Pairwise Independence and Derandomization

Michael Luby and Avi Wigderson

July, 1995

Abstract

This set of notes gives several applications of the following paradigm. The paradigm consists of two complementary parts. The first part is to design a probabilistic algorithm described by a sequence of random variables so that the analysis is valid assuming limited independence between the random variables. The second part is the design of a small probability space for the random variables such that they are somewhat independent of each other. Thus, the analysis of the algorithm holds even when the random variables used by the algorithm are generated according to the small space.

Acknowledgments

Avi Wigderson was invited as a Visiting McKay Professor to UC Berkeley for the month of February, 1995. Michael Luby was invited as a Visiting Professor to École Normale Supérieure in Paris for the month of June, 1995. Based on the series of lectures given during these two respective visits, Michael Luby compiled these notes into their current form,

We would like to thank the following people for their help in producing these notes. Preliminary drafts of notes based on Avi's lectures (Sections 1–15) were prepared by Sanjoy Dasgupta, Sean Hallgren, Elizabeth Sweedyk, and Eric Vigoda from UC Berkeley, and by Johannes Blömer and Ralph Werchner from the International Computer Science Institute. Eric Vigoda from UC Berkeley and Dana Randall from the Institute for Advanced Studies visited École Normale Supérieure during parts of the month of June, 1995, supported by a grant from the International Branch of the NSF. As part of their activities, Eric helped substantially in compiling these notes during this time, and Dana proofread the entire set of notes, made many good suggestions and found a number of glaring mistakes. Avy Sharell from Orsay prepared a preliminary draft of the notes for Section 23.

Table of Contents

Section 1: Pairwise Independence

Consider a set of random variables indexed by a set U, i.e., $\{Z_i : i \in U\}$, where $Z_i \in T$. For finite $t = |T|$, a uniform distribution assigns $\Pr[Z_x = \alpha] = 1/t$, for all $x \in U$, $\alpha \in T$. If this distribution were furthermore pairwise independent, we would have: for all $x \neq y \in U$, for all $\alpha, \beta \in T$,

$$\Pr[Z_x = \alpha, Z_y = \beta] = \Pr[Z_x = \alpha] \cdot \Pr[Z_y = \beta] = 1/t^2.$$

This is not the same as complete independence, as evidenced by the following set of three pairwise-independent variables ($U = \{1, 2, 3\}$, $T = \{0, 1\}$, $t = 2$):

s	Z_1	Z_2	Z_3
00	0	0	0
01	0	1	1
10	1	0	1
11	1	1	0

Each row s can be thought of as a function $h_s : U \to T$. Let $\mathcal{S}$ be the index set for these functions, where in this case $\mathcal{S} = \{0, 1\}^2$. For all $x \neq y \in U$, for all $\alpha, \beta \in T$,

$$\Pr_{s \in_R \mathcal{S}}[h_s(x) = \alpha \wedge h_s(y) = \beta] = 1/4 = 1/t^2$$

(Notice in particular that $\Pr_{s \in_R \mathcal{S}}[h_s(x) = h_s(y)] = 1/2 = 1/t$.) Any set of functions satisfying this condition is a 2-universal family of hash functions. Definitions and explicit constructions of 2-universal hash functions were first given by [36, Carter-Wegman] . The original applications described in [36, Carter-Wegman] were straightforward, similar to those described in the later section on hashing. As these notes indicate, subsequently 2-universal hashing has been applied in surprising ways to a rich variety of problems.

Section 2: Constructing Hash Functions

One simple way to construct a family of hash functions mapping $\{0,1\}^n \rightarrow \{0,1\}^n$ is to let $\mathcal{S} = \{0,1\}^n \times \{0,1\}^n$, and then for all $s = (a,b) \in \mathcal{S}$, for all $x \in \{0,1\}^n$ define $h_s(x) = ax + b$, where the arithmetic operations are with respect to the finite field GF$[2^n]$. Thus, each h_s maps $\{0,1\}^n \rightarrow \{0,1\}^n$ and $\mathcal{S}$ is the index set of the hash functions. For each $s = (a,b) \in \mathcal{S}$, we can write:

$$\begin{pmatrix} h_s(x) \\ h_s(y) \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

When $x \neq y$, the matrix is non-singular, so that for any $x, y \in \{0,1\}^n$, the pair $(h_s(x), h_s(y))$ takes on all 2^{2n} possible values (as s varies over all $\mathcal{S}$). Thus if s is chosen uniformly at random from $\mathcal{S}$, then $(h_s(x), h_s(y))$ is also uniformly distributed. This property of hash functions is called *2-universal.*

We can view $\mathcal{S}$ as the set of points in a sample space on the set of random variables $\{Z_x : x \in \{0,1\}^n\}$ where $Z_x(s) = h_s(x)$ for all $s \in \mathcal{S}$. With respect to the uniform distribution on $\mathcal{S}$, these random variables are pairwise independent, i.e., for all $x \neq y \in \{0,1\}^n$, for all $\alpha, \beta \in \{0,1\}^n$

$$\Pr_{s \in_R S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] = \Pr_{s \in_R S}[Z_x(s) = \alpha] \cdot \Pr_{s \in_R S}[Z_y(s) = \beta] = 1/2^{2n}.$$

To obtain a hash function that maps to $k < n$ bits, we can still use $\mathcal{S}$ as the function index family: The value of of the hash function indexed by s on input x is obtained by computing $h_s(x)$ and using the first k bits.

The imporant properties of these hash functions are:

- Pairwise independence.
- Succinctness – each function can be described as a $2n$-bit string. Therefore, randomly picking a function index requires only $2n$ random bits.
- The function $h_s(x)$ can easily be computed (in LOGSPACE, for instance) given the function index s and the input x.

In the sequel, unless otherwise specified we are referring to this set of pairwise independent hash functions and $\mathcal{S}$ denotes the set of indices for the hash family.

Consider, for example, the MAXCUT problem: given a graph $G = (V, E)$, find a two-coloring of the vertices $\chi : V \rightarrow \{0, 1\}$ so as to maximize $c(\chi) = |\{(x, y) \in E : \chi(x) \neq \chi(y)\}|$. We describe a solution to this problem that is guaranteed to produce a cut where at least half the edges cross the cut.

If the vertices are colored randomly (0 or 1 with probability 1/2) by choosing χ uniformly from the set of all possible $2^{|V|}$ colorings, then:

$$\mathrm{E}[c(\chi)] \quad = \quad \sum_{(x,y)\in E} \Pr[\chi(x) \neq \chi(y)] \quad = \quad \frac{|E|}{2}$$

Thus, there must always be a cut of size at least $\frac{|E|}{2}$. Let $\mathcal{S}$ be the index set for the hash family mapping $V \rightarrow \{0, 1\}$. Since the summation above only requires the coloring of vertices to be pairwise-independent, it follows that $\mathrm{E}[c(h_s)] = \frac{|E|}{2}$ when $s \in_R \mathcal{S}$. Since $|\mathcal{S}| = |V|^2$, we can deterministically try h_s for all $s \in \mathcal{S}$ in polynomial time (even in the parallel complexity class **NC**), and for at least one $s \in \mathcal{S}$, h_s defines a partition of the nodes where at least $\frac{|E|}{2}$ edges cross the partition.

This derandomization approach was developed and discussed in general terms in the series of papers [9, Chor-Goldreich], [26, Luby], [5, Alon-Babai-Itai]. There, the approach was applied to derandomize algorithms such as witness sampling, a fast parallel algorithm for finding a maximal independent set, and other graph algorithms.

Section 4: Hashing

Say we are mapping elements $N \subseteq U$ into a table T, with $t = |T|, n = |N|$. For our hash family mapping $U \to T$, if we choose $s \in_R \mathcal{S}$, what is the expected number of colliding pairs C?

$$\mathrm{E}[C] \quad = \quad \sum_{x \neq y \in N} \Pr_{s \in_R \mathcal{S}}[h_s(x) = h_s(y)] \quad = \quad \binom{n}{2} \cdot \frac{1}{t}$$

For instance, if $t = n^2$, then $\mathrm{E}[C] \leq \frac{1}{2}$ (and so the probability that h is 1-to-1 is $\geq \frac{1}{2}$). If $t = n$, then $\mathrm{E}[C] \leq \frac{n}{2}$.

One application that uses hashing and this observation is a particular implementation of a dictionary. Consider a model where we first insert a subset of entries N from some universe of possible words U into a dictionary, and then subsequently we want to be able to look-up possible words $x \in U$ to see if they are in N. Deterministic schemes based on balanced tree data structures do the insertion in time $\mathcal{O}(n \log n)$ and subsequent look-ups in time $\mathcal{O}(\log n)$ each.

Random hashing can speed this up considerably. A straightforward implementation of a dictionary using hashing is to hash U to a table with $\mathcal{O}(n^2)$ cells, where each cell can store an element of U, in such a way that no two entries in N map to the same cell. This scheme takes time $\mathcal{O}(n)$ to store N (assuming the table space does not have to be initialized, which is unrealistic) and constant time for each look-up.

The following two-level hashing scheme, due to [11, Fredman-Komlós-Szemerédi], also takes time $\mathcal{O}(n)$ to construct the dictionary and constant time for each look-up, but the advantage is that it uses only $\mathcal{O}(n)$ cells in total. Let $T = \{1, \ldots, n\}$.

(1) Pick $s \in_R \mathcal{S}$ and map N into T. For each $i \in T$, let N_i be the subset of N mapped to i by h_s, and let $n_i = |N_i|$. Let $C = \sum_{i \in T} \binom{n_i}{2}$ be the number of colliding pairs. If $C > n$ then start over at step (1), else go on to step (2).

(2) For each $i \in T$, if $n_i \geq 1$ then we allocate a table T_i of n_i^2 cells, and let $\mathcal{S}_i$ denote the index set for the hash family mapping $U \to T_i$. Pick $s_i \in_R \mathcal{S}_i$, and use h_{s_i} to map N_i to T_i. If h_{s_i} maps N_i 1-to-1 into T_i then this is a good choice for s_i, else rechoose s_i independently and try again until h_{s_i} does describe a 1-to-1 mapping.

Because $\mathrm{E}[C] \leq n/2$ in step (1), $\Pr[C \leq n] \geq 1/2$, and thus the expected number of times step (1) is repeated is at most 2. Similarly, in step (2), for each $i \in T$, the expected number of times till the mapping of N_i into T_i is 1-to-1 is at most 2. Thus, the overall expected time to construct the dictionary is $\mathcal{O}(n)$. The total number of cells used to store N is $D = \sum_{i \in T} n_i^2$. Noting that $D - 2C = |N| = n$, and that $C \leq n$, it follows that at most $3n$ cells are used to store N. Note that we need to also store s and all s_i for all $i \in \{1, \ldots, n\}$, but this takes at most $2(n+1)$ additional cells, since the description of each hash function takes two cells.

Each find operation takes constant time.

Section 5: RP and BPP

Recall that a language $\mathcal{L} \in \mathbf{NP}$ if there is a polynomial time **TM** M (where **TM** denotes a Turing machine) with the following properties. M has two inputs x and y, where x is the string for which membership in $\mathcal{L}$ is trying to be decided, and y is a potential witness for membership of x in $\mathcal{L}$. If $x \in \{0,1\}^n$ then $y \in \{0,1\}^r$, where r is polynomial in n. The output of $M(x,y)$ is a single bit. The running time of $M(x,y)$ is polynomial in $\|x\|$. For $x \in \{0,1\}^n$, let $W_x = \{y \in \{0,1\}^r : M(x,y) = 1\}$. The machine has the property that for all $x \in \{0,1\}^n$,

$$\begin{aligned} x \in \mathcal{L} &\Rightarrow |W_x| > 0, \\ x \notin \mathcal{L} &\Rightarrow |W_x| = 0. \end{aligned}$$

RP is the class of languages $\mathcal{L}$ where membership can be checked with one-sided error by a randomized, polynomial-time **TM**. Keeping the same notation as above, $\mathcal{L} \in \mathbf{RP}$ if there is **TM** M with the following properties. There is a constant $c_{\mathrm{yes}} > 0$ associated with M. For $x \in \{0,1\}^n$, let $W_x = \{y \in \{0,1\}^r : M(x,y) = 1\}$. For any $A \subseteq \{0,1\}^r$, let $\mu(A) = |A|/2^r$ be the fraction of r-bit strings which are in A. M has the property that for all $x \in \{0,1\}^n$,

$$\begin{aligned} x \in \mathcal{L} &\Rightarrow \mu(W_x) \geq c_{\mathrm{yes}}, \\ x \notin \mathcal{L} &\Rightarrow \mu(W_x) = 0. \end{aligned}$$

The way we can decide membership of $x \in \{0,1\}^n$ is to choose $y \in_R \{0,1\}^r$ and decide $x \in \mathcal{L}$ if $M(x,y) = 1$, i.e., if $y \in W_x$, and decide $x \notin \mathcal{L}$ if $M(x,y) = 0$, i.e., if $y \notin W_x$. Notice that the decision is always correct if $x \notin \mathcal{L}$, but the decision is only correct with probability c_{yes} if $x \in \mathcal{L}$. On the other hand, when $x \in \mathcal{L}$, if $y \in W_x$ then y is a witness to the fact that x really is in $\mathcal{L}$, i.e., we can have full confidence in our decision. The standard way to boost confidence in the decision is to choose $y_1, \ldots, y_k \in_R \{0,1\}^r$ and decide $x \in \mathcal{L}$ if, for any $i \in \{1, \ldots, k\}$, $y_i \in W_x$. Then, the probability of making an incorrect decision when $x \in \mathcal{L}$ is reduced to $(1 - c_{\mathrm{yes}})^k$.

BPP is the class of languages $\mathcal{L}$ where membership can be checked with two-sided error by a randomized, polynomial-time **TM**. Keeping the same notation as above, $\mathcal{L} \in \mathbf{BPP}$ if there are constants c_{yes} and c_{no} with $c_{\mathrm{yes}} > c_{\mathrm{no}}$, such that for all $x \in \{0,1\}^n$,

$$\begin{aligned} x \in \mathcal{L} &\Rightarrow \mu(W_x) \geq c_{\mathrm{yes}}, \\ x \notin \mathcal{L} &\Rightarrow \mu(W_x) \leq c_{\mathrm{no}}. \end{aligned}$$

We can decide membership in $\mathcal{L}$ exactly the same as for **RP**, but then there is a chance of making an incorrect decision both in the case when $x \in \mathcal{L}$ and when $x \notin \mathcal{L}$. The way to boost confidence in the decision is similar to that for **RP**: choose $y_1, \ldots, y_k \in_R \{0,1\}^r$ and decide $x \in \mathcal{L}$ if $y_i \in W_x$ for more than $k(\mathrm{c}_{\mathrm{no}} + \mathrm{c}_{\mathrm{yes}})/2$ of the $i \in \{1, \ldots, k\}$, and decide $x \notin \mathcal{L}$ otherwise.

Section 6: Complexity of Unique Solutions

NP-hard problems often have many possible solutions. Would it make it easier if we were assured of a unique solution? Specifically, we say an algorithm $A \in \mathbf{RP}$ *unique-solves* an **NP** language $\mathcal{L}$ if, for all x, the output of A is guaranteed to be "no" with high probability if $|W_x| = 0$ and the output of A is guaranteed to be "yes" with high probability if $|W_x| = 1$. Notice there is no requirement on the output of A if $|W_x|$ is neither 0 nor 1, i.e., A can output anything in this case. Because of this, A cannot be used directly to decide membership in $\mathcal{L}$. [35, Valiant-Vazirani] nevertheless show that A can be used indirectly to efficiently decide membership in $\mathcal{L}$. More specifically, [35, Valiant-Vazirani] show that if there is an $A \in \mathbf{RP}$ that unique-solves some **NP**-complete language $\mathcal{L}$ then $\mathbf{RP} = \mathbf{NP}$. The idea behind [35, Valiant-Vazirani] follows.

