1 Probabilistic Complexity Classes

1.1 BPP

BPP (Bounded Probabilistic Polynomial Time) is a class of languages that can be defined as follows:

$L \in$ **BPP** if $\exists$ PPT (Probabilistic Polynomial Time) algorithm A with the following properties:

- **Completeness:** $\forall x \in L, \underset{\text{A's coinflips}}{Pr}[A(x) = 1] > \frac{2}{3}$. In other words, if x is in the language, algorithm A will output 1 with probability greater than 2/3
- **Soundness:** $\forall x \in L, \underset{\text{A's coinflips}}{Pr}[A(x) = 1] < \frac{1}{3}$. Or the probability of the algorithm outputting 1 on an input not in the language is less that 1/3.

It is a large open question in computer science whether or not $P = BPP$.

1.1.1 Equality of Definition

The bound of 2/3 and 1/3 are selected arbitrarily. It is the case that any algorithm that demonstrates an advantage of $(\frac{1}{2} + \varepsilon)$ can be converted into an algorithm A' that has an negligible chance of failing. The conversion is as follows.

A' : For an algorithm A with advantage $(\frac{1}{2}+\varepsilon)$, run algorithm $A(x)$ k times with fresh randomness each time and output 1 if the majority of the k runs outputs 1, else output 0.

To demonstrate that this is correct, and to find the value needed for k, we need the following bound from probability.

Chernoff Bound: Let $X_1, X_2, ..., X_n$ be independent $\{0, 1\}$ random variables w/ probability that $X_i = 1$ is $0 \leq p \leq 1$.

- Let $S = \sum_{i=1}^{n} X_i$. Then the expected value $E[S] = p \cdot n$

The Chernoff Bound states that

$$\forall \varepsilon \;\; 0 \leq \varepsilon \leq 1, \; \Pr[S > (1+\varepsilon)E[S]] \leq e^{\frac{-\varepsilon^2 \cdot E[S]}{3}}$$

Plugging this into algorithm A', we will find the probability that $s > \frac{n}{2}$. Let $X_i = 1$ if on run i, it makes a mistake, else 0. This will give us the total number of mistakes that the algorithm A' makes, which is expected to be $n/3$, or $E[S] = n/3$. We set $\varepsilon = 1/2$ to get the following:

$$\Pr[S > (1 + \frac{1}{2})\frac{n}{3}] \leq e^{\frac{-\frac{1}{4}\cdot\frac{n}{3}}{3}}$$
$$\Pr[S > \frac{n}{2}] \leq e^{-\frac{n}{27}}$$

To get a bound of $1/e^{80}$, we would simply need to run the procedure $27 * 80$ times, a constant amount.

1.1.2 Other Probabilistic Complexity Classes

Clearly, since the properties of IP problems and BPP problems are the same, this strategy can be applied to IP problems.

EP (Expected Polynomial): The class of problems that are expected to run in polynomial time, but can potentially run in exponential time with negligible probability. (This can be achieved by running multiple algorithms in paralell)
(Did not write down the others he talked about)

2 co-NP $\in$ IP

co-NP is the class of all languages L such that for every string $x \notin L$ there exists a polynomial-size witness w with which a polynomial-time verifier can confirm that $x \notin L$.

For example, 3-SAT is an NP-Complete problem that aims to determine if x, a conjunction of 3-literal clauses can be satisfied given some truth assignment for all $x_0, ..., x_n$.

For example, an instance $\Phi = (x_1 \vee \overline{x_3} \vee x_5) \wedge (x_1 \vee \overline{x_7} \vee x_2)...$

This problem is in NP, as langauge membership of Φ can be conducted in polynomial time w/ witness $w =$ the satisfying truth assignment. However, testing if $\Phi \notin L$ is a much more difficult task, as a proof would essentially involve enumerating through all truth assignments to demonstrate none satisfy Φ. Because of this, the problem 3-UNSAT is co-NP (since testing if $x \notin$ 3-UNSAT is the same as $x \in$ 3-SAT). 3-UNSAT is a co-NP complete problem, meaning any problem in co-NP can be converted to an instance of 3-UNSAT in polynomial time. With this, we will show that 3-UNSAT $\in IP$ which shows that co-NP $\in IP$.

2.1 Arithmeticizing 3-Sat

Arithmeticization is the process of converting a valid instance of 3-UNSAT to an algebraic formula that will allow us to perform an interactive proof.

- If $x = 0$ x is False
- If $x \neq 0$ x is True
- $\vee$ gate (or) becomes $+$
- $\wedge$ gate (and) becomes $\cdot$
- $\neg x = 1 - x$

For notation purposes, we will refer to the arithmeticized boolean formula as $\Phi(x_1...x_n)$. To show that all truth assignments of $X = x_1...x_n$ do not satisfy Φ, we demonstrate the following:

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} ... \sum_{x_n \in \{0,1\}} \Phi(x_1...x_n) = 0$$

Or in other words, the sum of all truth assignments of the arithmeticized formula is 0. The proof idea involves the prover P demonstrating that they are able to perform calculations that are exponential in time, and providing the verifier with the ability to check that they are using the original equation as well as verifying that they are not cheating.

