Comments on RP

- In analogy to Proposition 36 (p. 326), a “yes” instance of an RP problem has many certificates (witnesses).
- There are no false positives.
- If we associate nondeterministic steps with flipping fair coins, then we can cast RP in the language of probability.
  - If $x \in L$, then $N(x)$ halts with “yes” with probability at least 0.5 .
  - If $x \notin L$, then $N(x)$ halts with “no.”

Comments on RP (concluded)

- The probability of false negatives is $\epsilon \leq 0.5$.
- But *any* constant between 0 and 1 can replace 0.5.
  - Repeat the algorithm $k = \lceil -\frac{1}{\log_2 \epsilon} \rceil$ times and answer "no" only if all runs answer "no."
  - The probability of false negatives becomes $\epsilon^k \leq 0.5$.
- In fact, ϵ can be arbitrarily close to 1 as long as it is at most $1 - 1/q(n)$ for some polynomial $q(n)$.
  - $-\frac{1}{\log_2 \epsilon} = O(\frac{1}{1-\epsilon}) = O(q(n))$.

Where RP Fits

- $P \subseteq RP \subseteq NP$.
  - A deterministic TM is like a Monte Carlo TM except that all the coin flips are ignored.
  - A Monte Carlo TM is an NTM with extra demands on the number of accepting paths.
- COMPOSITENESS $\in$ RP;[a] PRIMES $\in$ coRP; PRIMES $\in$ RP.[b]
  - In fact, PRIMES $\in$ P.[c]
- RP $\cup$ coRP is an alternative "plausible" notion of efficient computation.

[a] Rabin (1976) and Solovay and Strassen (1977).
[b] Adleman and Huang (1987).
[c] Agrawal, Kayal, and Saxena (2002).

ZPP[a] (Zero Probabilistic Polynomial)

- The class **ZPP** is defined as RP $\cap$ coRP.
- A language in ZPP has *two* Monte Carlo algorithms, one with no false positives and the other with no false negatives.
- If we repeatedly run both Monte Carlo algorithms, *eventually* one definite answer will come (unlike RP).
  - A *positive* answer from the one without false positives.
  - A *negative* answer from the one without false negatives.

[a] Gill (1977).

The ZPP Algorithm (**Las Vegas**)

```
1: {Suppose $L \in$ ZPP.}
2: {$N_1$ has no false positives, and $N_2$ has no false
   negatives.}
3: while true do
4:     if $N_1(x) =$ "yes" then
5:         return "yes";
6:     end if
7:     if $N_2(x) =$ "no" then
8:         return "no";
9:     end if
10: end while
```

ZPP (concluded)

- The *expected* running time for the correct answer to emerge is polynomial.
  - The probability that a run of the 2 algorithms does not generate a definite answer is 0.5 (why?).
  - Let $p(n)$ be the running time of each run of the while-loop.
  - The expected running time for a definite answer is

$$\sum_{i=1}^{\infty} 0.5^i i p(n) = 2p(n).$$

- Essentially, ZPP is the class of problems that can be solved, without errors, in expected polynomial time.

Large Deviations

- Suppose you have a *biased* coin.
- One side has probability $0.5 + \epsilon$ to appear and the other $0.5 - \epsilon$, for some $0 < \epsilon < 0.5$.
- But you do not know which is which.
- How to decide which side is the more likely side—with high confidence?
- Answer: Flip the coin many times and pick the side that appeared the most times.
- Question: Can you quantify the confidence?

The Chernoff Bound[a]

Theorem 69 (Chernoff (1952)) *Suppose* $x_1, x_2, \ldots, x_n$ *are independent random variables taking the values 1 and 0 with probabilities* p *and* $1-p$*, respectively. Let* $X = \sum_{i=1}^{n} x_i$*. Then for all* $0 \leq \theta \leq 1$*,*

$$\text{prob}[\, X \geq (1+\theta)\, pn \,] \leq e^{-\theta^2 pn/3}.$$

- The probability that the deviate of a **binomial random variable** from its expected value

  $$E[\, X \,] = E\left[\sum_{i=1}^{n} x_i\right] = pn$$

  decreases exponentially with the deviation.

[a]Herman Chernoff (1923–). The bound is asymptotically optimal.