Consider the following language CIRCUIT SAT: Given a circuit C with an r-bit input, is there a $y \in \{0,1\}^r$ such that $C(y) = 1$? A slightly more general **NP**-complete problem, Π, is the following: Given a circuit C with an r-bit input, a function h mapping $\{0,1\}^r$ to some set of values T, and a value $\alpha \in T$, is there an input y to C such that $C(y) = 1$ and $h(y) = \alpha$?

Theorem : If there is an algorithm $A \in \mathbf{RP}$ which unique-solves Π then $\mathbf{RP} = \mathbf{NP}$.

PROOF: We design an algorithm $B \in \mathbf{RP}$ that decides membership in CIRCUIT SAT based on A. On input a circuit C with an r-bit input, B works as follows:

- Choose $k \in_R \{1, \ldots, r\}$.
- Choose $s \in_R \mathcal{S}$, where $\mathcal{S}$ is the index set of the hash family that maps $U = \{0,1\}^r$ to $T = \{0,1\}^{k+1}$.
- Choose $\alpha \in_R \{0,1\}^{k+1}$.
- Call $A(C, h_s, \alpha)$. Give the same answer as A.

Note that $||s|| = \mathcal{O}(r)$, and so this reduction can be performed in random polynomial time. If C is not satisfiable, then clearly B will respond "no". If C is satisfiable, then for some $k \in \{1, \ldots, r\}$, $2^{k-1} \le N \le 2^k$, where N is the number of satisfying assignments (witnesses) to C. With probability $1/r$ we guessed this k correctly.

Assume we have the correct k. Previously, we saw that for a table T, with $|T| = aN$, the expected number of colliding pairs $\mathrm{E}[C] \leq N/(2a)$. In our case, $2 \leq a \leq 4$. Thus, $\mathrm{E}[C] \leq N/4$. Hence, with probability at least $1/2$, at most $N/2$ elements are paired and so at least $N/2$ table entries are singletons. Assume this is the case. Since $|T| \leq 4N$, Pr[there is a unique element that maps to $\alpha] \geq 1/8$.

Overall, the probability that we pick (h_s, α) so that (C, h_s, α) has a unique witness is at least $1/(16r)$. This can be boosted in the usual way. We thus have an **RP** algorithm for an **NP**-complete problem, implying **RP** = **NP**. $\square$

Section 7: BPP $\subseteq \Sigma_2$

Σ_2 corresponds to languages $\mathcal{L}$ which can be written in the form:

$$x \in \mathcal{L} \iff \exists z, \quad \forall w, \quad q_{\mathcal{L}}(x, z, w) = 1$$

where $q_{\mathcal{L}}$ is a polynomial time predicate, and $\|z\|$ and $\|w\|$ are polynomial in $\|x\|$.

The following proofs are due to [33, Sipser]. Consider the following class $\mathbf{BPP}'$, which is a very strong form $\mathbf{BPP}$: For all $\mathcal{L} \in \mathbf{BPP}'$, when $x \in \mathcal{L}$ then $|W_x| > 2^{r-1}$ and when $x \notin \mathcal{L}$ then $|W_x| \leq 2^{(r-1)/2}$. We can determine if $x \in \mathcal{L}$ for $\mathcal{L} \in \mathbf{BPP}'$ as follows. Pick a $s \in_R \mathcal{S}$ where our hash family maps $\{0,1\}^r \rightarrow \{0,1\}^{r-1}$). For the case of $x \notin \mathcal{L}$, the table T which h_s maps to has $|T| \geq |W_x|^2$. We know from our previous analysis that in such a situation, $\Pr[h_s \text{ is 1-to-1}] \geq \frac{1}{2}$. When $x \in \mathcal{L}$, $\Pr[h_s \text{ is 1-to-1}] = 0$ (since the table is too small). Thus, we have a way of distinguishing the two situations. We can decide if $x \notin \mathcal{L}$ by the following:

$$x \notin \mathcal{L} \quad \iff \quad \begin{array}{l} \exists s \in \mathcal{S},\ \forall y, y' \in W_x \text{ such that } y \neq y', \\ \quad h_s(y) \neq h_s(y') \end{array}$$

This is a Σ_2 form of the complement of $\mathcal{L}$ (note that membership in W_x takes only polynomial time to check). Therefore, $\overline{\mathbf{BPP}'} \subseteq \Sigma_2$ and $\mathbf{BPP}' \subseteq \Pi_2$. We now present the result of [33, Sipser].

Theorem : $\mathbf{BPP} \subseteq \Sigma_2$.

PROOF: Consider the following version $\mathbf{BPP}''$ of $\mathbf{BPP}$: For all $\mathcal{L} \in \mathbf{BPP}''$, when $x \in \mathcal{L}$ then $|W_x| > \frac{1}{r}2^r$ and when $x \notin \mathcal{L}$ then $|W_x| \leq \frac{1}{2r^2}2^r$. Using the usual amplification method, a language $\mathcal{L} \in \mathbf{BPP}$ can be easily reduced to a language $\mathcal{L}' \in \mathbf{BPP}''$, and thus $\mathbf{BPP}''$ is equivalent to $\mathbf{BPP}$.

We now show how to determine if $x \in \mathcal{L}$, where $\mathcal{L} \in \mathbf{BPP}''$. We use a table of size $t = \frac{1}{r^2}2^r$. Pick $s_1, s_2, \ldots, s_r \in_R \mathcal{S}$ where our hash family maps $\{0,1\}^r \rightarrow [\frac{1}{r^2}2^r]$). Since we can't get a 1-to-1 mapping for one particular table, we consider for every witness in W_x whether there is at least one mapping which isolates the witness. We notice h isolates $y \in W_x$ iff for all $y' \in W_x$ such that $y' \neq y$, we have $h(y) \neq h(y')$.

Define an event A: for all $y \in W_x$, there is an $i \in \{1, \ldots, r\}$ such that for all $y' \in W_x$ such that $y' \neq y$, $h_{s_i}(y) \neq h_{s_i}(y')$. We show that

$$\begin{aligned} x \in \mathcal{L} &\Rightarrow \Pr[A] = 0, \\ x \notin \mathcal{L} &\Rightarrow \Pr[A] > 0. \end{aligned}$$

Each h_{s_i} can isolate at most t elements. Hence, if $x \in \mathcal{L}$, the number of witnesses that can be isolated is $\leq tr = \frac{1}{r}2^r < |W_x|$, and thus there must be some witnesses in W_x that are not isolated by any of the r hash functions, and thus $\Pr[A] = 0$.

What if $x \notin \mathcal{L}$?

- Fix i, y, y'. $\Pr[h_{s_i}(y) = h_{s_i}(y')] = \frac{1}{t}$.
- Fix i, y. $\Pr[\exists y' \in W_x, y' \neq y, h_{s_i}(y) = h_{s_i}(y')] \leq \frac{|W_x|}{t} \leq \frac{1}{2}$.
- Fix y. $\Pr[\forall i, \exists y' \in W_x, y' \neq y, h_{s_i}(y) = h_{s_i}(y')] \leq \frac{1}{2^r}$.
- $\Pr[A] = 1 - \Pr[\exists y \in W_x, \forall i \in \{1, \ldots, r\}, \exists y' \in W_x, y' \neq y, h_{s_i}(y) = h_{s_i}(y')] \geq 1 - \frac{|W_x|}{2^r} \geq 1 - \frac{1}{2r^2}$.
- Thus, $\Pr[A] > 0$.

Membership in language $\mathcal{L}$ can thus be summarized as:

$$x \notin \mathcal{L} \iff \begin{array}{c} \exists h_{s_1}, \ldots, h_{s_r}, \ \ \forall y \in W_x, \ \exists i \in \{1, \ldots, r\}, \\ \forall y' \in W_x, y' \neq y, \ \ h_{s_i}(y) \neq h_{s_i}(y') \end{array}$$

We've shown $\mathbf{BPP} = \mathbf{BPP}'' \subseteq \Sigma_4$. Notice that the third quantifier $\exists i \in \{1, \ldots, r\}$ is of polynomial size. It is possible to eliminate this quantifier, and then the surrounding pair of "for all" quantifiers collapse together, giving $\mathbf{BPP} \subseteq \Sigma_2$. $\square$

Section 8: AM = IP

[16, Goldwasser-Sipser] show that **AM** = **IP**, that is, public coins are as powerful as private coins in interactive protocols. To illustrate this we look at the graph non-isomorphism problem: GNI = $\{(G_0, G_1)$: graphs G_0 and G_1 are not isomorphic $\}$. To show that GNI $\in$ **IP**, we must exhibit a prover P and verifier V such that, for some pair of constants c_{yes} and c_{no} with $c_{\text{yes}} > c_{\text{no}}$:

(1) For all $(G_0, G_1) \in$ GNI, P causes V to accept with probability at least c_{yes}.

(2) For all $(G_0, G_1) \notin$ GNI, every prover causes V to accept with probability at most c_{no}.

The coins of V are kept private.

Let G_0 and G_1 both be graphs on m nodes. The **IP** protocol proceeds as follows. V picks a random permutation of m nodes $\sigma \in_R S_m$ and a random graph index $b \in_R \{0, 1\}$ and sends $\sigma(G_b)$ to P, where $\sigma(G_b)$ is the graph indexed by b with the nodes in the order specified by σ. P then computes a bit $c \in \{0, 1\}$, which is supposed to be the index of the graph sent by V, and sends c to V. V accepts iff $b = c$. If $(G_0, G_1) \in$ GNI, i.e., the graphs are not isomorphic, then P can tell which graph is sent by V and thus can compute c correctly, so that $\Pr[V \text{ accepts}] = 1 = c_{\text{yes}}$. If the two graphs are isomorphic then, since the verifier sends a random permutation of the graph, the distribution on graphs received by the prover is the same whether $b = 0$ or $b = 1$, and since b is chosen randomly the prover can answer correctly with probability $1/2$, and thus $\Pr[V \text{ accepts}] = 1/2 = c_{\text{no}}$.

Clearly, this protocol does not work if the coins used by the verifier to choose b and σ are public.

Now we look at an **AM** protocol for the same problem. Define U to be the set of all m vertex graphs. Define $W \subseteq U$ to be the set of all graphs that V could have sent in the previous protocol, so $W = \{\sigma(G_b) : \sigma \in S_m,\ b \in \{0, 1\}\}$. Now,

$$\begin{aligned} G_0 \sim G_1 \quad &\Rightarrow \quad |W| = m!, \\ G_0 \not\sim G_1 \quad &\Rightarrow \quad |W| = 2(m!). \end{aligned}$$

(This isn't always true, but something with a similar effect can be arranged), so the prover has to try to convince the verifier that the set W is big. This is done by mapping the elements of W into a table T of size $4(m!)$ and looking at the probability that a random entry in T is filled. The **AM** protocol proceeds as follows. V randomly chooses $s \in_R \mathcal{S}$ and $\alpha \in_R \{1, \ldots, T\}$, and send h_s and α to P. P computes $\sigma \in S_m$ and $c \in \{0,1\}$ and sends to V the graph $\sigma(G_c)$. V accepts iff $h_s(\sigma(G_c)) = \alpha$.

Note that unlike in the **IP** protocol described above, all the random bits used by V, i.e., h_s and α, are sent to P and are thus public. We want to show that if G_0 is not isomorphic to G_1, then there is a fairly decent chance that the prover P will be able to find a graph in W (equivalently, a permutation $\sigma \in S_m$ and $b \in \{0,1\}$) which is mapped to α by the hash function h. The following calculations apply to mapping a subset $W \subseteq U$ of size N into a table T of size $2N$ (we are interested in $N = 2(m!)$). Below we show that given an index $\alpha \in \{1, \ldots, 2N\}$, Pr[at least one element in the size N set is mapped to α] $\geq 3/8 = c_{\text{yes}}$. Define E_i to be the event that element i is mapped to the given α. Then by inclusion-exclusion and using the fact that the hash family is 2-universal, the above probability is:

$$\Pr[E_1 \cup \ldots \cup E_N] \geq \sum_{i=1}^{N} \Pr[E_i] - \sum_{i<j} \Pr[E_i \cap E_j] = \frac{N}{2N} - \binom{N}{2} \frac{1}{4N^2} \geq \frac{3}{8}$$

Thus, if $x \in \mathcal{L}$ then $\Pr[V \text{ accepts}] \geq 3/8$. If $x \notin \mathcal{L}$, then the subset W is $1/4$ the size of the table, so $\Pr[V \text{ accepts}] \leq 1/4 = c_{\text{no}}$. The gap between these probabilities can be boosted in the usual way.

Section 9: Deterministic Amplification

Let $\mathcal{L}$ be a **RP** language. A randomized **TM** can decide membership of $x \in \{0,1\}^n$ in $\mathcal{L}$ by choosing $y \in_R \{0,1\}^r$ and then checking if $y \in W_x$. As mentioned before, we can reduce the error probability of misclassification by choosing $y_1, \ldots, y_k \in_R \{0,1\}^r$, counting the number of these strings that fall in W_x and basing the decision on the value of this number. If the y_i's are chosen independently, we need kr random bits to achieve an error probability of $2^{-\mathcal{O}(k)}$. We'd like to use fewer random bits and still achieve a reduced error probability.

In this application, a pseudo-random generator is a deterministic **TM**, G, that takes a random seed s and produces "pseudo-random" bits $G(s) = y_1, \ldots, y_k$, where each string is of length r. The algorithm is simply to test all k of these strings and to conclude that $x \in \mathcal{L}$ if for any $i \in \{1, \ldots, k\}$ $y_i \in W_x$ and $x \notin \mathcal{L}$ otherwise. Notice there is misclassification only when $x \in \mathcal{L}$ and $\{y_1, \ldots, y_k\} \subseteq \overline{W_x}$. We'll give several constructions.

Generator	Random bits	Error
Chor-Goldreich	$\mathcal{O}(r)$	$\mathcal{O}(1/k)$
Impagliazzo-Zuckerman	$\mathcal{O}(r+k^2)$	$2^{-\mathcal{O}(k)}$
Nisan	$\mathcal{O}(r \lg k)$	$2^{-\mathcal{O}(k)}$
AKS	$r + \mathcal{O}(k)$	$2^{-\mathcal{O}(k)}$

For the Chor-Goldreich generator we show the result for $\mathcal{L} \in$ **BPP**. For all the other generators, we show the result for $\mathcal{L} \in$ **RP**. For these results, we assume $c_{\text{yes}} \geq 1/2$. Thus, the probability of a misclassification when $x \in \mathcal{L}$ is $< 1/2$. All of these proofs can be extended to show an analogous result for $\mathcal{L} \in$ **BPP** using exactly the same generator.

The results of the following exercise are due to [1, Adleman] and [6, Bennett-Gill]. These results show there is a polynomial size sample space that can be used to classify all $x \in \{0,1\}^n$ as either being in $\mathcal{L}$ or not, where $\mathcal{L} \in$ **RP** or $\mathcal{L} \in$ **BPP**. The crucial property lacking from these results is that the sample space is not *efficiently constructible*. This property is the main point of the deterministic amplification constructions given in the following sections that reduce the number of random bits needed to find a witness with high probability.

Definition (P/poly): We say that $\mathcal{L} \in$ **P/poly** if there is a polynomial time **TM** $M(x, y)$ such that when $\|x\| = n$ then $\|y\| = r$, where r is polynomial in n, with the following property: For each positive integer n, there is an "advice string" $y \in \{0, 1\}^r$ with the property that, for all $x \in \{0, 1\}^n$,

$$\begin{aligned} x \in \mathcal{L} &\Rightarrow M(x, y) = 1, \\ x \notin \mathcal{L} &\Rightarrow M(x, y) = 0. \end{aligned}$$

We use the term "advice string" because, given the value of the advice string $y \in \{0, 1\}^r$, it is easy to decide membership in $\mathcal{L}$ for all $x \in \{0, 1\}^n$. Note that if it is possible to compute the value of the advice string $y \in \{0, 1\}^r$ in $n^{\mathcal{O}(1)}$ time, then $\mathcal{L} \in P$. However, in general it may not be possible to compute the advice string in $n^{\mathcal{O}(1)}$ time. One way of thinking about a language $\mathcal{L} \in$ **P/poly** is that membership in $\mathcal{L}$ can be decided in $n^{\mathcal{O}(1)}$ time with the aid of a polynomial amount of extra advice for each input length.

