and $m = 5$, as the binary representation of 6 in 5 bits is 00111, the corresponding chain formula would be $\varphi_{k,m}(a_1, a_2, a_3, a_4, a_5) = (a_1 \wedge (a_2 \wedge (a_3 \vee (a_4 \vee a_5))))$. We will omit the variables when it is clear from the context.

We briefly restate the following theorem from Chakraborty et al. (2015b).

Lemma A.5 (Chakraborty et al. (2015b))**.** *Given a natural number $m > 0$ and $k < 2^m$, and $\psi_{k,m}$ as constructed above. Then the size of the formula $\psi_{k,m}$, that is, $|\psi_{k,m}|$ is linear in m and $\psi_{k,m}$ has exactly k satisfying assignments.*

Since we need to convert the chain formula to an equivalent Horn formula while preserving all other properties of chain formula, we define a variant called *Horn-Chain-formula* as stated in Lemma 2.8 the Preliminaries section (Section 2). Before proceeding to prove the lemma, we restate it below.

Lemma A.6 (Lemma 2.8 restated)**.** *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 $j \in \{1, \dots, m\}$, 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 $\widetilde{a_1}$, 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.

□

A.3 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 Murphy (2012) 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 (2008), which is equivalent to log-linear models.

Definition A.7. 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 et al. (2020). However, instead of chain formulas, we use the notion of Horn-chain-formulas, introduced in Lemma A.6.

Lemma A.8. *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 A.6. 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:

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

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

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

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 $\sum_{\sigma_1 \in R_{\widehat{\varphi}} : \sigma_{1\downarrow S} = \sigma} 1 = \prod_{i \in \sigma_0} (2^{m_i} - k_i) \prod_{i \in \sigma_1} k_i$

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

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

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), we can say that

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

□

As it was also mentioned in Meel et al. (2020), we would like to emphasize the fact that the above lemma (Lemma A.8) 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 A.8 to test closeness to ideal Uniform-Horn-sampler.

B RELATED WORKS

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.

As it is often required for testing of samplers, 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 Valiant and Valiant (2011a) 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 Batu et al. (2000), Valiant (2011). Moreover, the basic problem of testing whether a distribution is uniform takes $\Theta(\sqrt{N})$ samples Paninski (2008), Goldreich and Ron (2011). 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. (2016) and Canonne et al. (2015). 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 (2019) 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 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 Canonne et al. (2015), 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 Chakraborty and Meel (2019) 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 Meel et al. (2020) used similar techniques in order to design the tester Barbarik2 for testing weighted samplers for CNF formulas. 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 (2021) studied the problem of equivalence testing in the context of testing probabilistic circuits.

In this work, we designed the first Weighted-Horn-sampler-tester Flash, along with the first Uniform-Horn-sampler-tester uFlash that have sound theoretical guarantees, which also work well in practice. As already mentioned in the introduction, the testers Barbarik and Barbarik2 do not work for Horn formulas. In order to achieve this, we defined the notion of Horn-chain-formula, as well as designed the subroutine Encode, which are crucial for our work.

C PROOF OF CORRECTNESS OF THE SUBROUTINE Encode

In this section, we provide the formal proof of correctness of our subroutine Encode, a crucial subroutine that is used in HornKernel. We formally state and prove the result below.

Lemma C.1. *(i) The formula Γ generated by* Encode *is a Horn formula.*

(ii) There are only three satisfying assignments of Γ, that is, $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$.

Proof. (i) First note that the clauses added by Encode are of size at most 2. Since any clause of size 1 is trivially a Horn clause, let us now analyse the size 2 clauses that have been added by Encode.

From the description of the subroutine Encode, it is clear that any clause of size 2 added by Encode is either one of the two forms: (a) $(x_1 \implies x_2)$, or (b) $(x_1 \implies \neg x_2)$, where x_1 and x_2 are two positive literals. Since both (a) and (b) are Horn clauses[12], the result follows.

