6.842 Randomness and Computation February 16, 2022

Lectures on Uniform Generation and Approximate Counting

Lecturer: Ronitt Rubinfield *Scribe: Yuchong Pan*

1 Uniform Generation for DNF

A *DNF formula* is a Boolean formula that consists of "OR of ANDs", e.g., $\varphi(x_1, x_2, x_3) = x_1\overline{x_2} \vee \overline{x_1}x_3$. The task is to output a random satisfying assignment to a DNF formula, uniformly over all satisfying assignments.

Consider the special case with only one conjunction. We can satisfy literals in the conjunction, and set the others randomly. For instance, for $\varphi(x_1, x_2, x_3) = x_1\overline{x_2}$, we set $x_1 = \mathrm{T}, x_2 = \mathrm{F}$, and set $x_3 \in \{\mathrm{T}, \mathrm{F}\}$ uniformly.

Now consider the two conjunction case, e.g., $\varphi(x_1, x_2, x_3) = x_1x_2 \vee x_3$. We give an algorithm attempt in Algorithm 1. However, two problems arise, as illustrated in Figure 1. Firstly, conjunctions have different numbers of assignments. Secondly, some assignments can be chosen in multiple ways. In particular, using Algorithm 1, the probability that TTF is chosen is $1/2 \cdot 1/2 = 1/4$, the probability that TTT is chosen is $1/2 \cdot 1/2 + 1/2 \cdot 1/4 = 3/8$, and that the probability that each of TFT, FTT, FFT is chosen is $1/2 \cdot 1/4 = 1/8$.

1 pick $i \in \{1, 2\}$ uniformly
2 set variables in conjunction i to T
3 set others uniformly

Algorithm 1: An algorithm attempt of uniform generation for DNF with 2 conjunctions.

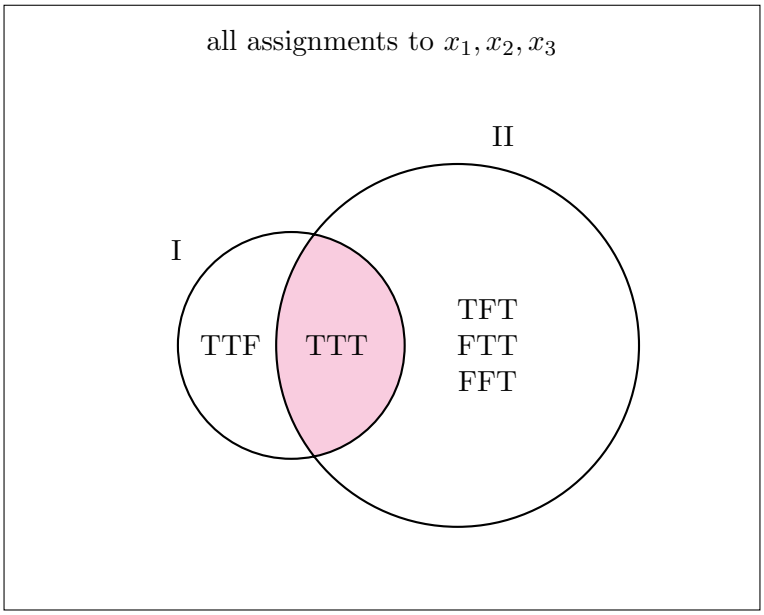

Figure 1: An illustration of problems in Algorithm 1.

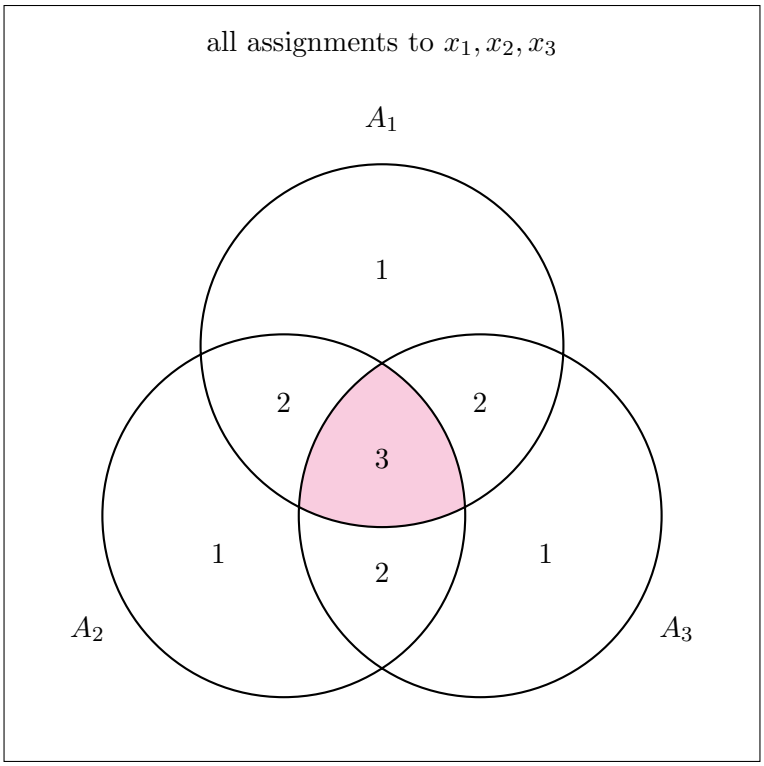

Figure 2: An example for rejection sampling, where the number in each region indicates the number of conjunctions each assignment satisfies.