Exercise 1 : Prove that **RP** $\subseteq$ **P/poly** and **BPP** $\subseteq$ **P/poly**.

Section 10: The Chor-Goldreich Generator

We first describe the generator due to [9, Chor-Goldreich]. We show how this generator works for $\mathcal{L} \in \mathbf{BPP}$. We assume that $c_{\text{yes}} \geq 3/4$ and that $c_{\text{no}} \leq 1/4$. Let $\mathcal{S}$ be the index set for the hash family mapping $\{0,1\}^r \rightarrow \{0,1\}^r$, and let $s \in_R \mathcal{S}$. We let $G(s) = h_s(1), \ldots, h_s(k)$; i.e., $y_i = h_s(i)$. Then the y_i's are uniformly distributed and pairwise independent. The algorithm concludes that $x \in \mathcal{L}$ if at least $k/2$ of the strings $y_1, \ldots, y_k$ are in W_x and $x \notin \mathcal{L}$ otherwise.

Theorem : The probability of misclassifying $x \in \{0,1\}^n$ with respect to membership in $\mathcal{L}$ is at most $4/k$.

PROOF: Define

$$Z_i = \begin{cases} 1 & \text{if } y_i \in W_x \\ 0 & \text{otherwise} \end{cases}$$

The Z_i's are also identically distributed and pairwise independent with mean $\mu = \mu(W_x)$ and variance $\sigma^2 = \mu(1-\mu) \leq 1/4$. Since the variance of $\sum_{i=1}^{k} Z_i$ is $k\sigma^2$, it follows (using the Chebyshev inequality) that

$$\Pr\left[\left|\sum_{i=1}^{k} Z_i - \mu k\right| > \frac{k}{4}\right] \leq \frac{16k\sigma^2}{k^2} \leq \frac{4}{k}.$$

$\square$

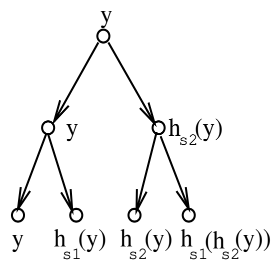
Section 11: Nisan's Generator

We now describe the generator of [28, Nisan]. Let $\ell = \log k$. Let $\mathcal{S}$ be the index set for the hash family mapping $\{0,1\}^r \rightarrow \{0,1\}^r$, and let $s_1, \ldots, s_\ell \in_R \mathcal{S}$. Let $G_0(y) = y$ and define inductively for $i \geq 1$,

$$G_{i+1}(s_1, \ldots, s_{i+1}, y) = \langle G_i(s_1, \ldots, s_i, y), G_i(s_1, \ldots, s_i, h_{s_{i+1}}(y)) \rangle,$$

where $\langle a, b \rangle$ denotes the concatenation of strings a and b. For example, $G_2(s_1, s_2, y) = \langle y, h_{s_1}(y), h_{s_2}(y), h_{s_1}(h_{s_2}(y)) \rangle$. A more obvious way to visualize this generator is with a complete binary tree as shown in the following figure. A hash function is assigned to each level of this tree. The root of the tree is assigned the seed value y, and for any node w on level $i \geq 1$ is assigned value v_w, where

$$v_w = \begin{cases} v_{parent(w)} & \text{if } w \text{ is the left child of parent(w)} \\ h_{s_{\ell-i+1}}(v_{parent(w)}) & \text{otherwise} \end{cases}$$



Notice $G_\ell(s_1, \ldots, s_\ell, y)$ is simply the concatenation of the strings on level ℓ of this tree.

Before proving this generator works we need the following technical lemma, which is also of independent interest.

Hash Mixing Lemma : Let $\epsilon = 2^{-r/3}$ for some fixed parameter r. Then for all $A, B \subseteq \{0,1\}^r$, and for all but an ϵ fraction of $s \in \mathcal{S}$,

$$|\Pr_{y \in_R \{0,1\}^r}[y \in A, h_s(y) \in B] - \Pr_{y,z \in_R \{0,1\}^r}[y \in A, z \in B]| \leq \epsilon$$

PROOF: We want to bound the number of $s \in \mathcal{S}$ such that

$$|\Pr_{y \in_R \{0,1\}^r}[y \in A, h_s(y) \in B] - \mu(A)\mu(B)| \geq \epsilon$$

This is exactly the number of $s \in \mathcal{S}$ such that

$$|\Pr_{y \in_R A}[h_s(y) \in B] - \mu(B)| \geq \epsilon/\mu(A)$$

Define the indicator random variable

$$Z_y^{h_s} = \begin{cases} 1 & \text{if } h_s(y) \in B \\ 0 & \text{otherwise} \end{cases}$$

A hash function h_s is "bad" if

$$\left| \sum_{y \in A} Z_y^{h_s} - |A|\mu(B) \right| \geq \frac{\epsilon |A|}{\mu(A)} = \epsilon 2^r$$

By Chebyshev,

$$\Pr_{s \in_R \mathcal{S}} \left[\left| \sum_{y \in A} Z_y^{h_s} - |A|\mu(B) \right| \geq \epsilon 2^r \right] \leq \frac{\mu(B)|A|}{\epsilon^2 2^{2r}} < \epsilon$$

$\square$

Theorem : If $x \in \mathcal{L}$ then $\Pr[G_\ell(s_1, \ldots, s_\ell, y) \subseteq \overline{W}_x] \leq \mu(\overline{W}_x)^{2^\ell} + (\ell + 2)\epsilon$ where $\epsilon = 2^{-r/3}$.

PROOF: The $\epsilon\ell$ in the error term handles the hash functions that are "bad" for the Hash Mixing Lemma. Assume, for the moment, that h_s for all $s \in \mathcal{S}$ satisfy the Hash Mixing Lemma. We show that

$$\Pr_{y \in_R \{0,1\}^r} [G(s_1, \ldots, s_\ell, y) \subseteq \overline{W}_x] \leq \mu(\overline{W}_x)^{2^\ell} + 2\epsilon$$

Inductively assume it is true for $\ell - 1$.

Let $A = B = \{y : G(s_1, \ldots, s_{\ell-1}, y) \subseteq \overline{W}_x\}$. These are the "bad" y, i.e., those y for which we decide that $x \notin \mathcal{L}$ when in fact $x \in \mathcal{L}$. Now using the Hash Mixing Lemma,

$$\begin{aligned} \Pr[G(s_1, \ldots, s_\ell, y) \subseteq \overline{W}_x] &\leq \Pr[G(s_1, \ldots, s_{\ell-1}, y) \subseteq \overline{W}_x]^2 + \epsilon \\ &\leq (\mu(\overline{W}_x)^{2^{\ell-1}} + 2\epsilon)^2 + \epsilon \\ &\leq \mu(\overline{W}_x)^{2^\ell} + 2\epsilon, \end{aligned}$$

where the first inequality holds by the Hash Mixing Lemma, and the second inequality holds by the induction hypothesis.

Each s_i, $i \in \{1, \ldots, \ell\}$, had an ϵ chance of being bad for the Hash Mixing Lemma, and so

$$\Pr_{y \in_R \{0,1\}^r}[G(s_1, \ldots, s_\ell, x) \subseteq \overline{W}_x] \leq \mu(\overline{W}_x)^{2^\ell} + \ell\epsilon + 2\epsilon.$$

$\square$

Section 12: The Impagliazzo-Zuckerman Generator

Let ℓ and k be integer parameters. (A good setting is to make $k \approx \ell \approx \sqrt{r}$). The generator described in [19, Impagliazzo-Zuckerman] produces $k+1$ potential witnesses, each of length r, from only $3r + k\ell$ random bits. Let $\mathcal{S}$ be the index set for the hash family mapping $\{0,1\}^r \rightarrow \{0,1\}^{r-\ell}$. The generator is defined by a function $G : \{0,1\}^{2r} \times \{0,1\}^r \times \{\{0,1\}^\ell\}^k \longrightarrow \{\{0,1\}^r\}^{k+1}$:

$$G(s, Y_1, Z_1, \ldots, Z_k) \longmapsto (Y_1, Y_2, \ldots, Y_{k+1})$$

where $s \in_R \mathcal{S}$, $Y_1 \in_R \{0,1\}^r$, and $Z_i \in_R \{0,1\}^\ell$, for $1 \leq i \leq k$. The Y_i's are defined by:

$$Y_{i+1} = \langle h_s(Y_i), Z_i \rangle, \; i = 1, \ldots k$$

Theorem : If $x \in \mathcal{L}$ then,

$$\Pr[G(s, Y_1, Z_1, \ldots, Z_k) \subseteq \overline{W}_x] \leq \mu(\overline{W}_x)^{k+1} + 2^{1-\ell/2}$$

Several definitions and lemmas are needed to prove the theorem. Let P and Q be two probability distributions on a set A. The L_1-distance $\|\cdot\|_1$ and the L_2-distance $\|\cdot\|_2$ between P and Q are defined as

$$\|P - Q\|_1 = \sum_{i \in A} |P_i - Q_i|$$

and

$$\|P - Q\|_2 = \left(\sum_{i \in A} (P_i - Q_i)^2 \right)^{\frac{1}{2}}.$$

If Π denotes the uniform distribution on a set A then a distribution P on the set A is called *ϵ-uniform* or *ϵ-quasi-random* if $\|P - \Pi\|_1 \leq \epsilon$.

The *collision probability* $c(P)$ of a probability distribution on a set A is defined as

$$c(P) = \Pr_{i,j \in_P A}[i = j] = \sum_{i \in A} P_i^2$$

The next lemma states a simple condition for when a probability distribution is ϵ-uniform.

Lemma 1 : If $c(P) \leq (1+\epsilon^2)/|A|$ then P is ϵ-uniform.

PROOF: By the Cauchy-Schwartz inequality, if $v \in \mathbf{R}^n$ then $\|v\|_1 \leq \sqrt{n}\|v\|_2$. Applying this to $P - \Pi$ yields

$$\|P-\Pi\|_1^2 \leq |A| \sum_{i \in A} (\Pi_i - P_i)^2 = |A| \left(\sum_{i \in A} \Pi_i^2 - 2 \sum_{i \in A} \Pi_i P_i + \sum_{i \in A} P_i^2 \right).$$

Since Π is the uniform distribution $\sum_{i \in A} \Pi_i P_i = 1/|A|$ and $\sum_{i \in A} \Pi_i^2 = 1/|A|$. By assumption $\sum_{i \in A} P_i^2 = c(P) \leq (1+\epsilon^2)/|A|$. □

Lemma 2 : Let $\mathcal{S}$ be the index set of the hash family that maps U to T. Let P be the distribution $\langle s, h_s(x) \rangle$, where $s \in_R \mathcal{S}$ and $x \in_R W \subseteq U$ and let $A = \mathcal{S} \times T$. Then,

$$c(P) = \frac{1 + \frac{|T|}{|W|}}{|A|}.$$

PROOF: Let $s, s' \in_R \mathcal{S}$ and $x, x' \in_R W$. Then,

$$\begin{aligned}
c(P) &= \Pr[\langle s, h_s(x) \rangle = \langle s', h_{s'}(x') \rangle] \\
&= \Pr[s = s'] \Pr[h_s(x) = h_{s'}(x') | s = s'] \\
&= \Pr[s = s'] \Pr[h_s(x) = h_s(x')] \\
&= \Pr[s = s'] \left(\Pr[x = x'] + \Pr[h_s(x) = h_{s'}(x') | x \neq x'] \right) \\
&= \frac{1}{\mathcal{S}} \left(\frac{1}{|W|} + \frac{1}{|T|} \right) \\
&= \frac{1}{|\mathcal{S}||T|} \left(1 + \frac{|T|}{|W|} \right)
\end{aligned}$$

□

The following lemma is from [20, Impagliazzo-Levin-Luby].

Leftover-Hash-Lemma : Let $\mathcal{S}$ be the index set of the hash family that maps U to T. For $s \in_R \mathcal{S}$ and $x \in_R W \subseteq U$, the distribution $\langle s, h_s(x) \rangle$ is ϵ-uniform, where $\epsilon = \sqrt{|T|/|W|}$.

PROOF: Apply Lemma 2 and Lemma 1 in sequence. □

Proof of Theorem : The proof is by induction on k for a fixed value of ℓ. Let $\text{error}_k = \Pr[G(s, Y_1, Z_1, \ldots, Z_k) \subseteq \overline{W}_x]$ be the error probability with respect to k. It is clear that $\text{error}_0 \leq \mu(\overline{W}_x)$. For $k \geq 1$,

$$\begin{aligned}\text{error}_k &= \Pr[Y_1 \in \overline{W}_x] \Pr[G(s, Y_2, Z_2, \ldots, Z_k) \subseteq \overline{W}_x | Y_1 \in \overline{W}_x] \\ &= \mu(\overline{W}_x) \Pr[G(s, Y_2, Z_2, \ldots, Z_k) \subseteq \overline{W}_x | Y_1 \in \overline{W}_x],\end{aligned}$$

where $Y_2 = \langle h_s(Y_1), Z_1 \rangle$. Let $\epsilon = 1/\sqrt{2^\ell \mu(\overline{W}_x)}$. Then, it follows that

$$\begin{aligned}&\Pr[G(s, Y_2, Z_2, \ldots, Z_k) \subseteq \overline{W}_x | Y_1 \in \overline{W}_x] \\ &\quad \leq \Pr[G(s, \hat{Y}_2, Z_2, \ldots, Z_k) \subseteq \overline{W}_x] + \epsilon,\end{aligned}$$

where $\hat{Y}_2 \in_R \{0,1\}^r$. This is because the distribution $\langle s, h_s(Y_1) \rangle$ is ϵ-uniform by the Leftover-Hash-Lemma, where $s \in_R \mathcal{S}$ and $Y_1 \in_R \overline{W}_x$, and thus

$$\| \langle s, \langle h_s(Y_1), Z_1 \rangle, Z_2, \ldots, Z_k \rangle, \langle s, \hat{Y}_2, Z_2, \ldots, Z_k \rangle \|_1 \leq \epsilon,$$

and this implies that the behavior of G on these two distributions can differ by at most ϵ.

The induction hypothesis implies that

$$\Pr[G(s, \hat{Y}_2, Z_2, \ldots, Z_k) \subseteq \overline{W}_x] \leq \mu(\overline{W}_x)^k + 2^{1-\ell/2}.$$

This implies

$$\begin{aligned}\text{error}_k &\leq \mu(\overline{W}_x) \left(\mu(\overline{W}_x)^k + 2^{1-\ell/2} + 1/\sqrt{2^\ell \mu(\overline{W}_x)} \right) \\ &\leq \mu(\overline{W}_x)^{k+1} + 2^{1-\ell/2},\end{aligned}$$

where the last inequality uses $\mu(\overline{W}_x) \leq 1/2$ and $\sqrt{\mu(\overline{W}_x)} \leq 1$.

$\square$

Section 13: The Expander Mixing Lemma

Let $G = (U, E)$ be a d-regular undirected graph with n nodes ($|U| = n$). The adjacency matrix of G is a symmetric $n \times n$-matrix M with

$$M(i,j) = \begin{cases} 0 & (i,j) \notin E \\ 1 & (i,j) \in E \end{cases}.$$

Every such matrix has an orthonormal basis of eigenvectors. Let these eigenvectors for M be the vectors $\mathbf{r}_0, \ldots, \mathbf{r}_{n-1} \in \mathbf{R}^n$ with the corresponding eigenvalues $\lambda_0, \ldots, \lambda_{n-1} \in \mathbf{R}$. Define $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$. We let $\cdot$ denote multiplication of matrices over the reals. Whenever a vector is involved in a multiplication, we use the convention that it is a row vector if it is to the left of $\cdot$ and a column vector if it is to the right of $\cdot$. Thus, if a and b are equal length vectors then $a \cdot b$ denotes the inner product of a and b over the reals. We have for all $0 \leq i, j \leq n-1$:

$$\begin{aligned} \mathbf{r}_i \cdot \mathbf{r}_j &= \delta_{i,j} \\ M \cdot \mathbf{r}_i &= \lambda_i \, \mathbf{r}_i \end{aligned}$$

Every row of M consists of d ones and $n - d$ zeros. Hence the vector of ones in all components (denoted $\bar{1} \in \mathbf{R}^n$) is an eigenvector of M corresponding to the eigenvalue d. Furthermore all eigenvalues are real-valued and no larger than d. Without loss of generality, we assume $\mathbf{r}_0 = \bar{1}/\sqrt{n}$ and $\lambda_0 = d$. Let

$$\lambda = \max_{1 \leq i \leq n-1} |\lambda_i|$$

denote the second largest eigenvalue, that is the maximum factor by which a vector orthogonal to $\mathbf{r}_0$ is stretched when multiplied by M.