(ii) We will prove this by using induction on the number of variables of Γ. Let us assume that Γ be a m variable Horn clause generated by Encode and $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$. Now we will prove that the same holds for any $(m+1)$ variable Horn formula generated by Encode.

[12] $(\neg x_1 \implies x_2)$ is not a Horn clause, so we can not add the clause $(x_1 \iff \neg x_2)$ which results in the possibility of $\widetilde{0} \in R_\Gamma$.

Let us now consider the value of the $(m+1)$th variable x_{m+1} in σ_1 and σ_2. Now there can be following 4 cases:

(a) $x_{m+1} = 0$ in both σ_1 and σ_2. In this case, we pick another common False literal, x_{fLit} (if it exists), and we add the clause $(x_{fLit} \iff x_{m+1})$, and conjuncts it with $\neg x_{fLit}$. If no such literal x_{fLit} exists, then we just add the clause $\neg x_{m+1}$. This ensures that $x_{m+1} = 0$ in Γ. Hence, following the induction hypothesis, we can conclude that, $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$ in this case.

(b) $x_{m+1} = 1$ in both σ_1 and σ_2. In this case, we first pick a common True literal x_{tLit}, and then proceeding in similar fashion as Case (a), we can infer that $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$ holds here as well.

(c) $x_{m+1} = 0$ in σ_1 and $x_{m+1} = 1$ in σ_2. Let x_{diff1} denote the first literal where $x_i = 1$ in σ_1 and $x_i = 0$ in σ_2, and x_{diff2} be the first literal where $x_i = 1$ in σ_2 and $x_i = 0$ in σ_1. In this case, we add the clause $(x_{m+1} \iff x_{diff2})$. Thus, if $x_{diff2} = 1$, then using the fact that $x_{m+1} = 1$ in σ_2 and the induction hypothesis, we infer that the sampled witness is σ_1 and when $x_{diff2} = 0$, then similarly the sampled witness is σ_0 or $\widetilde{0}$.
However, if no such x_{diff2} exists, then we have $diff2 = m+1$, and so we only add the clause $(x_{diff1} \implies \neg x_{m+1})$. Thus, if $x_{diff1} = 1$, then $x_{m+1} = 0$ and the sampled witness is σ_1 and if $x_{diff1} = 0$, then x_{m+1} can be both 0 or 1. Thus, if $x_{m+1} = 1$, then the sampled witness is σ_2 or if $x_{m+1} = 0$, then the sampled witness is $\widetilde{0}$.

(d) $x_{m+1} = 1$ in σ_1 and $x_{m+1} = 0$ in σ_2. Following similar arguments like Case (c), we can decide that, $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$.

Thus, $R_\Gamma = \{\sigma_1, \sigma_2, \widetilde{0}\}$ holds in all the cases, and this completes the proof of (ii).

□

D OUR UNIFORM-HORN-SAMPLER-TESTER uFlash

In this section we describe our Uniform-Horn-sampler-tester uFlash, which works for a very simple subclass (uniform) of general weighted Horn sampler. Our Uniform-Horn-sampler-tester uFlash, similar to Flash, takes as input a black-box Horn sampler $\mathcal{G}$, a Horn formula φ, three parameters $\varepsilon, \eta, \delta$, such that $\varepsilon \in (0, 1/3)$, $\eta > 9\varepsilon$, $\delta > 0$, and outputs ACCEPT with probability at least $1 - \delta$, if $\mathcal{G}$ is ε-close (in ℓ_∞ distance) to Uniform-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 $1 - \delta$.

The basic framework of our tester uFlash is very similar to Flash. The point of designing a Uniform-Horn-sampler-tester explicitly, in parallel to Flash, is to highlight that uFlash works with much lower number of samples[13] and time complexity as compared to Flash. The subroutines used by uFlash are the same subroutines used by Flash, that is, HornKernel (Algorithm 2), Encode (Algorithm 3), Bias (Algorithm 6), NewVars (Algorithm 4), and RemoveZeros (Algorithm 5). But there are subtle changes in the algorithm of uFlash which utilizes the simplicity of testing Uniform-Horn-sampler, as compared to Weighted-Horn-sampler, and makes uFlash fast.