The proof uses the following:

- Arithmetisized clause for any assignment must be ≤ 3 ($(1 + 1 + 1)$ is at most 3)
- There are m clauses, so the total value must be $\leq 2^n \cdot 3^m$ (The max value is obtained where all m clauses yield a value of 3 for all 2^n possible truth value assignments)
- We pick a prime $q > 2^n 3^m$ and perform all calculations over a finite field $\mathbb{F}_q$

GOAL: The prover P needs to show the verifier V that:

1. P is capable of finding the aformentioned summation over $\Phi(X)$.

2. P must be using the same Φ as the verifier.

3. The summation over Φ must be equal to 0.

The procedure is performed as follows:

1. Verifier V selects a large prime q such that $q > 2^n 3^m$, and sends q to P to determine the field in which all arithmetic operations will be performed $\mathbb{F}_q$.

2. The infinitely powerful prover P defines a univariate polynomial (polynomial with single free variable) defined as

$$f_1(x_1) = \sum_{x_2 \in \{0,1\}} ... \sum_{x_n \in \{0,1\}} \Phi(x_1 ... x_n) = 0$$

   This first function will use x_1 as the free variable. It is important to note since there are m clauses, the highest potential degree of x_i will be m, so there are potentially m terms in $f_1(x_1)$ corresponding to m integer coefficients. Therefore, the size of this polynomial is polynomial. P then sends $f_1(x_1)$ to V.

3. V checks if $f_1(0) + f_1(1) = 0$ and rejects if it does not. This is addressing the third goal of the interaction. If Φ is unsatisfiable, this should be the case. (Note: $f_1(0) + f_1(1)$ essentially 'completes' the summation over all possible truth assignments. Since f_1 is the polynomial representing the summation over $x_2 ... x_n$, calculating $f(x_1)$ for all values of x_1 will be equivalent to the summation of the original multivariable polynomial over all possible truth assignments).

4. If step 3 is successful, then V selects a random $r_1 \overset{\$}{\leftarrow} \mathbb{F}_q$ and calculates $f_1(r_1) = v_1$. V then sends r to P.

5. P then calculates:

$$f_2(x_2) = \sum_{x_3 \in \{0,1\}} ... \sum_{x_n \in \{0,1\}} \Phi(r_1, x_2 ... x_n)$$

   This is the original polynomial with $x_1 = r_1$ with the free variable x_2 over all other assignment values of $x_3 ... x_n$. The function is sent to V, who checks if $f_2(0) + f_2(1) = r_1$. Again, if it does not V rejects. Steps 4 and 5 are repeated for all x_i values, replacing each x_{i-1} with the r_{i-1} value sent by V.

6. Once all n variables have been replaced, V checks if replacing all variables in the original arithmeticized equation with the randomly selected values r_i is the same as the final value calculated in the procedure:

   $$\Phi(R) = f_m(r_m)$$

   Where $R = (r_1, ..., r_n)$.

Now we argue completeness and soundness. Firstly, completeness follows directly from the interaction. With an honest prover and verifier, the chance $PV(\Phi)$ accepts if $\Phi \in L$ is 1 as all the calculations are directly executed using the original polynomial derived from the formula. Now we argue soundness:

Soundness:
To argue soundness, we need to demonstrate that a dishonest verifier cannot change the polynomial in the exchange or use a different polynomial than the one generated from arithmetization with greater than negligible probability. Consider the first polynomial sent to the verifier $f_1(x_1)$. Recall the final step of the algorithm is testing if $f_m(r_m) = \Phi(R)$. If f_{m-1} was not constructed using the initial Φ, the only way this check passes is if f_m intersects with Φ at a random location r, which can be shown to be negligible according to the following.

Schwartz-Zippel Lemma: Let P and Q be two polynomials of degree at most d over finite field $\mathbb{F}$. The probability that for $r \overset{\$}{\leftarrow} \mathbb{F}$, $\Pr[P(r) = Q(r)] \leq \frac{d}{|\mathbb{F}|}$. Intuitively, this is because two polynomials of degree d can intersect at most d times, so the chance that, at any random point r, they intersect is the number of intersections over the field size. For multivariable polynomials, intersections can occur at most $n * d$ times where n is the number of variables and d is the highest degree of any variable. So the probability that a dishonest verifier can produce an interaction that passes the protocol is the same as the probability that any polynomial that is not Ψ intersects with Ψ at some random point r, which, following from the lemma, can be at most $\frac{n \cdot m}{|\mathbb{F}|}$ (Negligile as $|\mathbb{F}| \geq 2^n 3^m$).