Multiplication of a vector $\mathbf{z} \in \mathbf{R}^n$ with M can easily be expressed using the eigenvectors and eigenvalues: Setting $\gamma_i = \mathbf{z} \cdot \mathbf{r}_i$ we have

$$\begin{aligned} \mathbf{z} &= \sum_{0 \leq i \leq n-1} \gamma_i \mathbf{r}_i \\ \text{and} \quad M \cdot \mathbf{z} &= \sum_{0 \leq i \leq n-1} \lambda_i \gamma_i \mathbf{r}_i. \end{aligned}$$

For two sets $A, B \subseteq U$ denote the set of (directed) edges from A to B in G by $E(A, B) = \{(v, w) \in U^2 : (v, w) \in E\}$. Fixing A and an integer b, and picking a set B of size b uniformly at random, the expected size of $E(A,B)$ is $\frac{d|A|b}{n}$. The following lemma states that for *any* set B the size of $E(A,B)$ is close to its expectation, where "close" depends on the value of λ.

Expander Mixing Lemma : For all $A, B \subseteq U$ it holds that

$$\left| |E(A,B)| - \frac{d|A||B|}{n} \right| \leq \lambda \sqrt{|A||B|} \leq \lambda\, n.$$

PROOF: Let $\chi_A \in \mathbf{R}^n$ denote the indicator vector of A, *i.e.* a "one" is at the position corresponding to a vertex $v \in A$ and a "zero" for $v \notin A$. χ_B is the corresponding vector for B. Set

$$\begin{aligned} \alpha_i &= \chi_A \cdot \mathbf{r}_i \\ \beta_i &= \chi_B \cdot \mathbf{r}_i \end{aligned}$$

Then $\alpha_0 = |A|/\sqrt{n}$ and $\beta_0 = |B|/\sqrt{n}$, and we have

$$\begin{aligned} |E(A,B)| &= \sum_{i \in A,\, j \in B} M(i,j) \\ &= \chi_A \cdot M \cdot \chi_B \\ &= \left(\sum_{0 \leq i \leq n-1} \alpha_i\, \mathbf{r}_i \right) \cdot \left(\sum_{0 \leq j \leq n-1} \lambda_j\, \beta_j\, \mathbf{r}_j \right) \\ &= \sum_{0 \leq i \leq n-1} \lambda_i\, \alpha_i\, \beta_i \\ &= \frac{d|A||B|}{n} + \sum_{1 \leq i \leq n-1} \lambda_i\, \alpha_i\, \beta_i \\ \Longrightarrow \quad \left| |E(A,B)| - \frac{d|A||B|}{n} \right| &\leq \lambda \sum_{0 \leq i \leq n-1} |\alpha_i\, \beta_i| \\ &\leq \lambda\, ||\alpha||_2\, ||\beta||_2 \\ &= \lambda\, ||\chi_A||_2\, ||\chi_B||_2 \\ &= \lambda \sqrt{|A||B|} \end{aligned}$$

Another way to state the Expander Mixing Lemma is that for all $A, B \subseteq U$,

$$|\Pr_{x,s}[x \in A, e_s(x) \in B] - \Pr_{x,y}[x \in A, y \in B]| \leq \frac{\lambda}{d},$$

where in the first random experiment x is a uniformly chosen node, $s \in_R \{1, \ldots, d\}$, and $e_s(x)$ is the neighbor of x indexed by s, whereas in the second experiment x and y are two independently and uniformly chosen nodes. Note the resemblance with the Hash Mixing Lemma, where $e_s(x)$ is substituted by $h_s(x)$.

There are explicit constructions of symmetric matrices with a small second largest eigenvalue λ, corresponding to graphs with good expansion properties. For all integers n' and d' there is an explicit construction of an n node d-regular graph G with $n' \leq n \leq 2n'$, $d' \leq d \leq 2d'$ and $\lambda \leq d^{9/10}$. (For example, see either [25, Lubotzky-Phillips-Sarnak] or [27, Margulis].) For every node $x \in U$ and integer $s \in \{1, \ldots, d\}$ the s-th neighbor e_s of x in G can be computed in logarithmic space. (To simplify the presentation we assume in the sequel that we can construct expanders for all values of n and d.)

Section 14: The Karp-Pippenger-Sipser Generator

The [23, Karp-Pippenger-Sipser] generator uses the explicit construction of expanders: The set $\{0,1\}^r$ is identified with the nodes of a 2^r node k-regular expander with $\lambda \leq k^{9/10}$. The seed to the generator G is a string $z \in_R \{0,1\}^r$, and $G(z)$ produces $y_1, \ldots, y_k$, which are the k neighbors of z in the expander graph. Thus, this scheme uses exactly r random bits.

Theorem : If $x \in \mathcal{L}$ then

$$\Pr[\{y_1, \ldots, y_k\} \subseteq \overline{W}_x] \leq 2k^{-1/10}.$$

PROOF: Let $A \subseteq \{0,1\}^r$ be the set of nodes z with the property that all neighbors of z are in $\overline{W}_x$. Thus

$$\Pr[\{y_1, \ldots, y_k\} \subseteq \overline{W}_x] = |A|/2^r.$$

From $E(A, W_x) = \emptyset$ and the Expander Mixing Lemma it follows that

$$\begin{aligned} \frac{k|A||W_x|}{2^r} &\leq \lambda 2^r \\ \Longrightarrow \quad \frac{|A|}{2^r} &\leq \frac{\lambda 2^r}{k|W_x|} \\ &\leq 2k^{-1/10} \qquad (\text{since } |W_x| \geq 2^{r-1}) \end{aligned}$$

□

Section 15: The Ajtai-Komlós-Szemerédi Generator

[3, Ajtai-Komlós-Szemerédi] show how to simulate a randomized log-space computation using $O\left(\frac{(\log n)^2}{\log\log n}\right)$ random bits by a deterministic log-space computation. [10, Cohen-Wigderson] and [31, Nisan-Zuckerman] observe that the AKS generator can also be used for amplification: Let $n = 2^r$ and identify the set $\{0,1\}^r$ with the nodes of a d-regular n-node expander graph G. Set d to some constant value so that $\lambda \leq d/4$ can be achieved. Choose the nodes $y_1, \ldots, y_k$ as the nodes visited on a random walk of length k starting at a random node $z \in_R \{0,1\}^r$. The random walk is determined by the starting point z and integers $i_j \in \{1, \ldots, d\}$ for $j \in \{1, \ldots, k\}$ describing which edge to use in j-th step of the walk. Thus, y_1 is the i_1-th neighbor of z and, for $j \geq 2$, y_j is the i_j-th neighbor of y_{j-1}. The number of random bits used is $r + k\log d = r + \mathcal{O}(k)$.

Theorem : If $x \in \mathcal{L}$ then

$$\Pr[\{y_1, \ldots, y_k\} \subseteq \overline{W}_x] = 2^{-\Theta(k)}.$$

PROOF: To bound the error probability we describe the probability distribution after subsequent steps of the random walk by an n-dimensional vector. Let $\mathbf{p}_0$ be the vector describing the initial distribution, i.e., the distribution of z, which is $\mathbf{p}_0(v) = 1/n$ for all $v \in \{0,1\}^r$. Let M be the adjacency matrix of G and set $\hat{M} = M/d$. Thus, the distribution after the first step, i.e., the distribution of y_1, is $\mathbf{p}_1 = \hat{M} \cdot \mathbf{p}_0$. We are interested in the probability that all nodes $y_1, \ldots, y_k$ are contained in $\overline{W}_x$. The probability of $y_1 \in \overline{W}_x$ is obtained by cancelling out the components of $\mathbf{p}_1$ corresponding to nodes in W_x and summing up the other components. Let $P_{\overline{W}_x}$ be the diagonal matrix with ones in the positions corresponding to elements of $\overline{W}_x$ and zeros elsewhere. Then

$$\Pr[y_1 \in \overline{W}_x] = ||P_{\overline{W}_x} \cdot \hat{M} \cdot \mathbf{p}_0||_1$$

Continuing this process yields

$$\Pr[y_1 \in \overline{W}_x \wedge \ldots \wedge y_k \in \overline{W}_x] = ||\left(P_{\overline{W}_x} \cdot \hat{M}\right)^k \cdot \mathbf{p}_0||_1. \qquad (1)$$

For any vector $\mathbf{z} = \sum_{0 \leq i \leq n-1} \gamma_i \mathbf{r}_i$ we have

$$\begin{aligned} ||P_{\overline{W}_x} \cdot \hat{M} \cdot \mathbf{z}||_2 &\leq ||P_{\overline{W}_x} \cdot \hat{M} \cdot \gamma_0 \mathbf{r}_0||_2 \\ &+ ||P_{\overline{W}_x} \cdot \hat{M} \cdot \sum_{1 \leq i \leq n-1} \gamma_i \mathbf{r}_i||_2 \end{aligned}$$

Using $\hat{M} \cdot r_0 = r_0$ and replacing $P_{\overline{W}_x}$ by I, we continue the inequalities as follows:

$$\begin{aligned}
&\leq \quad \|P_{\overline{W}_x} \cdot \gamma_0 \, \mathbf{r}_0\|_2 + \|\hat{M} \cdot \sum_{1 \leq i \leq n-1} \gamma_i \, \mathbf{r}_i\|_2 \\
&\leq \quad \sqrt{\mu(\overline{W}_x)} \, \|\gamma_0 \, \mathbf{r}_0\|_2 + \| \sum_{1 \leq i \leq n-1} \frac{\lambda_i}{d} \gamma_i \, \mathbf{r}_i\|_2 \\
&\leq \quad \sqrt{\mu(\overline{W}_x)} \, \|\gamma_0 \, \mathbf{r}_0\|_2 + \frac{\lambda}{d} \| \sum_{1 \leq i \leq n-1} \gamma_i \, \mathbf{r}_i\|_2 \\
&\leq \quad \left(\sqrt{\mu(\overline{W}_x)} + \frac{\lambda}{d} \right) \|\mathbf{z}\|_2
\end{aligned}$$

(The last inequality is based on the fact that both $\gamma_0 \, \mathbf{r}_0$ and $\sum_{1 \leq i \leq n-1} \gamma_i \, \mathbf{r}_i$ are both projections of z.) Applying this inequality to (1) and using Cauchy-Schwarz we are able to bound the error probability:

$$\begin{aligned}
\Pr[y_1 \in \overline{W}_x \wedge \ldots \wedge y_k \in \overline{W}_x] \quad &\leq \quad \sqrt{n} \, \| \left(P_{\overline{W}_x} \cdot \hat{M} \right)^k \cdot \mathbf{p}_0\|_2 \\
&\leq \quad \sqrt{n} \left(\sqrt{\mu(\overline{W}_x)} + \frac{\lambda}{d} \right)^k \|\mathbf{p}_0\|_2 \\
&\leq \quad \sqrt{n} \left(\sqrt{1/2} + 1/4 \right)^k 1/\sqrt{n} \\
&= \quad 2^{-\Theta(k)}.
\end{aligned}$$

$\square$

Section 16: Limited Independence Probability Spaces

We describe constructions of probability spaces that induce limited independence on a sequence of random variables. These are extensions of constructions described in Section 2.

Modulo Prime Space : Let p be a prime number. The sample space is the set of all pairs $\mathcal{S} = \{(a,b) : a,b \in \mathcal{Z}_p\}$, where $\mathcal{Z}_p = \{0,\ldots,p-1\}$. The distribution on the sample points is uniform, i.e., $(a,b) \in_R \mathcal{S}$. Let ζ be an indeterminate and consider the polynomial

$$p_{a,b}(\zeta) = (a\zeta + b) \bmod p,$$

where $(a,b) \in \mathcal{S}$. For all $i \in \mathcal{Z}_p$, define random variable

$$X_i(a,b) = p_{a,b}(i).$$

For brevity, we sometimes use X_i in place of $X_i(a,b)$.

Claim : $X_0,\ldots,X_{p-1}$ are uniformly distributed in $\mathcal{Z}_p$ and pairwise independent.

PROOF: For any pair $i,j \in \mathcal{Z}_p$, $i \neq j$, and for any pair of values $\alpha,\beta \in \mathcal{Z}_p$, there is a unique solution $a,b \in \mathcal{Z}_p$ to the pair of equations:

- $p_{a,b}(i) = \alpha$.
- $p_{a,b}(j) = \beta$.

Thus, $\Pr_{A,B}[X_i(A,B) = \alpha \wedge X_j(A,B) = \beta] = 1/p^2$. ⊔⊓

Recall the definition in Section 1 of pairwise independence. The following is a generalization of this definition.

Definition (k-wise independence): Let $X_1,\ldots,X_m$ be a sequence of random variables with values in a set N. We say the random variables are k-wise independent if, for all $1 \le i_1 < \cdots < i_k \le m$ and for all $\alpha_1,\ldots,\alpha_k \in N$,

$$\Pr[X_{i_1} = \alpha_1 \wedge \cdots \wedge X_{i_k} = \alpha_k] = \Pr[X_{i_1} = \alpha_1] \cdots \Pr[X_{i_k} = \alpha_k].$$

Exercise 2 : Let p be a prime number and let $m \leq p$. Generalize the Modulo Prime Space to a probability space where $X_0, \ldots, X_{m-1} \in_R \mathcal{Z}_p$ are k-wise independent, where the size of the probability space is p^k.

The Modulo Prime Space can be generalized as follows. The following construction is a more detailed description of the one presented in Section 2.

Linear Polynomial Space : Let $\mathcal{F}$ be any finite field and consider the polynomial

$$p_{a,b}(\zeta) = a\zeta + b$$

over $\mathcal{F}$, where $a, b \in \mathcal{F}$. Identify the integers $\{0, \ldots, |\mathcal{F}| - 1\}$ with the elements of $\mathcal{F}$. The sample space is $\mathcal{S} = \{(a, b) : a, b \in \mathcal{F}\}$ and the distribution on $\mathcal{S}$ is $(a, b) \in_R \mathcal{S}$. For all $i \in \mathcal{F}$, define random variable

$$X_i(a, b) = p_{a,b}(i),$$

where i on the left side of the equality is treated as an index and on the right side of the equality it is the corresponding element of $\mathcal{F}$.

The random variables $X_0, \ldots, X_{|\mathcal{F}|-1}$ are uniformly distributed in $\mathcal{F}$ and pairwise independent. A field with nice properties is $\mathrm{GF}[2^n]$, the Galois field with 2^n elements.

Mapping between $\{0,1\}^n$ and $\mathrm{GF}[2^n]$: There is a natural mapping between $\{0,1\}^n$ and polynomials in one variable ζ of degree $n-1$ over $\mathrm{GF}[2]$. Namely, if $a \in \{0,1\}^n$ and $\langle a_0, \ldots, a_{n-1} \rangle$ are the bits of a then the corresponding polynomial is

$$a(\zeta) = \sum_{i=0}^{n-1} a_i \zeta^i.$$

These polynomials are the field elements of $\mathrm{GF}[2^n]$. Let $a \in \{0,1\}^n$ and $b \in \{0,1\}^n$ and let $a(\zeta)$ and $b(\zeta)$ be the corresponding polynomials. Computing $a+b$ over $\mathrm{GF}[2^n]$ consists of computing $a \oplus b$, where $\oplus$ is vector addition over $\mathrm{GF}[2]$. Equivalently, computing $a+b$ over $\mathrm{GF}[2^n]$ consists of computing $a(\zeta) + b(\zeta)$ over $\mathrm{GF}[2]$, i.e., for all $i \in \{0, \ldots, n-1\}$, the i^{th} coefficient of $a(\zeta) + b(\zeta)$ is $a_i \oplus b_i$. Computing $a \cdot b$ over $\mathrm{GF}[2^n]$ consists of computing $a(\zeta) \cdot b(\zeta) \bmod r(\zeta)$, where $a(\zeta) \cdot b(\zeta)$ is polynomial multiplication over $\mathrm{GF}[2]$ that results in a polynomial of degree $2n-2$, and $r(\zeta)$ is a fixed irreducible polynomial of degree n. The zero element of $\mathrm{GF}[2^n]$ is the identically zero polynomial with coefficients $a_i = 0$ for all $i \in \{0, \ldots, n-1\}$, and the identity element is the polynomial with coefficients $a_0 = 1$ and $a_i = 0$ for all $i \in \{1, \ldots, n-1\}$.