Theorem D.1. *Given a Horn sampler $\mathcal{G}$, access to an ideal Uniform-Horn-sampler $\mathcal{I}_\mathcal{U}$, 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* uFlash *takes $\widetilde{\mathcal{O}}(\frac{1}{(\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2)})$ many samples, and:*

(i) If $\mathcal{G}$ is an ε-AAU Horn sampler, uFlash *outputs ACCEPT with probability $\geq (1-\delta)$.*

(ii) If $\mathcal{G}$ is η-far from $\mathcal{I}_\mathcal{U}$ and $\mathcal{G}$ is subquery consistent, uFlash *outputs REJECT with probability $\geq (1-\delta)$,*

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

D.1 Algorithm Description of uFlash

Unlike Flash, uFlash (Algorithm 7) first calculates the parameters T, N, M from input parameters ε, η, δ. Then uFlash draws t many samples from the Horn sampler $\mathcal{G}$ under test, and draw another sample 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, in the same way as Flash. Thus, Γ_1 is a set of samples obtained from the distribution $D_{\mathcal{G}(\varphi)}$, while Γ_2 is a set of samples drawn from uniform distribution $\mathcal{U}$ 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, to get a Horn formula $\hat{\varphi}$ that employs the conditioning. The rest of

[13] Sample complexity of uFlash is independent of tilt.

the algorithm of uFlash is very same as Flash. uFlash obtains M many satisfying assignments of $\hat{\varphi}$ using $\mathcal{G}$, the sampler-under-test in Line 17. Then it calls the subroutine RemoveZeros (Algorithm 5) in Line 18 to remove all the $\widetilde{0}$ and checks 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 occurrences of σ_1 and σ_2 is less than N, uFlash outputs REJECT and terminates the algorithm as in Flash. Otherwise, it employs the subroutine Bias in Line 21 to determine the fraction of witnesses of σ_1 (when projected on S) in the obtained sample set after applying RemoveZeros. If this fraction is more than T, then it outputs REJECT. If uFlash 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 uniform distribution $\mathcal{U}$.

Algorithm 7: uFlash ($\mathcal{G}, \mathcal{I}_{\mathcal{U}}, S, \varepsilon, \eta, \delta, \varphi$)

1 $t \leftarrow \frac{10}{\eta(\eta-9\varepsilon)} \log_e\left(\frac{1}{\delta}\right)$;
2 $z \leftarrow \log_e\left(\frac{2t}{\delta}\right)$;
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 $N \leftarrow \frac{8z \cdot H}{(H-L)^2}$;
7 $X \leftarrow 2\left(\frac{1-\varepsilon}{3-\varepsilon}\right)$;
8 $M \leftarrow \left(\frac{\sqrt{z}+\sqrt{z+4NX}}{2X}\right)^2$;
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 $\hat{\varphi} \leftarrow \mathsf{HornKernel}(\varphi, \sigma_1, \sigma_2, M)$;
17 $\Gamma_3 \leftarrow \mathcal{G}(\hat{\varphi}, S, M)$;
18 $\widehat{\Gamma}_3 \leftarrow \mathsf{RemoveZeros}(\Gamma_3, \sigma_1, \sigma_2)$;
19 **if** $\left|\widehat{\Gamma}_3\right| < N$ **then**
20 **return** REJECT
21 $Bias \leftarrow \mathsf{Bias}(\sigma_1, \widehat{\Gamma}_3, S)$;
22 **if** $Bias > T$ **then**
23 **return** REJECT
24 **return** ACCEPT

D.2 Proof of Correctness of uFlash

Before proceeding to prove the correctness of uFlash, let us first restate the theorem regarding uFlash (Theorem D.1).