The first problem is easy to fix; we simply replace line 2 in Algorithm 1 to "pick $i \in \{1, 2\}$ *proportional to the number of satisfying assignments for that conjunction.*" For the second problem, we introduce the notion of *rejection sampling*. Let m be the number of conjunctions in φ. For each $i \in [m]$, let $A_i = \{\bar{x} = (x_1, \ldots, x_n) : \bar{x} \text{ satisfies the } i^{\text{th}} \text{ conjunction}\}$. Consider the example illustrated in Figure 2. In the shaded region in Figure 2, each assignment is 3 times more likely to be picked; therefore, we correct our algorithm by tossing a coin of bias $1/3$ to decide if to output the assignment. We summarize our final algorithm in Algorithm 2.

1 $A_i = \{\bar{x} = (x_1, \ldots, x_n) : \bar{x}$ satisfies the i^{th} conjunction$\}$ for each $i \in [m]$
2 **repeat**
3 pick $i \in [m]$ with probability $|A_i| / \sum_{j=1}^{m} |A_j|$
4 pick $\bar{b} \in A_i$ uniformly
5 $t_{\bar{b}} \leftarrow |\{j \in [m] : \bar{b} \in A_j\}|$ // $t_{\bar{b}} \geq 1$ since $\bar{b}$ satisfies C_i
6 output $\bar{b}$ with probability $1/t_{\bar{b}}$
7 **until** *success*

Algorithm 2: Uniform generation for DNF, where the input is a DNF formula $\varphi = \bigvee_{i=1}^{m} C_i$ and C_i is a conjunction for each $i \in [m]$.

We show the uniformity of Algorithm 2. For each $\bar{b}$ that satisfies φ,

$$\mathbb{P}\left[\bar{b} \text{ is output in round } i\right] = \frac{1}{t_{\bar{b}}} \sum_{\substack{j \in [m] \\ \bar{b} \in A_j}} \mathbb{P}[\text{pick } j \text{ in round } i] \cdot \frac{1}{|A_j|}$$

$$= \frac{1}{t_{\bar{b}}} \sum_{\substack{j \in [m] \\ \bar{b} \in A_j}} \frac{|A_j|}{\sum_{k=1}^{m} |A_k|} \cdot \frac{1}{|A_j|}$$
$$= \frac{1}{t_{\bar{b}}} \sum_{\substack{j \in [m] \\ \bar{b} \in A_j}} \frac{1}{\sum_{k=1}^{m} |A_k|}$$
$$= \frac{1}{\sum_{k=1}^{m} |A_k|}.$$

Hence, the probability that $\bar{b}$ is output in round i is the same for all $\bar{b}$ satisfying φ, so the generation is uniform. Moreover, $\mathbb{P}[\text{loop success}] \geq 1/\max t_{\bar{b}} \geq 1/m$, so $\mathbb{E}[\#\text{ loops until success}] \leq m$. This shows that Algorithm 2 runs in polynomial time in expectation.

2 Counting Problems

Definition 1. Let #P denote the class of problems that count the number of accepting paths in a polynomial time nondeterministic Turing machines.

Definition 2. We say that a problem is #P-*complete* if

(a) it is in #P;

(b) every problem in #P has a polynomial time reduction to it.

Let #SAT denote the problem that counts the number of assignments satisfying a Boolean formula φ. Then #SAT is #P-complete. Similarly, let #DNF denote the problem that counts the number of assignments satisfying a DNF formula φ. Although DNF $\in$ P, we show below that #DNF $\notin$ P. However, it turns out that #DNF is #P-complete.

Proposition 3. #DNF $\notin$ P.

Proof. We show that if #DNF $\in$ P, then CNF $\in$ P. By De Morgan's law, a clause $\overline{A \vee B} = \overline{A} \wedge \overline{B}$, which is a conjunction. Therefore, Given a CNF formula φ in n variables, φ is satisfiable if and only if $\overline{\varphi}$ (which is a DNF formula) has at most $2^n - 1$ satisfying assignments. □

3 Approximate Counting

Definition 4. A *fully polynomial randomized approximation scheme (FPRAS)* for DNF is an algorithm which, given a DNF formula φ and $\varepsilon > 0$, outputs $y \in \mathbb{Z}_+$ such that $z/(1+\varepsilon) \leq y \leq z(1+\varepsilon)$ with probability at least 3/4 in time polynomial in $|\varphi$ and $1/\varepsilon$, where z is the number of satisfying assignments to φ.

In Problem Set 1 Problem 1, we shall show that Definition 4 implies that an FPRAS runs in time polynomial in $|\varphi|$, $1/\varepsilon$ and $\log \delta$ if we replace the probability bound 3/4 with $1 - \delta$.

Proposition 5. *If there exists an FPRAS for* #SAT*, then* SAT $\in$ BPP.

Proof. We give a probabilistic algorithm for SAT. Given a Boolean formula φ, we call the FPRAS on φ with $\varepsilon = 1/2$ (or any constant); if the output of the FPRAS is positive, then output "satisfiable"; else output "unsatisfiable."

If φ is satisfiable, then the number of satisfying assignments is at least 1, so $y \geq 1/(1+\varepsilon) > 0$, which impies that the algorithm outputs "satisfiable." Otherwise, the number of satisfying assignments is 0, so $y = 0$, which implies that the algorithm outputs "unsatisfiable." □