In the Modulo Prime Space, $X_0, \ldots, X_{p-1}$ are pairwise independent and the size of the space is p^2. We describe a way to construct a pairwise independent probability space for $\{0,1\}$-valued random variables that has size linear in the number of random variables.

Inner Product Space : Let ℓ be a positive integer. The sample space is $\mathcal{S} = \{0,1\}^\ell$ and the distribution on sample points is $a \in_R \mathcal{S}$. For all $i \in \{0,1\}^\ell \setminus \{0^\ell\}$, define random variable

$$X_i(a) = a \odot i = \left(\sum_{j=1}^{\ell} a_j \cdot i_j \right) \bmod 2.$$

(We use $\odot$ to denote multiplication of matrices over GF[2], where we are using the convention that a vector to the left of $\odot$ is considered a row vector and a vector to the right of $\odot$ is viewed as a column vector.)

Claim : $X_1, \ldots, X_{2^\ell - 1}$ are uniformly distributed and pairwise independent.

Exercise 3 : Prove the pairwise independence property for the Inner Product Space.

Exercise 4 : Let p be a positive integer and let $X_1, \ldots, X_n \in_R \mathcal{Z}_p$ be a sequence of four-wise independent random variables. Define random variable

$$Y = \min\{(X_i - X_j) \bmod p : 1 \le i < j \le n\}.$$

Prove there is a constant $c > 0$ such that for any $\alpha \le 1$

$$\Pr[Y \le \alpha p / n^2] \ge c\alpha.$$

Hint : Let N be the set of $n(n-1)/2$ unordered pairs $\{(i,j) : 1 \le i < j \le n\}$. For fixed α, consider the sequence of $\{0,1\}$-valued random variables $\{Z_e : e \in N\}$, where if $e = (i,j)$ then $Z_e = 1$ if $|X_i - X_j| \le \alpha p/n^2$ and $Z_e = 0$ otherwise. Using the first two terms of the inclusion-exclusion formula, show that for any α,

$$\Pr[\exists e \in N : Z_e = 1] \ge \sum_{e \in N} \Pr[Z_e = 1] - \sum_{e,e' \in N, e \ne e'} \Pr[Z_e = 1 \wedge Z_{e'} = 1].$$

Section 17: One-Way Functions

In the next couple of sections we introduce one-way functions and pseudo-random generators, and show how to construct a pseudo-random generator from a one-way function. The reason for interest in these cryptographic functions and for the reduction from a one-way function to a pseudo-random generator is that there are a lot of natural examples of functions that seem to be one-way, and pseudo-random generators are extremely useful in the design of cryptographic protocols and in derandomization of algorithms.

Intuitively, a one-way function is a function that is easy to compute but hard for any time-bounded adversary to invert on a random input. To gauge the success of an adversary in breaking a cryptographic function, we use the following measure.

Definition (time/success ratio): The *time/success ratio* of an adversary for breaking a cryptographic function is $T(n)/\delta(n)$, where $T(n)$ is the run time of the adversary and $\delta(n)$ is the success probability of the adversary with respect to inputs parameterized by n. The definition of the success probability depends on the cryptographic function.

Definition (one-way function): Let $f(x)$ be a function computable in time polynomial in $\|x\|$. The success probability (inverting probability) of adversary A for f is

$$\delta(n) = \Pr_{x \in_R \{0,1\}^n}[f(A(f(x))) = f(x)].$$

Then, f is a $\mathbf{S}(n)$-secure one-way function if every A has time/success ratio at least $\mathbf{S}(n)$.

Definition (one-way permutation): Exactly the same as the definition of a one-way function, except that $\|f(x)\| = \|x\|$ and f as a function of $x \in \{0,1\}^n$ is a permutation.

Examples of Conjectured one-way functions

Here are some natural examples that may eventually be proven to be one-way functions. Plenty of others can be found in the literature. In the following, p and q are primes of length n.

Factoring problem : Define $f(p,q) = pq$. It is possible to compute pq given p and q in $n^{\mathcal{O}(1)}$ time. However, there is no known polynomial-time function that on input pq can produce p and q on average for randomly chosen pairs of primes $\langle p, q \rangle$

Discrete log problem : Let g be a generator of $\mathcal{Z}_p^*$, i.e., for all $y \in \mathcal{Z}_p^*$, there is a unique $x \in \mathcal{Z}_{p-1}$ such that $g^x = y \bmod p$. Given p, g and $x \in \mathcal{Z}_{p-1}$, define $f(p, g, x) = \langle p, g, g^x \bmod p \rangle$. It is possible to compute $g^x \bmod p$ given p, g and x in $n^{\mathcal{O}(1)}$ time. The discrete log function is a permutation as a function of x, i.e., the unique inverse of $f(p, g, x)$ is $\langle p, g, x \rangle$. The values of p and g are not necessarily chosen randomly. The prime p is selected to have special properties which seem in practice to make the discrete log function hard to invert. An example of such a property is that p is selected so that that $p - 1$ has some fairly large prime divisors. For a large class of primes p and generators g there is no known polynomial-time function that on input p, g and $g^x \bmod p$ can produce x on average for $x \in_R \mathcal{Z}_{p-1}$.

Root extraction problem : Given p, q, $e \in \mathcal{Z}_{p-1}$ and $y \in \mathcal{Z}_p$, define $f(p, q, e, y) = \langle pq, e, y^e \bmod pq \rangle$. It is possible to compute $y^e \bmod pq$ given pq, e and y in $n^{\mathcal{O}(1)}$ time. To make the inversion problem hard, it is important that the factorization of the modulus is not part of the output, because given the factorization an inverse can be found in $n^{\mathcal{O}(1)}$ time. The value of the exponent e is not necessarily chosen randomly. For example, if $e = 2$ then the problem is to extract square roots, and this still seems to be a hard problem on average. There is no known polynomial-time function that on input pq, e and $y^e \bmod pq$ can produce an $y' \in \mathcal{Z}_p$ such that ${y'}^e = y^e \bmod pq$ when p and q are randomly chosen according to a distribution for which factoring is hard and $y \in_R \mathcal{Z}_p$. There is a strong connection between this problem when $e = 2$ and the factoring problem.

Subset sum problem : Let $a \in \{0,1\}^n$ and $b \in \{0,1\}^{n \times n}$. Given a and b, define $f(a, b) = \langle \sum_{i=1}^n a_i \cdot b_i, b \rangle$, where $a_i \in \{0,1\}$ and b_i is an n-bit integer in this expression and where the sum is over the integers. It is possible to compute $\sum_{i=1}^n a_i \cdot b_i$ given a and b in $n^{\mathcal{O}(1)}$ time. However, there is no known polynomial-time function that on input $\sum_{i=1}^n a_i \cdot b_i$ and b can produce $a' \in \{0,1\}^n$ such that $\sum_{i=1}^n a'_i \cdot b_i = \sum_{i=1}^n a_i \cdot b_i$ on average when $a \in_R \{0,1\}^n$ and $b \in_R \{0,1\}^{n \times n}$.

Exercise 5 : Let $A \in_R \{0,1\}^n$ and let $B \in_R \{0,1\}^{n \times (n+1)}$. Prove that the probability

$$f(A, B) = \langle \sum_{i=1}^{n} A_i \cdot B_i, B \rangle$$

has a unique inverse is lower bounded by a constant strictly greater than zero independent of n. Note that in contrast to the previous definition where $\|B_i\| = n$, here $\|B_i\| = n + 1$.

Section 18: Hidden Bit Theorem

The main result of these sections is the construction of a pseudo-random generator from any one-way permutation. In this section, we present the main technical content of this reduction, the Hidden Bit Theorem, which is due to [14, Goldreich-Levin].

There are several technical parts in the reduction from any one-way permutation f to a pseudo-random generator g. Intuitively, the Hidden Bit Theorem is the part that transforms the one-wayness of f into a bit b such that: (1) b is completely determined by information that is available to any adversary; (2) nevertheless b looks random to any appropriately time-restricted adversary. It is from this bit b that the generator g eventually derives its pseudo-randomness. The guarantee from the reduction is that any successful adversary for distinguishing the output of g from a truly random string can be converted into an adversary for predicting b, which in turn can be converted into an adversary for inverting f.

The definition of a computationally hidden but statistically meaningful bit and the realization of its importance as a basic building block for cryptographic constructions is from [7, Blum-Micali].

The construction of a hidden bit using the inner product bit, the Hidden Bit Theorem and the Hidden Bit Technical Theorem are all from [14, Goldreich-Levin]. The simpler proof given here of Hidden Bit Technical Theorem is due to C. Rackoff, R. Venkatesan and L. Levin, inspired by [4, Alexi-Chor-Goldreich-Schnorr].

Definition (inner product bit is hidden): Let $f(x)$ be a polynomial-time computable function. Let $x \in \{0,1\}^n$ and $z \in \{0,1\}^n$. Then, the *inner product bit* of $f(x)$ is $x \odot z$. The success probability (prediction probability) of adversary A for the inner product bit of f is

$$\delta(n) = \Pr_{x,z \in_R \{0,1\}^n}[A(f(x),z) = x \odot z] - \Pr_{x,z \in_R \{0,1\}^n}[A(f(x),z) \neq x \odot z].$$

Then, the inner product bit of f is a $\mathbf{S}(n)$-secure if every A has time/success ratio at least $\mathbf{S}(n)$.

Hidden Bit Theorem : If f is a one-way function then the inner product bit of f is hidden. In particular, there is a **TM** M such that if A is an adversary with time/success ratio $\mathbf{S}(n)$ for predicting the inner product bit then M^A is an adversary with time/success ratio $\mathbf{S}(n)^c$ for inverting f for some constant $c > 0$. (M^A denotes M making calls to the adversary A.)

PROOF: Suppose there is an adversary A for the inner product bit of f with success probability $\delta(n)$ and run time $T(n)$. We describe a **TM** M such that M^A is an adversary for f as a one-way function.

For $x \in \{0,1\}^n$ define

$$\delta_x^A = \Pr_{z \in_R \{0,1\}^n}[A(f(x), z) = x \odot z] - \Pr[A(f(x), z) \neq (x \odot z)].$$

Let $X \in_R \{0,1\}^n$. Because, for any $x \in \{0,1\}^n$, $|\delta_x^A| \leq 1$ and because $\mathrm{E}_X[\delta_X^A] = \delta(n)$, it follows that $\Pr_X[\delta_X^A \geq \delta(n)/2] \geq \delta(n)/2$. The **TM** M we describe below has the property that if $\delta_x^A \geq \delta(n)/2$ then M^A on input $f(x)$ succeeds in producing an x' such that $f(x') = f(x)$ with probability at least $1/2$. From this it follows that the inverting probability of M^A for f is at least $\delta(n)/4$.

Suppose the input to M^A is $f(x)$, where $\delta_x^A \geq \delta(n)/2$. Let S be the **TM** described below in the Hidden Bit Technical Theorem (page 36) and let $B(z) = A(f(x), z)$. The first step of M^A is to run S^B with input $\delta = \delta(n)/2$. When S makes an oracle query to B with input z, M runs A on input $\langle f(x), z\rangle$ and returns the answer $B(z) = A(f(x), z)$ to S. Because $\delta_x^A \geq \delta(n)/2$, by the Hidden Bit Technical Theorem, x is in the list $\mathcal{L}$ produced by S^B with probability at least $1/2$. The final step of M^A to do the following for all $x' \in \mathcal{L}$: Compute $f(x')$ and if $f(x') = f(x)$ then output x'.

The success probability of M^A for inverting $f(X)$ is at least $\delta(n)/4$. From the Hidden Bit Technical Theorem, it is not hard to see that the running time of M^A is dominated by the running time of S making queries to A to produce the list $\mathcal{L}$, which is $\mathcal{O}(n^3 T(n)/\delta(n)^4)$, where $T(n)$ is the running time of A. Thus, the time/success ratio of M^A is $\mathcal{O}(n^3 T(n)/\delta(n)^5)$. ◻

Generalized Inner Product Space

For the proof of the Hidden Bit Technical Theorem we use the following generalization of the Inner Product Space (page 30).

Generalized Inner Product Space : Let $\ell = \lceil \log(m+1) \rceil$. The sample space is $\mathcal{S} = \{0,1\}^{n \times \ell}$ and the distribution on sample points is $v \in_R \mathcal{S}$. For all $j \in \{0,1\}^\ell$, define random variable

$$T_j(v) = v \odot j.$$

It can be verified that $T_1(v), \ldots, T_m(v)$ are uniformly distributed on $\{0,1\}^n$ and pairwise independent.

Hidden Bit Technical Theorem : Let $B(z)$ be a **TM** which runs in time polynomial in n. and for each $x \in \{0,1\}^n$ define

$$\delta_x^B = \Pr_{z \in_R \{0,1\}^n}[B(z) = x \odot z] - \Pr_{z \in_R \{0,1\}^n}[B(z) \neq x \odot z].$$

There is a **TM** S such that for any B, S^B on input $\delta > 0$ produces a list $\mathcal{L} \subseteq \{0,1\}^n$ with the following property: For all $x \in \{0,1\}^n$, if $\delta_x^B \geq \delta$ then $x \in \mathcal{L}$ with probability at least $1/2$, where this probability is with respect to the random bits used by **TM** S^B. The running time of S^B is $\mathcal{O}(n^3 T/\delta^4)$, where T is the running time of B.

PROOF: For the proof, we find it convenient to consider bits as being $\{1,-1\}$-valued instead of $\{0,1\}$-valued. For $b \in \{0,1\}$, we let $\overline{b} = (-1)^b \in \{1,-1\}$.