Theorem D.2 (Theorem D.1 restated)**.** *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* uFlash *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, uFlash *outputs ACCEPT with probability at least $1-\delta$.*

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

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

D.2.1 Completeness Theorem

Theorem D.3 (Completeness)**.** *If the Horn sampler $\mathcal{G}$ is an ε-additive almost-uniform (AAU) Horn sampler, then* uFlash *outputs ACCEPT with probability at least $(1-\delta)$.*

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

Lemma D.4. *If $\mathcal{G}$ is an 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 D.5. *Given that $\widetilde{0}$ has appeared less than $(M-N)$ times out of the M samples, the probability that* uFlash *outputs REJECT in each iteration is at most $\frac{\delta}{2t}$.*

Assuming Lemma D.4 and Lemma D.5 hold, now we proceed to prove Theorem D.3.

Proof. Note that uFlash (Algorithm 7) outputs REJECT when either number of times $\widetilde{0}$ 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 D.4, we know that the probability of Case (i) is at most $\frac{\delta}{2t}$. Also, by Lemma D.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 $\frac{\delta}{t}$, uFlash outputs REJECT in any iteration. Since there are t many iterations of uFlash, the probability that uFlash outputs ACCEPT is at least $(1-\frac{\delta}{t})^t \geq 1-\delta$. This completes the proof of Theorem D.3.

□

Now we proceed to prove Lemma D.4.

Proof of Lemma D.4. Let us define the following binary random variable:

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

Since $\mathcal{G}$ is ε-AAU, from Definition 2.4 we can say that

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)} \leq \frac{1+\varepsilon}{1-\varepsilon} \tag{3}$$

Similarly, we have

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \leq \frac{1+\varepsilon}{1-\varepsilon} \tag{4}$$

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

$$\begin{aligned} \mathbb{E}[Y_j] &= \mathbb{P}_{\mathcal{G}}(Y_j = 1) \\ &= \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0}) \\ &= \frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})} \\ &\leq \frac{1+\varepsilon}{3-\varepsilon} \end{aligned} \tag{5}$$

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 and Equation 4.

Now consider the random variable Y defined as $Y = \sum_{j=1}^{M} Y_j$. Following Equation 5, we can say that, $\mathbb{E}[Y] \leq \frac{1+\varepsilon}{3-\varepsilon} M$.

Applying Chernoff bound A.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 uFlash.

□

Proof of Lemma D.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 uFlash. Now, recall the set $\widehat{\Gamma}_3$ from uFlash that contains only σ_1 and σ_2 after removing all $\widetilde{0}$ 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 uFlash.

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

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

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

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

□

D.2.2 Soundness Theorem

Theorem D.6 (Soundness)**.** *If $\mathcal{G}$ is η-far from the ideal Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$ and $\mathcal{G}$ is subquery consistent,* uFlash *outputs REJECT with probability at least $1-\delta$.*

Before proceeding to prove Theorem D.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 D.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 D.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 D.7 and Lemma D.8 hold, we are now ready to prove Theorem D.6.

Proof of Theorem D.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 D.7 and Lemma D.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 uFlash returns REJECT is:

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

As $t = \frac{10}{\eta(\eta-9\varepsilon)} \log_e \frac{1}{\delta}$, when $\mathcal{G}$ is η-far from the Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$, uFlash rejects $\mathcal{G}$ with probability at least $1-\delta$. This completes the proof of Theorem D.6.