Fix $x \in \{0,1\}^n$ such that $\delta_x^B \geq \delta$. We can write

$$\delta_x^B = \mathrm{E}_{z \in_R \{0,1\}^n}\left[\overline{B(z)} \cdot \overline{x \odot z}\right].$$

For all $i = 1, \ldots, n$, let $e_i \in \{0,1\}^n$ be the bit string $\langle 0^{i-1}, 1, 0^{n-i}\rangle$ and let

$$\mu_i = \delta_x^B \cdot \overline{x}_i.$$

It follows that

$$\mathrm{E}_{z \in_R \{0,1\}^n}\left[\overline{B(z)} \cdot \overline{x \odot (e_i \oplus z)}\right] = \mu_i.$$

This is because $\odot$ distributive over $\oplus$ implies that

$$\overline{x \odot (e_i \oplus z)} = \overline{x \odot e_i} \cdot \overline{x \odot z}$$

and because $\overline{x \odot e_i} = \overline{x}_i$, and thus

$$\overline{B(z)} \cdot \overline{x \odot (e_i \oplus z)} = \overline{B(z)} \cdot \overline{x \odot z} \cdot \overline{x}_i.$$

Setting $z' = e_i \oplus z$ it is easy to see that $z' \in_R \{0,1\}^n$ when $z \in_R \{0,1\}^n$ and $z = e_i \oplus z'$. Thus,

$$\mathrm{E}_{z' \in_R \{0,1\}^n}\left[\overline{B(e_i \oplus z')} \cdot \overline{x \odot z'}\right] = \mu_i.$$

The idea is to compute, simultaneously for all $i \in \{1, \ldots, n\}$, a good approximation Y_i of μ_i. We say that Y_i is a good approximation if $|Y_i - \mu_i| < \delta$. Define

$$\mathrm{bit}(Y_i) = \begin{cases} 0 & \text{if } Y_i \geq 0 \\ 1 & \text{if } Y_i < 0 \end{cases}$$

Because $|\mu_i| \geq \delta$, if Y_i is a good approximation then $\mathrm{bit}(Y_i) = x_i$. Let $m = \lceil 2n/\delta^2 \rceil$ and let $T_1, \ldots, T_m \in_R \{0,1\}^n$ be pairwise independent random variables. Let

$$Y_i = 1/m \cdot \sum_{j=1}^{m} \overline{B(e_i \oplus T_j)} \cdot \overline{x \odot T_j}.$$

Then, using the pairwise independence of the random variables and the fact that, for all j,

$$\mathrm{E}\left[\left(\overline{B(e_i \oplus T_j)} \cdot \overline{x \odot T_j} - \mu_i\right)^2\right] \leq 1,$$

it follows that

$$\mathrm{E}[(Y_i - \mu_i)^2] \leq 1/m.$$

From Chebychev's inequality it then follows that

$$\Pr[|Y_i - \mu_i| \geq \delta] \leq \mathrm{E}[(Y_i - \mu_i)^2]/\delta^2 \leq 1/(m\delta^2) \leq 1/(2n).$$

From this it follows that

$$\Pr[\exists i \in \{1, \ldots, n\} : |Y_i - \mu_i| \geq \delta] \leq 1/2,$$

and so

$$\Pr[\forall i \in \{1, \ldots, n\} : |Y_i - \mu_i| < \delta] \geq 1/2. \tag{2}$$

The only remaining difficulty is how to compute Y_i given $T_1, \ldots, T_m$. Everything is relatively easy to compute, except for the values of $x \odot T_j$ for all $j \in \{1, \ldots, m\}$. If $T_1, \ldots, T_m$ are chosen in the obvious way, i.e., each is chosen independently of all the others, then we need to be able to compute $x \odot T_j$ correctly for all $j \in \{1, \ldots, m\}$ and there is probably no feasible way to do this. (Recall that we don't know the value of x.) Instead, the approach is to take advantage of the fact that the analysis only requires $T_1, \ldots, T_m$ to be pairwise independent.

Let $\ell = \lceil \log(m+1) \rceil$ and let $v \in \{0,1\}^{n \times \ell}$. Let $T_1(v), \ldots, T_m(v)$ be as described in the Generalized Inner Product Space (page 35), i.e., for all $v \in \{0,1\}^{n \times \ell}$ and for all $j \in \{0,1\}^{\ell} - 0^{\ell}$, $T_j(v) = v \odot j$. As we describe, this particular construction allows feasible enumeration of all possible values of $x \odot T_j(v)$ for all $j \in \{1, \ldots, m\}$ without knowing x. Because of the properties stated above,

$$x \odot T_j(v) = x \odot (v \odot j) = (x \odot v) \odot j.$$

Thus, it is easy to compute, for all $j \in \{1, \ldots, m\}$, the value of $x \odot T_j(v)$ given $x \odot v$. From this we can compute, for all $i \in \{1, \ldots, n\}$,

$$Y_i(v) = 1/m \cdot \sum_{j=1}^{m} \overline{B(e_i \oplus T_j(v))} \cdot \overline{(x \odot v) \odot j}.$$

The key point is that there are only $2^\ell = \mathcal{O}(m)$ possible settings for $x \odot v$, and we try them all. For any x and v there is some $\beta \in \{0,1\}^\ell$ such that $\beta = x \odot v$. Let

$$Y_i(\beta, v) = 1/m \cdot \sum_{j=1}^{m} \overline{B(e_i \oplus T_j(v))} \cdot \overline{\beta \odot j},$$

i.e., $Y_i(\beta, v)$ is the value obtained when β is substituted for $x \odot v$ in the computation of $Y_i(v)$. Consider choosing $v \in_R \{0,1\}^{n \times \ell}$. Since from equation (2) above, the probability that $Y_i(x \odot v, v)$ is a good approximation for all $i \in \{1, \ldots, n\}$ is at least one-half, it follows that with probability at least one-half there is at least one $\beta \in \{0,1\}^\ell$ such that $Y_i(\beta, v)$ is simultaneously a good approximation for all $i \in \{1, \ldots, n\}$. For this value of β and for such a v, $\langle \mathrm{bit}(Y_1(\beta, v), \ldots, \mathrm{bit}(Y_n(\beta, v)) \rangle$ is equal to x.

Adversary S^B on input $\delta > 0$:

$m \leftarrow \lceil 2n/\delta^2 \rceil$.

$\ell \leftarrow \lceil \log(m+1) \rceil$.

$\mathcal{L} \leftarrow \emptyset$.

Choose $v \in_R \{0,1\}^{n \times \ell}$.

For all $\beta \in \{0,1\}^\ell$ do:

  For all $j = 1, \ldots, m$ do:

    Compute $T_j(v) = v \odot j$.

  For all $i = 1, \ldots, n$ do:

    Compute $Y_i(\beta, v) = 1/m \cdot \sum_{j=1}^{m} \overline{B(e_i \oplus T_j(v))} \cdot \overline{\beta \odot j}$.

  $\mathcal{L} \leftarrow \mathcal{L} \cup \{\langle \mathrm{bit}(Y_1(\beta, v)), \ldots, \mathrm{bit}(Y_n(\beta, v)) \rangle\}$.

From the above analysis, it follows that $x \in \mathcal{L}$ with probability at least $1/2$, where this probability is over the random choice of v.

As long as the running time T for computing B is large compared to n (which it is in our use of the Hidden Bit Technical Theorem to prove the Hidden Bit Theorem), the running time of S^B is $O\left(n^3 T/\delta^4\right)$. ◻

The following exercise shows that the inner product bit is special, i.e., it is certainly not the case that any bit of the input to f is hidden if f is a one-way function.

Exercise 6 : Describe a one-way permutation $f(x)$ where the first bit of x is not hidden given $f(x)$. Let $f(x)$ be any polynomial-time computable function. Show that if x_i can be predicted with probability greater than $1 - 1/(2n)$ given $\langle f(x), i\rangle$ when $x \in_R \{0,1\}^n$ and $i \in_R \{1, \ldots, n\}$ then f is not a one-way function.

The converse of the Hidden Bit Theorem is not true, i.e., there is a function f where the inner product bit is hidden but f is not a one-way function. This is the point of the following exercise.

Exercise 7 : Describe a polynomial-time computable function $f(x)$ which is certainly not a one-way function but for which the inner product bit is provably 2^n-secure.

Section 19: Pseudo-random Generators

[7, Blum-Micali] introduce the concept of a pseudo-random generator that is useful for cryptographic (and other) applications, and gave it the significance it has today by providing the first provable construction of a pseudo-random generator based on the conjectured difficulty of a well-known and well-studied computational problem. In particular, both the definition of pseudo-random generator based on the next bit test and the construction of a pseudo-random generator based on the difficulty of the discrete log problem (page 32) can be found in [7, Blum-Micali].

[37, Yao] introduces the now standard definition of a pseudo-random generator, and shows an equivalence between this definition and the next bit test introduced in [7, Blum-Micali]. The standard definition of a pseudo-random generator introduced by [37, Yao] is based on the concept of computational indistinguishability introduced previously in [15, Goldwasser-Micali].

Definition (pseudo-random generator): Let $g(x)$ be a polynomial-time computable function where $\ell(n) = \|g(x)\|$, $n = \|x\|$, and $\ell(n) > n$. The stretching parameter of $g(x)$ is $\ell(n) - n$. The success probability (distinguishing probability) of adversary A for g is

$$\delta(n) = \Pr_{x \in_R \{0,1\}^n}[A(g(x)) = 1] - \Pr_{z \in_R \{0,1\}^{\ell(n)}}[A(z) = 1].$$

Then, g is a $\mathbf{S}(n)$-secure pseudo-random generator if every A has time/success ratio at least $\mathbf{S}(n)$.

The following exercise shows that an immediate application of the Hidden Bit Theorem is the construction of a pseudo-random generator from a one-way permutation.

Exercise 8 : From the Hidden Bit Theorem, show that if $f(x)$ is a one-way permutation then $g(x, z) = \langle f(x), z, x \odot z\rangle$ is a pseudo-random generator that stretches by 1 bit. The reduction should describe a **TM** M with the property that if A is an adversary for distinguishing g with time/success ratio $\mathbf{S}(n)$ then M^A is an adversary for inverting f with time/success ratio $\mathbf{S}(n)^c$ for some constant $c > 0$.

The simple construction of a pseudo-random generator given in the previous exercise was one of the motivating forces behind the work of [14, Goldreich-Levin]. The reduction from an arbitrary one-way function to a pseudo-random generator can be found in [18, Håstad-Impagliazzo-Levin-Luby].

We can construct a pseudo-random generator that stretches by an arbitrary polynomial amount based on any one-way permutation. Let $f(x)$ be a one-way permutation . Define $g(x, z)$, where $\|z\| = \|x\| = n$, as

$$g(x, z) = \langle z, x \odot z, f(x) \odot z, f^{(2)}(x) \odot z, \ldots, f^{(\ell(n)-n-1)}(x) \odot z \rangle,$$

where $f^{(i)}$ is the function f composed with itself i times.

Theorem : If f is a one-way permutation then g is a pseudo-random generator. In particular, there is a **TM** M with the property that if A is an adversary for distinguishing g with time/success ratio $\mathbf{S}(n)$ then M^A is an adversary for inverting f with time/success ratio $\mathbf{S}(n)^c$ for some constant $c > 0$.

This theorem is a combination of a theorem due to [13, Goldreich-Goldwasser-Micali] and the Hidden Bit Theorem (page 34) of [14, Goldreich-Levin].

Exercise 9 : Prove the above theorem.

Section 20: #P and Approximate Counting

Recall that a language $\mathcal{L} \in \mathbf{NP}$ if there is an associated **TM** M such that, for all $x \in \{0,1\}^n$, $x \in \mathcal{L}$ iff $|W_x| \geq 1$, where $W_x = |\{y \in \{0,1\}^r : M(x,y) = 1\}|$. A function $f \in \#\mathbf{P}$ if there is an **NP** language $\mathcal{L}$ with an associated **TM** M such that, for all $x \in \{0,1\}^n$, $f(x) = |W_x|$. In words, $f(x)$ is the number of witnesses that show $x \in \mathcal{L}$. In particular, note that $f(x) = 0$ iff $x \notin \mathcal{L}$, and thus it is clear that a polynomial time algorithm for computing f immediately implies $\mathbf{P} = \mathbf{NP}$. The definition of the complexity class $\#\mathbf{P}$, and the realization of its importance, are due to [34, Valiant]. Examples of $f \in \#\mathbf{P}$ are the following:

- If x is the description of a graph then $f(x)$ is the number of perfect matchings in the graph, else $f(x) = 0$.
- If x is the description of a graph then $f(x)$ is the number of Hamiltonian tours in the graph, else $f(x) = 0$.
- If x is the description of a DNF boolean formula then $f(x)$ is the number of truth assignments that satisfy the formula, else $f(x) = 0$.
- If x is the description of a CNF boolean formula then $f(x)$ is the number of truth assignments that satisfy the formula, else $f(x) = 0$.

As there is a notion of completeness for **NP**, there is an analogous notion of completeness for $\#\mathbf{P}$. Intuitively, if a function f is $\#\mathbf{P}$-complete and if there is a polynomial time algorithm for computing f then there is a polynomial time algorithm for computing every $g \in \#\mathbf{P}$.

As shown in [34, Valiant], all four examples described above are $\#\mathbf{P}$-complete functions. Most often it is the case that if the language $\mathcal{L}$ is **NP**-complete then it takes little effort to show that the associated counting problem f is $\#\mathbf{P}$-complete, and this is the case for second and fourth examples. The first and third examples are more interesting because the associated **NP**-language can be decided in polynomial time. The proof that the third example is $\#\mathbf{P}$-complete is rather straightforward from the $\#\mathbf{P}$-completeness of the fourth example. However, the $\#\mathbf{P}$-completeness of the first example is not at all straightforward.

It turns out that many important counting problems are $\#\mathbf{P}$-complete, and unless $\mathbf{P} = \mathbf{NP}$ there is no hope of finding polynomial time algorithms for these problems. On the other hand, in practice it is often useful to provide a good approximation of the number of solutions. As before, let $\mu(W_x) = |W_x|/2^r$ be the fraction of witnesses for x among all possible witnesses. There are two potential definitions of what a good estimate means:

(1) Y is an ϵ-good absolute approximation of $\mu(W_x)$ if

$$\mu(W_x) - \epsilon \leq Y \leq \mu(W_x) + \epsilon.$$

(2) Y is an ϵ-good relative approximation of $\mu(W_x)$ if

$$\mu(W_x)(1 - \epsilon) \leq Y \leq \mu(W_x)(1 + \epsilon).$$

An estimate Y is more useful and meaningful if it is an ϵ-good relative approximation, especially in the typical case when $\mu(W_x)$ is small.

In recent years, a body of work has been devoted to finding fast algorithms to approximate $\#\mathbf{P}$-complete functions. Let f be a $\#\mathbf{P}$ function. Following [21, Karp-Luby], we say a randomized algorithm A provides a fully polynomial randomized approximation scheme (abbreviated to *fpras*) for f if, for every pair of positive input parameters (ϵ, δ), and for every input $x \in \{0,1\}^n$,

(1) $A(x, \epsilon, \delta)$ is an ϵ-good relative approximation of $f(x)$ with probability at least $1 - \delta$. The probability is with respect to the source of randomness used by A.

(2) The run time $A(x, \epsilon, \delta)$ is bounded by a polynomial in $\|x\|$, $1/\epsilon$ and $\log(1/\delta)$.

Let us say that A is a *weak fpras* if requirement (1) in the above definition is changed to say that $A(x)$ is an ϵ-good absolute approximation instead of an ϵ-good relative approximation. Based on the standard sampling algorithm described below, it is easy to see there is a weak fpras for every $f \in \#\mathbf{P}$. On the other hand, a fpras for the CNF counting problem immediately implies $\mathbf{RP} = \mathbf{NP}$.

In the following sections, we develop a fpras for the DNF counting problem and for a related problem. The following simple and standard sampling algorithm at a very high level provides the general outline for the algorithms for both problems. Suppose we have a finite (but large) universe U of known size $|U|$, and our goal is to estimate the size of some set $G \subset U$ of unknown size. A trial of the algorithm for estimating $|G|$ consists of the following two steps:

(1) Choose $s \in_R U$.

(2) See if $s \in G$.

Let b be an easily computable upper bound on $|U|/|G|$. The algorithm performs $N = 4b \ln(2/\delta)/\epsilon^2$ independent trials, and the output Y is the fraction of these N trials where an element of G is chosen, multiplied by $|U|$. A standard analysis using an inequality due to Bernstein [32, Renyi] shows that for $\epsilon < 1$,

$$\Pr[|G|(1-\epsilon) \leq Y \leq |G|(1+\epsilon)] \geq 1-\delta. \qquad (3)$$

(See for example [22, Karp-Luby-Madras] for a proof.)

The key points about the sampling algorithm are the following:

(a) The universe U should be defined in such a way that $|U|$ is easy to compute.

(b) Steps (1) and (2) of the trial can be performed efficiently.

(c) $|G|$ is known a priori to be a significant fraction of $|U|$, i.e. b is polynomially bounded.

Section 21: DNF Counting

Let $y = \langle y_1, \ldots, y_r \rangle$ be a collection of r-boolean variables and let F be a boolean formula in disjunctive normal form (DNF formula); i.e, $F = c_1 \vee c_2 \vee \cdots \vee c_m$, where $c_i = z_{i_1} \wedge \cdots \wedge z_{i_{\ell_i}}$ for some set of literals

$$\{z_{i_1}, \ldots, z_{i_{\ell_i}}\} \subseteq \{y_1, \ldots y_r, \bar{y}_1, \ldots, \bar{y}_r\}.$$

For a truth assignment $a \in \{0,1\}^r$ to y, let $M(F,a) = 1$ if a satisfies F, and let $M(F,a) = 0$ otherwise. Let $f(F) = |\{a \in \{0,1\}^r : M(F,a) = 1\}|$. It is clear that $f \in \#\mathbf{P}$, and, as mentioned before, f is $\#\mathbf{P}$-complete. We describe below a fpras algorithm for f due to [21, Karp-Luby], [22, Karp-Luby-Madras].

A naive approach to approximate $f(F)$ is the following. Let the sample space be the set $\{0,1\}^r$ of all possible truth assignments. Choose several random truth assignments, and estimate $f(F)$ by the fraction of these truth assignments that satisfy F. The problem with this approach is that if F is satisfied by an exponentially small percentage of truth assignments then this approach requires an exponential number of samples.

Instead, we design the following sample space. Let C_i be the set of truth assignments that satisfy clause c_i. Let

$$U = \{(i,a) : i \in \{1, \ldots, m\} \wedge a \in C_i\}$$

and let $G \subseteq U$ be defined by

$$G = \{(i,a) \in U : \text{there is no } j < i \text{ such that } (j,a) \in U\}.$$

Notice that $|U| = \sum_{i \in \{1,\ldots,m\}} |C_i|$, and $|C_i| = 2^{r-\ell_i}$, and thus $|U|$ is easy to compute. Note also that $|G| = f(F)$, because for each truth assignment a that satisfies F there is a unique smallest index i such that $a \in C_i$. Furthermore,

$$\frac{|U|}{|G|} \leq \frac{\sum_{i \in \{1,\ldots,m\}} |C_i|}{\max_{i \in \{1,\ldots,m\}} |C_i|} \leq m.$$

Therefore, from Equation 3 of the previous section, we can approximate $f(F)$ with $N = \frac{4m}{\epsilon^2} \log(\frac{1}{\delta})$ trials. In each trial we will

1. Choose index $i \in \{1, \ldots, m\}$ with probability $|C_i|/|U|$. This requires time $\mathcal{O}(\log m)$ time (with some preprocessing, we leave as an easy exercise what preprocessing to do to achieve this time bound.)

2. Choose $a \in_R C_i$. This step takes $\mathcal{O}(r)$ time.

3. See if $(i, a) \in G$. This can be done in the obvious way in time $\mathcal{O}(rm)$.

4. The value produced by the trial is $|U|$ if $(i, a) \in G$, and 0 otherwise.

The overall estimate is the average of the values produced by the N trials. By Equation 3, this is guaranteed to be an ϵ-good relative estimate of $f(F)$ with probability at least $1 - \delta$. The overall running time is $\mathcal{O}(\frac{rm^2}{\epsilon^2} \log(\frac{1}{\delta}))$.

Section 22: GF[2] Polynomial Counting

Let $y = \langle y_1, \ldots, y_r \rangle$ be a collection of r variables over GF[2] and let F be a polynomial over GF[2] with respect to the variables y, i.e., i.e, $F = t_1 \oplus t_2 \oplus \cdots \oplus t_m$, where term $t_i = y_{i_1} \odot \cdots \odot y_{i_{\ell_i}}$ for some subset $\{y_{i_1}, \ldots, y_{i_{\ell_i}}\}$ of y. For an assignment $a \in \{0,1\}^r$ to y, let $M(F,a) = 1$ if a is a zero of F, i.e., a satisfies an even number of the m terms of F, and $M(F,a) = 0$ otherwise. Let $f(F) = |\{a \in \{0,1\}^r : M(F,a) = 1\}|$. It is **#P**-complete to compute f. The following fpras for approximating f is due to [24, Karpinski-Luby]

We design two different fpras algorithms A_0 and A_1; A_0 is used in the case when F does not contain the constant term 1 and A_1 is used in the case when F contains the term 1. Note that the term 1 corresponds to the product of the empty set of variables, and is satisfied by all assignments to y. The analyses of the two algorithms are very similar. The running time of A_0 is $\mathcal{O}(rm^2 \ln(1/\delta)/\epsilon^2)$ and the running time of A_1 is $\mathcal{O}(rm^3 \ln(1/\delta)/\epsilon^2)$.

We first describe algorithm A_0. Let U be the set $\{0,1\}^r$ of all assignments to y, and let H_{even} be the set of all assignments that satisfy an even number of terms. A trial of the algorithm consists of choosing an assignment $a \in_R \{0,1\}^r$ and testing if $a \in H_{\text{even}}$. The outcome of a trial is $|U| = 2^r$ if $a \in H_{\text{even}}$ and the outcome is 0 otherwise. The output of the algorithm is the average of the outcomes of all the trials.

The most time consuming part of the trial is to test if $a \in H_{\text{even}}$, and this takes time $\mathcal{O}(rm)$. The corollary on page 49 shows that $|U|/|H_{\text{even}}| \le m+1$, and thus $N = 4(m+1)\ln(2/\delta)/\epsilon^2$ trials suffice. Thus, the total running time of A_0 is $\mathcal{O}(rm^2 \ln(2/\delta)/\epsilon^2)$.

We now describe algorithm A_1. The outline of the algorithm is borrowed from the DNF approximation algorithm described in the previous section. We provide a self-contained description of the algorithm. Let F be the input polynomial with the constant term 1 discarded. Thus, the problem is to approximate the number of assignments that satisfy an odd number of terms of F. For all $i = 1, \ldots, m$, let T_i be the set of assignments that make term t_i evaluate to 1. Analogous to the DNF algorithm, let

$$U = \{(i,a) : i \in \{1, \ldots, m\} \wedge a \in T_i\}$$

and let $G \subseteq U$ be defined by

$$G = \{(i,a) \in U : \text{there is no } j < i \text{ such that } (j,a) \in U\}.$$

Let $G_{\mathrm{odd}} \subseteq G$ be defined by

$$G_{\mathrm{odd}} = \{(i,a) \in G : |\{j \in \{1,\ldots,m\} : j \in T_i\}| \bmod 2 = 1\},$$

i.e., (i,a) is in G_{odd} if it is in G and if a makes an odd number of terms evaluate to 1. The key point is that $|G_{\mathrm{odd}}|$ is the quantity we want to approximate. One trial of the algorithm consists of choosing $(i,a) \in_R U$ and then testing if $(i,a) \in G_{\mathrm{odd}}$: if yes then the value produced by the trial is $|U|$, else the value is 0.

We now verify that the criteria (a), (b) and (c) described on page 44 are satisfied. The computation of $|U|$ and the method for choosing $(i,a) \in_R U$ is analogous to the method for the DNF approximation algorithm described in the previous section. The most time consuming part of the trial is to test if $(i,a) \in G_{\mathrm{odd}}$, and this takes $\mathcal{O}(rm)$ time.

The final portion of the analysis is to show that $|U|/|G_{\mathrm{odd}}|$ is not too large. As described in the previous section, $|U|/|G| \leq m$. The theorem given below shows that $|G|/|G_{\mathrm{odd}}| \leq m$. It follows that $|U|/|G_{\mathrm{odd}}| \leq m^2$, and thus from Equation 3 it follows that $N = 4m^2 \ln(2/\delta)/\epsilon^2$ trials suffice. Thus, the total running time of A_1 is $\mathcal{O}(rm^3 \ln(2/\delta)/\epsilon^2)$.

Theorem : Let F be a multivariate polynomial over GF[2] with no duplicate terms and m terms in total. Let H be the set of assignments to the variables that satisfy at least one term, and let H_{odd} be the set of assignments that satisfy an odd number of terms. Then,

$$|H|/|H_{\mathrm{odd}}| \leq m.$$

PROOF: The basic idea of the proof is to define a function $h : H \rightarrow H_{\mathrm{odd}}$ in such a way that the mapping is at most m-to-1, i.e. for each $a \in H_{\mathrm{odd}}$, $|h^{-1}(a)| \leq m$. From this the theorem follows.

The mapping h is defined as follows. For each $a \in H$, choose any term t_i that is satisfied by a such that there is no term t_j which is satisfied and which contains all the variables in t_i. It is always possible to choose such a term because F does not contain two identical terms. Without loss of generality, let this be term t_1 and let $S = \{y_1, \ldots, y_k\}$ be the variables in t_1 (all of these variables are equal to 1 in a). For any $S' \subseteq S$, let $a(S')$ be the assignment obtained from a by changing the values of all variables in $S - S'$ from 1 to 0.

Claim : There is at least one $S' \subseteq S$ such that $a(S')$ satisfies an odd number of terms of F.

Proof of of Claim : For each $S' \subseteq S$, let $p(S')$ be the parity of the number of terms that are satisfied by assignment $a(S')$ and let $q(S')$ be

	$\emptyset$	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	{1,2,3}
$\emptyset$	1	0	0	0	0	0	0	0
{1}	1	1	0	0	0	0	0	0
{2}	1	0	1	0	0	0	0	0
{3}	1	0	0	1	0	0	0	0
{1,2}	1	1	1	0	1	0	0	0
{1,3}	1	1	0	1	0	1	0	0
{2,3}	1	0	1	1	0	0	1	0
{1,2,3}	1	1	1	1	1	1	1	1

Figure 1: The matrix R for $k = 3$

the parity of the number of terms t_i such that $t_i \cap S = S'$. By the way term t_1 is chosen, t_1 is the only term t_i that satisfies $t_i \cap S = S$, and thus $q(S) = 1$. We can view $p(\cdot)$ and $q(\cdot)$ as column vectors of length 2^k with entries from GF[2], where the first entry corresponds to $S' = \emptyset$ and the last entry corresponds to $S' = S$. Then, it can be verified that there is a $2^k \times 2^k$ lower triangular matrix R over GF[2] with main diagonal 1 such that $R \odot q(\cdot) = p(\cdot)$. In particular, row S' in R has a 1 in column S'' if and only if $S'' \subseteq S'$. (See Figure 1.)

Because R is invertible over GF[2] and because $q(\cdot) \not\equiv 0$, it follows that for at least one $S' \subseteq S$, $p(S') = 1$. For this S', $a(S')$ satisfies an odd number of terms. This complete the proof of the claim. ❏

We now complete the proof of the theorem. To define $h(a)$, we arbitrarily choose any S' such that $a(S')$ satisfies an odd number of terms and let $h(a) = a(S')$. Finally, we argue that for each $b \in H_{\text{odd}}$ there are at most m distinct assignments $a \in H$ such that $h(a) = b$. This is because each such a is either equal to b, or is obtained by taking one of the terms not satisfied by b (there are at most $m - 1$, since b must satisfy an odd number and thus at least one term) and setting the values of all variables in this term to 1. ❏

Note that the theorem holds even in the case when F contains the constant term 1. This fact is used in the proof of the following corollary.

Corollary : Let F be a multivariate polynomial over GF[2] with no duplicate terms, no occurence of the constant term 1 and m terms in total. Let U be the set of all assignments to the variables, and let H_{even} be the set of assignments that satisfy an even number of terms. Then, $|U|/|H_{\text{even}}| \leq m + 1$.

PROOF: Let $\bar{F} = F \oplus 1$. Then, $\bar{H}_{\text{odd}} = H_{\text{even}}$. Because F does not contain the term 1, $\bar{F}$ contains $m+1$ terms in total, no duplicate terms, and every assignment satisfies the constant 1 term, and thus $\bar{H} = U$. By the above theorem, $|U|/|H_{\text{even}}| = |\bar{H}|/|\bar{H}_{\text{odd}}| \leq m+1$. ◻

The bound given in the theorem is optimal. To see this, let m be a power of two and let $F = \prod_{i=1,\ldots,\log m}(1 \oplus y_i)\prod_{j=\log m,\ldots,r} y_j$. When F is viewed as a polynomial over GF[2] and expanded out the number of terms is m. When viewed over GF[2], $F = 1$ has a unique solution, whereas when viewed as a DNF formula, F has m satisfying assignments.

The bound given in the above corollary is also optimal. To see this, consider $F = 1 \oplus \prod_{i=1,\ldots,r}(1 \oplus y_i)$. When F is viewed as a polynomial over GF[2] and expanded out the 1 term is cancelled and the total number of terms is $m = 2^r - 1$. When viewed over GF[2], $F = 0$ has a unique solution, and thus $|U|/|H_{\text{even}}| = 2^r = m+1$.

Section 23: Bounded Depth Circuit Counting

Recall that in Sections 9–15 we introduced ways to reduce the amount of randomness when amplifying the probability of correctly deciding membership of x with respect to an **RP** or **BPP** language $\mathcal{L}$. However, the number of random bits needed was still at least the number to choose a single potential witness at random, i.e., at least r bits. In this section, we show how to deterministically decide membership for a **BPP** language $\mathcal{L}$ where the **NP TM** M associated with $\mathcal{L}$ is restricted to be expressible as a constant depth unbounded fan-in circuit. While this restriction on M may seem to be severe at first glance (and it is), such a machine M is nevertheless powerful enough to express a rich class of #**P**-complete problems. For example, for the DNF counting problem described in Section 21, the associated **TM** M can be expressed as a depth 2 unbounded fan-in circuit.

Definition ($\mathcal{C}_n^d$): Let $\mathcal{C}_n^d$ be the set of all circuits with n boolean input variables $z = \langle z_1, \ldots, z_n \rangle$ of depth d. A circuit $C \in \mathcal{C}_n^d$ consists of $\wedge$-gates and $\vee$-gates, where each gate is allowed unbounded fan-in. C consists of d levels of gates, where all gates at a given level are the same type. All the gates at level 1 have as inputs any mixture of variables and their negations. For all $i \in \{2, \ldots, d\}$, all gates at level i receive their inputs from the gates at level $i-1$. There is a single gate at level d, and either its value or the negation of its value is considered to be the output $C(z) \in \{0,1\}$ of C.

For $x \in \{0,1\}^n$, $x \odot x = \oplus_{i \in \{1,\ldots,n\}} x_i$ is the parity of the number of ones in x.

Definition (predicting the parity of its inputs): For any circuit $C \in \mathcal{C}_n^d$, let p_C be the prediction probability of C for the parity of its input, i.e.,

$$p_C = \Pr_{z \in_R \{0,1\}^n}[C(z) = z \odot z] - 1/2.$$

Let T_C be the total number of gates in C. The time/success ratio of C for predicting the parity of its inputs is $S_C = T_C / p_C$.

The following lower bound theorem is a culmination of a number of papers, i.e., [12, Furst-Saxe-Sipser], [2, Ajtai], [38, Yao1], [8, Cai], [17, Håstad].

Parity Theorem : There is a constant $\kappa > 0$ such that for any $C \in \mathcal{C}_n^d$, the time/success ratio S_C of C satisfies $S_C \geq 2^{n^{\kappa/d}}$.

Let $\mathcal{L} \in \mathbf{BPP}$ and let M be the **TM** associated with $\mathcal{L}$ with associated constants $\mathrm{c_{yes}}$ and $\mathrm{c_{no}}$. Suppose that the computation of $M(x, y)$ for a fixed value of $x \in \{0,1\}^n$ as a function of $y \in \{0,1\}^r$ can be expressed as a circuit $C \in \mathcal{C}_r^d$. Let $g : \{0,1\}^\ell \to \{0,1\}^r$ be a function.

Definition (distinguishing probability of C for g): For a circuit $C \in \mathcal{C}_r^d$, we let δ_C be the distinguishing probability of C for g (analogous to the definition of a pseudo-random generator (page 40)), i.e.,

$$\delta_C = |\Pr_{s \in_R \{0,1\}^\ell}[C(g(s)) = 1] - \Pr_{y \in_R \{0,1\}^r}[C(y) = 1]|.$$

Let T_C be the total number of gates in C. The time/success ratio of C for distinguishing g is $S_C = T_C/\delta_C$.