□

Proof of Lemma D.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 D.8. Since $\mathcal{G}$ is η-far from the Uniform-Horn-sampler $\mathcal{I}_{\mathcal{U}}$, we can say that

$$\sum_{x \in W} \left(\mathbb{P}_{\mathcal{G}}(\varphi, x) - \frac{1}{N}\right) = \sum_{x \in W_{-1}} \left(\frac{1}{N} - \mathbb{P}_{\mathcal{G}}(\varphi, x)\right) \geq \frac{\eta}{2} \tag{6}$$

Therefore from Equation (6),

$$\frac{|W_{-1}|}{N} \geq \frac{\eta}{2} \tag{7}$$

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

Thus,

$$\sum_{x \in W_1} \mathbb{P}_{\mathcal{G}}(\varphi, x) \geq \frac{\eta}{2} - \frac{\eta + 9\varepsilon}{4} \geq \frac{\eta - 9\varepsilon}{4} \tag{8}$$

Hence, from the independence of σ_1 and σ_2 we have proven the Lemma from Equation (7) and Equation (8). □

D.2.3 Sample Complexity of uFlash

Theorem D.9. *The sample complexity of* uFlash *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 uFlash 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 $N = \frac{8zH}{(H-L)^2}$. 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 $M = \left(\frac{\sqrt{z}+\sqrt{z+4NX}}{2X}\right)^2 \leq \frac{z+4NX}{X^2}$, $X = 2\left(\frac{1-\varepsilon}{3-\varepsilon}\right)$, we can say that

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

□

E CORRECTNESS OF OUR WEIGHTED-HORN-SAMPLER-TESTER Flash

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

Theorem E.1. *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* Flash *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* $\mathcal{I}_{\mathcal{W}}$*,* Flash *outputs ACCEPT with probability at least* $1 - \delta$*.*

(ii) If $\mathcal{G}$ *is* η*-far from the Weighted-Horn-sampler* $\mathcal{I}_{\mathcal{W}}$*,* Flash *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* $\frac{1}{\delta}$*.*

E.1 Completeness Property of Flash

Theorem E.2 (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* $(1 - \delta)$*,* Flash *will output ACCEPT.*

Proof of Theorem E.2. To begin with, first note that, the algorithm of Flash runs the for loop for t many times. To prove the Theorem E.2, 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 Flash 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 E.3 and Lemma E.4.

The proof of Theorem E.2 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 Flash is at most $\delta/2t$.

Lemma E.3. *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 E.4. *Given that $\widetilde{0}$ has appeared less than $(M - N)$ times out of the M samples, the probability that* Flash *outputs REJECT in any iteration is at most $\frac{\delta}{2t}$.*

Assuming Lemma E.3 and Lemma E.4 hold, we now prove Theorem E.2 as follows:

First note that $\hat{\varphi}$ has three satisfying assignments: $\{\sigma_1, \sigma_2, \widetilde{0}\}$ 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:

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

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

Case (i) When we draw M samples in Line 19 of Flash (Algorithm 1), we obtain more than $(M - N)$ many $\widetilde{0}$, that is, $|\mathcal{I}| < N$. In this case, Flash 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 Flash 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 E.3 and Lemma E.4.

Since the probability that Flash outputs REJECT in some i-th iteration is at most $\frac{\delta}{t}$, then the probability that Flash 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 Flash outputs ACCEPT with probability at least $1 - \delta$.

□

Proof of Lemma E.3. Recall from the Algorithm 1 in Line 19, we sample M many samples from $\hat{\varphi}$, which are contained in Γ_3. So Γ_3 consists of witnesses from the set $\{\sigma_1, \sigma_2, \widetilde{0}\}$. Let us first define the following binary random variable,

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

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,

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)} \leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \widetilde{0})}{(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1)} \tag{9}$$

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \widetilde{0})}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \widetilde{0})}{(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2)} \tag{10}$$

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

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

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 (9) and Equation (10).

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:

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

Let us define τ as $\tau = (1-\varepsilon)wt\left(\sigma_1\right) + (1-\varepsilon)wt\left(\sigma_2\right) + (1+\varepsilon)wt\left(\widetilde{0}\right)$. 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 A.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.

□

Proof of Lemma E.4. When M samples drawn from the sampler $\mathcal{G}$ in Line 19, if at least N many samples belong to the set $\{\sigma_1, \sigma_2\}$, then Flash outputs REJECT if and only if $Bias$ is more then T. Let us now consider an iteration j among the t iterations of Flash. Recall the set $\mathcal{I}$ on Γ_3 defined as $\mathcal{I} = \left\{j \mid \Gamma_3[j]_{\downarrow S} \neq \widetilde{0}\right\}$. 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,

$$\mathbb{E}[Y_j] = \frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1) + \mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \tag{11}$$

Since $\mathcal{G}$ is ε-close to ideal Weighted-Horn-sampler $\mathcal{I}_{\mathcal{W}}$, from Definition 2.4, we can say that:

$$\frac{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_1)}{\mathbb{P}_{\mathcal{G}}(\hat{\varphi}, S, \sigma_2)} \leq \frac{(1+\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1)}{(1-\varepsilon)\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2)} \tag{12}$$

Following the facts that $\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_1) = \frac{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_1)}{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_1)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_2)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \widetilde{0})}$ and $\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\hat{\varphi}, S, \sigma_2) = \frac{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_2)}{\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_1)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \sigma_2)+\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi, S, \widetilde{0})}$, from Equation (11) and Equation (12), 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 Flash 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 A.4) as follows:

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

□

E.2 Soundness Property of Flash

Theorem E.5 (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 $(1-\delta)$,* Flash *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 Flash will output REJECT with high probability. In order to prove this, we need the following two lemmas.

Lemma E.6. *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 E.7. *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 E.6 and Lemma E.7, we can estimate the probability of rejection of Flash 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 Flash 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 E.6. 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,

$$\begin{aligned}
&\mathbb{P}\left(Bias \leq T \textit{ in } i\textit{-th iteration } \mid (\sigma_1 \in W_1 \wedge \sigma_2 \in W_{-1})\right) \\
&\leq \exp\left(-\frac{(H - L)^2 N}{8H}\right) \\
&\leq \frac{\delta}{2t} && \text{[applying Chernoff bound A.4]} \\
&\leq \frac{1}{5} && \text{[as } \delta < 0.5 \text{ and } t \geq 2\text{]}
\end{aligned}$$

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

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

□

Proof of Lemma E.7. 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,

$$\sum_{x\in W_0\cup W_1}(\mathbb{P}_{\mathcal{G}}(\varphi,x)-\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x))=\sum_{x\in W_{-1}}(\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x)-\mathbb{P}_{\mathcal{G}}(\varphi,x))\geq\frac{\eta}{2} \tag{13}$$

Therefore from Equation (13),

$$\sum_{x\in W_{-1}}\mathbb{P}_{\mathcal{I}_{\mathcal{W}}}(\varphi,x)\geq\frac{\eta}{2} \tag{14}$$

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,

$$\sum_{x\in W_1}\mathbb{P}_{\mathcal{G}}(\varphi,x)\geq\frac{\eta}{2}-\frac{\eta+9\varepsilon}{4}=\frac{\eta-9\varepsilon}{4} \tag{15}$$

Since the events $\sigma_1\in W_1$ and $\sigma_2\in W_{-1}$ are independent, from Equation 14 and Equation 15, we can say that

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

□

E.3 Sample Complexity of Flash

Theorem E.8. *The sample complexity of* Flash *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 poly-logarithmic 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 Flash (Algorithm 1) takes t samples in the for loop in Line 5. Also, it takes M samples in Line 6 in each iteration of the for loop. So, Flash 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 Flash is $\widetilde{\mathcal{O}} \left(\frac{\mathsf{tilt}^3}{\eta(\eta-9\varepsilon)(\eta-3\varepsilon)^2} \right)$. This completes the proof of Theorem E.8.