The key to the constructions of [29, Nisan], [30, Nisan-Wigderson] is to design g in such a way that the following properties hold:

- The length ℓ of the input to g is much shorter than the length r of its output.
- The time to compute $g(s) \in \{0,1\}^r$ given $s \in \{0,1\}^\ell$ is polynomial in r.
- The distinguishing probability δ_C of C for g satisfies

$$\delta_C < (\mathrm{c_{yes}} - \mathrm{c_{no}})/2.$$

Given these properties, to decide membership of $x \in \mathcal{L}$ is easy: Simply run $C(g(s))$ for all $s \in \{0,1\}^\ell$, and then decide $x \in \mathcal{L}$ iff the fraction of these inputs on which C produces the value 1 is at least $(\mathrm{c_{yes}} + \mathrm{c_{no}})/2$. It is not hard to verify that membership of x in $\mathcal{L}$ is always decided correctly. The run time for this procedure is 2^ℓ times the time for computing g on inputs of length ℓ (in the construction below, this takes time that is almost linear in the length r of the output of g) plus the time for computing C on inputs of length r.

Note that we can view g as generating a distribution on the r-bit input to C consisting of only 2^ℓ sample points that appears pseudo-random to C. If it were possible to set $\ell = c\log(r)$ for some constant $c > 0$ then membership of x in $\mathcal{L}$ could be decided in polynomial time. If this value of ℓ were achievable for $C \in \mathcal{C}_r^r$ then this would imply that $\mathbf{BPP} = \mathbf{P}$. (Contrast this with the results of Exercise 1 on page 15).

We now describe the generator g of [29, Nisan], [30, Nisan-Wigderson]. Set

$$\ell = \log(r)^{c(d+1)},$$

where $c > 0$ is a parameter that can be thought of as constant. Let $t_1, \ldots, t_r \subset \{1, \ldots, \ell\}$ be sets that satisfy the following two properties:

(1) For all $i \in \{1, \ldots, r\}$, $|t_i| = \sqrt{\ell}$.

(2) For all $i, j \in \{1, \ldots, r\}$, $i \neq j$, $|t_i \cap t_j| \leq \log(r)$.

We leave as Exercise 10 the efficient construction of the sets $t_1, \ldots, t_r$ with these two properties. For all $s \in \{0,1\}^\ell$, for each $i \in \{1, \ldots, r\}$, define function $b_i(s) = \oplus_{j \in t_i} s_j$, i.e., $b_i(s)$ is the parity of the number of ones in the bits of s indexed by t_i. Finally, let

$$g(s) = \langle b_1(s), \ldots, b_r(s) \rangle.$$

The following theorem is due to [28, Nisan], [30, Nisan-Wigderson].

Theorem : Let $q(r) = 2^{\log(r)^{c\kappa/2}}/r^3$. For all $C \in \mathcal{C}_r^d$, the time/success ratio S_C of C for distinguishing g satisfies $S_C \geq q(r)$.

PROOF: Suppose that $C \in \mathcal{C}_r^d$ has size T_C and distinguishing probability δ_C that satisfies $T_C/\delta_C < q(r)$. We show this implies there is a circuit $C' \in \mathcal{C}_{\sqrt{\ell}}^{d+1}$ such that the size $T_{C'}$ of C' is at most $r^2 + T_C$ and such that the prediction probability $p_{C'}$ of C' for the parity of its inputs is at least δ_C/r. From this it follows that the time/success ratio

$$T_{C'}/p_{C'} \leq r^3 \cdot T_C/\delta_C < 2^{(\sqrt{\ell})^{\frac{\kappa}{d+1}}},$$

and by the Parity Theorem, such a circuit C' with $\sqrt{\ell}$ inputs of depth $d+1$ cannot exist. Thus, it must be the case that $S_C \geq q(r)$.

The circuit C' will be derived from C and g based on the properties of the generator g. We first use a hybrid argument first used by [37, Yao] that has become standard. Let $s \in_R \{0,1\}^\ell$ and $y \in_R \{0,1\}^r$. Consider the following sequence of $r+1$ distributions on r-bit strings:

0^{th} **distribution:** $\langle b_1(s), \ldots, b_r(s) \rangle$.

i^{th} **distribution:** $\langle y_1, \ldots, y_i, b_{i+1}(s), \ldots, b_r(s) \rangle$.

r^{th} **distribution:** $\langle y_1, \ldots, y_r \rangle$.

Let $R_i \in \{0,1\}^r$ be the random variable distributed according to the i^{th} distribution, and let $p_i = \Pr_{R_i}[C(R_i) = 1]$. Note that $R_0 = g(s)$ and $R_r = y$, and thus

$$\delta_C = |\Pr_{R_0}[C(R_0) = 1] - \Pr_{R_r}[C(R_r) = 1]| = |p_0 - p_r|.$$

Assume without loss of generality that $p_0 - p_r$ is positive. It follows from the triangle inequality that there is some $i \in \{1, \ldots, r\}$ such that

$$\delta_i = p_{i-1} - p_i \geq \delta_C / r.$$

Fix such an i. Note that R_{i-1} and R_i both depend only on s and on $y_1, \ldots, y_i$, and that the only difference between R_{i-1} and R_i is that the i^{th} bit of R_{i-1} is $b_i(s)$ and the i^{th} bit of R_i is y_i.

Without loss of generality, let $t_i = \{1, \ldots, \sqrt{\ell}\}$, i.e., $b_i(s)$ depends on the first $\sqrt{\ell}$ bits of s. Let $s' = \langle s_1, \ldots, s_{\sqrt{\ell}} \rangle$. By an averaging argument, there is a setting of values for $y_1, \ldots, y_i$ and $s_{\sqrt{\ell}+1}, \ldots, s_\ell$ such that the distinguishing probability of C for R_{i-1} and R_i remains at least δ_i conditional on these fixed values for $y_1, \ldots, y_i$ and $s_{\sqrt{\ell}+1}, \ldots, s_\ell$. Note that in the conditional distributions both R_{i-1} and R_i both depend only on s'.

Let $F'(s', a)$ be the function of s' which can be thought of as computing C where the first $i-1$ inputs bits are set to the values for $y_1, \ldots, y_{i-1}$ fixed above, the i^{th} input is set to $a \in \{0,1\}$, and the remaining $r - i$ input bits are computed as $b'_{i+1}(s'), \ldots, b'_r(s')$, where $b'_j(s')$ is the value of $b_j(s', s_{\sqrt{\ell}+1}, \ldots, s_\ell)$ when $s_{\sqrt{\ell}+1}, \ldots, s_\ell$ are fixed as described above. The above analysis shows that

$$\delta_i = \Pr_{s' \in_R \{0,1\}^{\sqrt{\ell}}}[F'(s', s' \odot s') = 1] - \Pr_{s' \in_R \{0,1\}^{\sqrt{\ell}}}[F'(s', y_i) = 1].$$

There are two cases to consider, i.e., when $y_i = 1$ or $y_i = 0$. We assume that $y_i = 1$, as the case when $y_i = 0$ is similar. We leave it as an exercise to prove that in this case the prediction probability $p_{F'}$ of $F'(s', 1)$ for the parity of its inputs is at least $\delta_i \geq \delta_C / r$. The intuition is that $F'(s', a)$ is more biased towards producing a 1 when $a = s' \odot s'$ then when $a = 1$, and thus if $F'(s', 1)$ produces a 1 it is more likely that $s' \odot s' = 1$ then 0, whereas if $F'(s', 1) = 0$ then it is more likely that $s' \odot s' = 0$ then 1.

We now show that $F'(s', y_i)$ can be computed by a small circuit $C'(s')$ of depth $d+1$. Note that since for all $j \in \{i+1, \ldots, r\}$, $|t_j \cap t_i| \leq \log(r)$, it follows that $b'_j(s')$ depends on at most $\log(r)$ bits of s'. Furthermore, any function of k-bits can be expressed as either a DNF or CNF circuit with at most 2^k gates at the first level. Suppose without loss of generality that the first level of gates of C are $\wedge$-gates. Then, we can express $b'_j(s')$ as a DNF circuit C_j with at most r $\vee$-gates at the first level and a single $\wedge$-gate at the second level. We can then merge C_j into the circuit C by feeding all the values of the $\vee$-gates of C_j directly into the $\wedge$-gates at the first level of C that the j^{th} input of C originally fed into. In the end, we get a circuit $C'(s')$ with $\sqrt{\ell}$ inputs which computes $F'(s', y_i)$, where the depth of C' is $d+1$ and the number of gates $T_{C'}$ in C' is at most $T_C + r(r-i) \le T_C + r^2$.

$\sqcup\!\!\!\!\sqcap$

Corollary : Let $\mathcal{L}$ be a language in **BPP**. Suppose for all $x \in \{0,1\}^n$ the associated **TM** $M(x,y)$ as a function of $y \in \{0,1\}^r$ can be expressed as a circuit $C \in \mathcal{C}_r^d$ for some fixed positive integer d, where the number of gates T_C in C is at most r^c for some fixed positive value c. Then, membership of x in $\mathcal{L}$ can be decided by a deterministic computation in time $2^{\log(r)^{O(1)}}$.

Note that the above corollary applies to derandomize at least partially the randomized approximation algorithm for DNF counting described in Section 21. However, the Parity Theorem does not help at all directly in derandomizing the randomized approximation algorithm for GF[2] Polynomial Counting described in Section 23, because such a polynomial can easily compute the parity of the number of ones in its input.

Exercise 10 : Describe an algorithm that on input positive integers r, ℓ and s with $s \le \ell$, produces r sets $t_1, \ldots, t_r \subset \{1, \ldots, \ell\}$ such that

- For all $i \in \{1, \ldots, r\}$, $|t_i| = s$.
- For all $i, j \in \{1, \ldots, r\}$, $j \ne i$,

$$|t_j \cap t_i| \le \log\left(\frac{rs(1+\frac{s}{\ell})^s}{\ell}\right).$$

The run time of the algorithm should be polynomial in r and ℓ. Note that when $s = \sqrt{\ell}$ and $\ell \ge 8$ then $|t_j \cap t_i| \le \log(r)$.

Exercise 11 : Finish the proof of the above theorem by showing how the distinguishing probability can be converted into prediction probability.

Abbreviations :

- **STOC**: Proceedings of the ACM Symposium on Theory of Computing
- **FOCS**: Proceedings of the IEEE Foundations of Computer Science

References

[1] L. Adleman, "Two Theorems on Random Polynomial Time", **FOCS**, 1978, pp. 75–83.

[2] M. Ajtai, "$\sum_1^1$-Formulae on Finite Structures", *Annals of Pure and Applied Logic*, Vol. 24, 1983, pp. 1–48.

[3] M. Ajtai, J. Komlos, E. Szemeredi. "Deterministic simulation in LOGSPACE," **STOC**, 1987, p. 132.

[4] W. Alexi, B. Chor, O. Goldreich, C. Schnorr, "RSA/Rabin Functions: Certain Parts are as Hard as the Whole", *SIAM J. on Computing*, Vol. 17, No. 2, April 1988, pp. 194–209.

[5] N. Alon, L. Babai, A. Itai, "A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem", *Journal of Algorithms*, Vol. 7, 1986, pp. 567–583.

[6] C. Bennett, J. Gill, "Relative to a random oracle A, $\mathbf{P}^A \neq \mathbf{NP}^A \neq co-\mathbf{NP}^A$ with probability one", *Siam J. on Computing*, Vol. 10, 1981, pp. 96–113.

[7] M. Blum, S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Computing*, Vol. 13, 1984, pp. 850–864 A preliminary version appears in **FOCS**, 1982, pp. 112–117.

[8] J. Cai, "With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy", *J. of Computer and System Sci.*, Vol. 38, 1989, pp. 68–85. A preliminary version appears in **STOC**, 1986, pp. 21–29.

[9] B. Chor, O. Goldreich. "On the power of two-point sampling," *Journal of Complexity*, Vol. 5, 1989, pp. 96–106.

[10] A. Cohen, A. Wigderson. "Dispersers, Deterministic Amplification, and Weak Random Sources," **FOCS**, 1989, pp. 14–19.

[11] M. Fredman, J. Komlos, E. Szemeredi. "Storing a sparse table in O(1) worst case access time," *Journal of the ACM*, Vol. 31, 1984, pp. 538–544.

[12] M. Furst, J. Saxe, M. Sipser, "Parity, Circuits and the Polynomial Time Hierarchy", **FOCS**, 1981, pp. 260–270.

[13] O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions", *J. of ACM*, Vol. 33, No. 4, 1986, pp. 792–807. A preliminary version appears in **FOCS**, 1984.

[14] O. Goldreich, L. Levin, "A Hard-Core Predicate for any One-way Function", **STOC**, 1989, pp. 25–32.

[15] S. Goldwasser, S. Micali, "Probabilistic Encryption", *J. of Computer and System Sci.*, Vol. 28, 1984, pp. 270–299. A preliminary version appears in **STOC**, 1982, pp. 365–377.

[16] S. Goldwasser, M. Sipser. "Private coins vs. public coins in interactive proof systems," **STOC**, 1986, pp. 59–68.

[17] J. Håstad, "Computational limitations for small depth circuits", Ph.D. thesis, MIT press, 1986.

[18] J. Håstad, R. Impagliazzo, L. Levin, M. Luby, "Construction of a pseudo-random generator from any one-way function", *ICSI Technical Report*, 1991, No. 91–068.

[19] R. Impagliazzo, D. Zuckerman. "How to recycle random bits," **FOCS**, 1990, pp. 248–253.

[20] R. Impagliazzo, L. Levin, M. Luby, "A Pseudo-random generator from any one-way function," **STOC**, 1989, pp. 12–24.

[21] R.Karp, M. Luby, "Monte-Carlo Algorithms for the Planar Multiterminal Network Reliability Problem," *J. of Complexity*, Vol. 1, 1985, pp. 45–64.

[22] R. Karp, M. Luby, N. Madras, "Monte-Carlo Approximation Algorithms for Enumeration Problems," *J. of Algorithms*, Vol. 10, No. 3, 1989, pp. 429–448.

[23] R. Karp, N. Pippenger, M. Sipser, appears in: M. Sipser, "Expanders, Randomness, or Time versus Space", *First Annual Conference on Structure in Complexity Theory*, 1986, pp 325–329.

[24] M. Karpinski, M. Luby, "Approximating the Number of Solutions to a GF[2] Formula," *Journal of Algorithms*, Vol. 14, No. 2, March 1993, pp. 280–287.

[25] A. Lubotzky, R. Phillips, P. Sarnak, "Explicit expanders and the Ramanujan conjectures", **STOC**, 1986, pp. 240–246; (See also: A. Lubotzky, R. Phillips, P. Sarnak, "Ramanujan graphs", *Combinatorica*, Vol. 8, 1988, pp. 261–277).

[26] M. Luby, "A Simple Parallel Algorithm for the Maximal Independent Set Problem," *SIAM J. on Computing*, Vol. 15, No. 4, November 1986, pp. 1036–1053

[27] G. Margulis, "Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators" *Problemy Peredachi Informatsii*, Vol. 24, 1988, pp. 51–60 (in Russian). (English translation in *Problems of Information Transmission*, Vol. 24, 1988, pp. 39–46).

[28] N. Nisan, "RL $\subseteq$ SC," **STOC**, 1992, pp. 619–623.

[29] N. Nisan, "Pseudorandom bits for constant depth circuits", *Combinatorica*, Vol. 1, 1991, pp. 63-70.

[30] N. Nisan, A. Wigderson. "Hardness vs. Randomness", *J. of Comp. Sci. and Sys*, Vol. 49, No. 2, 1994, pp. 149–167.

[31] N. Nisan, D. Zuckerman. "More deterministic simulation in Logspace," **STOC**, 1993, pp. 235–244.

[32] A. Renyi, **Probability Theory**, 1970, North-Holland, Amsterdam.

[33] M. Sipser. "A complexity theoretic approach to randomness," **STOC**, 1983, pp. 330–335.

[34] L. Valiant, "The complexity of computing the permanent", *Theoretical Computer Science*, 1979, No. 8, 1979, pp. 189–201.

[35] L. Valiant, V. Vazirani. "NP is as easy as detecting unique solutions," *Theoretical Computer Science*, Vol. 47, 1986, pp. 85–93.

[36] M. Wegman, J. Carter. "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279.

[37] A. Yao, "Theory and Applications of Trapdoor Functions", **FOCS**, 1982, pp. 80–91.

[38] A. Yao, "Separating the Polynomial-Time Hierarchy by Oracles", **FOCS**, 1985, pp. 1–10.