□

F EXTENDED EXPERIMENTAL RESULTS

In this section, we describe the extended experimental results of our Uniform-Horn-sampler-tester uFlash and Weighted-Horn-sampler-tester Flash. As mentioned in the Evaluation Results section of our main paper (Section 4), 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. As mentioned in the main paper, 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 [Thurley (2006)]. 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 uFlash. Moreover, in the evaluation of our Weighted-Horn-sampler-tester Flash, 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.

F.1 Results of our Uniform-Horn-sampler-tester uFlash:

Here we present the extended results of our Uniform-Horn-sampler-tester uFlash. For baseline, we employed the tester from Batu et al. (2001), Paninski (2008). 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 3, we compare our Uniform-Horn-sampler-tester uFlash with respect to the sampler UniGen to the baseline over the 11 benchmark instances. 1st column of Table 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 uFlash, and the output of uFlash with respect to that particular benchmark instance. We find that uFlash outputs ACCEPT in all 11 benchmark instances. As it is evident from the entries of the table, uFlash 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 4, we compare our Uniform-Horn-sampler-tester uFlash when run with the sampler QuickSampler along with the baseline tester over the 11 benchmark instances. Similar to Table 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 uFlash, and the output of uFlash on that instance. It turns out that uFlash 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 uFlash is much smaller compared to the baseline approach.

Finally, in Table 5, we present our results when we run our Uniform-Horn-sampler-tester uFlash for the sampler STS. The organization of this table follows in similar fashion as Table 3 and Table 4. As in the case of QUICKSAMPLER, uFlash outputs REJECT in all 11 instances here as well. Similar to the case of UNIGEN and QUICKSAMPLER, it is clear that uFlash 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		uFlash		
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 3: Evaluation results of uFlash with UNIGEN

		Baseline		uFlash		
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 4: Evaluation results of uFlash with QUICKSAMPLER

		Baseline		uFlash		
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: Evaluation results of uFlash with STS

F.2 Results of our Weighted-Horn-sampler-tester Flash:

In this section, we present the extended results of the experiments of our Weighted-Horn-sampler-tester Flash to test the samplers WUNIGEN, WQUICKSAMPLER and WSTS. As mentioned in the main paper, for each of the 11 benchmark

				Baseline		wUniGen		
	Benchmark	Model Count	tilt	#Samples	Time (s)	#Samples	Time (s)	o/p
Benchmark-I	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
Benchmark-II	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 6: Evaluation results of Flash with wUniGen

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. (2000). For any Horn formula φ, the tester of Batu et al. (2000) 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 uFlash, as the sample complexity of Batu et al. (2000) is very large, we have estimated the average time taken for any benchmark instance by the baseline tester.

In Table 6, we present the results of our Weighted-Horn-sampler-tester Flash to test the sampler wUniGen. 1st column of Table 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 Flash and its output on that benchmark instance respectively. Among the 22 benchmark instances, in 8 instances, Flash outputs ACCEPT, whereas there are 6 instances of TLE and 8 instances of DNS. For the instances where Flash does not output TLE or DNS, it is clear that Flash outperforms the baseline tester by a large order of magnitude.

In Table 7, we present our results of Flash with respect to wQuickSampler. The organization of this table is similar to that of Table 6. Our Weighted-Horn-sampler-tester Flash 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 Flash is much better compared to the baseline approach.

Finally, in Table 8, we show the results of Flash in order to test the sampler STS. The organization of the table follows in similar line to the preceding tables. It turns out that Flash outputs REJECT in all of the 22 benchmark instances here. Also, similar to wUniGen and wQuickSampler, Flash outperforms the baseline tester by a large magnitude.

				Baseline		wQuickSampler		
	Benchmark	Model Count	tilt	#Samples	Time (s)	#Samples	Time (s)	o/p
Benchmark-I	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
Benchmark-II	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 7: Evaluation results of Flash with wQuickSampler

				Baseline		wSTS		
	Benchmark	Model Count	tilt	#Samples	Time (s)	#Samples	Time (s)	o/p
Benchmark-I	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
Benchmark-II	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 8: Evaluation results of Flash with wSTS