The Proof

- Let t be any positive real number.
- Then
$$\text{prob}[\, X \geq (1+\theta)\, pn \,] = \text{prob}[\, e^{tX} \geq e^{t(1+\theta)\, pn} \,].$$
- Markov's inequality (p. 525) generalized to real-valued random variables says that
$$\text{prob}\left[\, e^{tX} \geq kE[\, e^{tX} \,] \right] \leq 1/k.$$
- With $k = e^{t(1+\theta)\, pn}/E[\, e^{tX} \,]$, we have
$$\text{prob}[\, X \geq (1+\theta)\, pn \,] \leq e^{-t(1+\theta)\, pn} E[\, e^{tX} \,].$$

The Proof (continued)

- Because $X = \sum_{i=1}^{n} x_i$ and x_i's are independent,

$$E[\, e^{tX} \,] = (E[\, e^{tx_1} \,])^n = [\, 1 + p(e^t - 1) \,]^n.$$

- Substituting, we obtain

$$\begin{aligned} \text{prob}[\, X \geq (1+\theta)\, pn \,] &\leq e^{-t(1+\theta)\, pn} [\, 1 + p(e^t - 1) \,]^n \\ &\leq e^{-t(1+\theta)\, pn} e^{pn(e^t - 1)} \end{aligned}$$

  as $(1+a)^n \leq e^{an}$ for all $a > 0$.

The Proof (concluded)

- With the choice of $t = \ln(1+\theta)$, the above becomes

$$\text{prob}[\, X \geq (1+\theta)\, pn \,] \leq e^{pn[\, \theta - (1+\theta) \ln(1+\theta) \,]}.$$

- The exponent expands to

$$-\frac{\theta^2}{2} + \frac{\theta^3}{6} - \frac{\theta^4}{12} + \cdots$$

  for $0 \leq \theta \leq 1$.

- But it is less than

$$-\frac{\theta^2}{2} + \frac{\theta^3}{6} \leq \theta^2 \left(-\frac{1}{2} + \frac{\theta}{6} \right) \leq \theta^2 \left(-\frac{1}{2} + \frac{1}{6} \right) = -\frac{\theta^2}{3}.$$

Power of the Majority Rule

From prob$[\, X \leq (1-\theta)\, pn \,] \leq e^{-\theta^2 pn/2}$ (prove it):

Corollary 70 *If* $p = (1/2) + \epsilon$ *for some* $0 \leq \epsilon \leq 1/2$, *then*

$$\text{prob}\left[\sum_{i=1}^{n} x_i \leq n/2 \right] \leq e^{-\epsilon^2 n/2}.$$

- The textbook's corollary to Lemma 11.9 seems incorrect.[a]
- Our original problem (p. 587) hence demands, e.g., $n \approx 1.4k/\epsilon^2$ independent coin flips to guarantee making an error with probability $\leq 2^{-k}$ with the majority rule.

[a]See Dubhashi and Panconesi (2012) for many Chernoff-type bounds.

BPP[a] (Bounded Probabilistic Polynomial)

- The class **BPP** contains all languages L for which there is a precise polynomial-time NTM N such that:
  - If $x \in L$, then at least $3/4$ of the computation paths of N on x lead to "yes."
  - If $x \notin L$, then at least $3/4$ of the computation paths of N on x lead to "no."
- So N accepts or rejects by a *clear* majority.

[a]Gill (1977).

Magic 3/4?

- The number 3/4 bounds the probability (ratio) of a right answer away from 1/2.
- Any constant *strictly* between 1/2 and 1 can be used without affecting the class BPP.
- In fact, as with RP,

$$\frac{1}{2} + \frac{1}{q(n)}$$

  for any polynomial $q(n)$ can replace 3/4 (p. 582).
- The next algorithm shows why.

The Majority Vote Algorithm

Suppose L is decided by N by majority $(1/2) + \epsilon$.

```
1: for $i = 1, 2, \ldots, 2k + 1$ do
2:     Run $N$ on input $x$;
3: end for
4: if "yes" is the majority answer then
5:     "yes";
6: else
7:     "no";
8: end if
```

Analysis

- The running time remains polynomial: $2k + 1$ times N's running time.
- By Corollary 70 (p. 592), the probability of a false answer is at most $e^{-\epsilon^2 k}$.
- By taking $k = \lceil 2/\epsilon^2 \rceil$, the error probability is at most 1/4.
- Even if ϵ is any inverse polynomial, k remains a polynomial in n.

Aspects of BPP

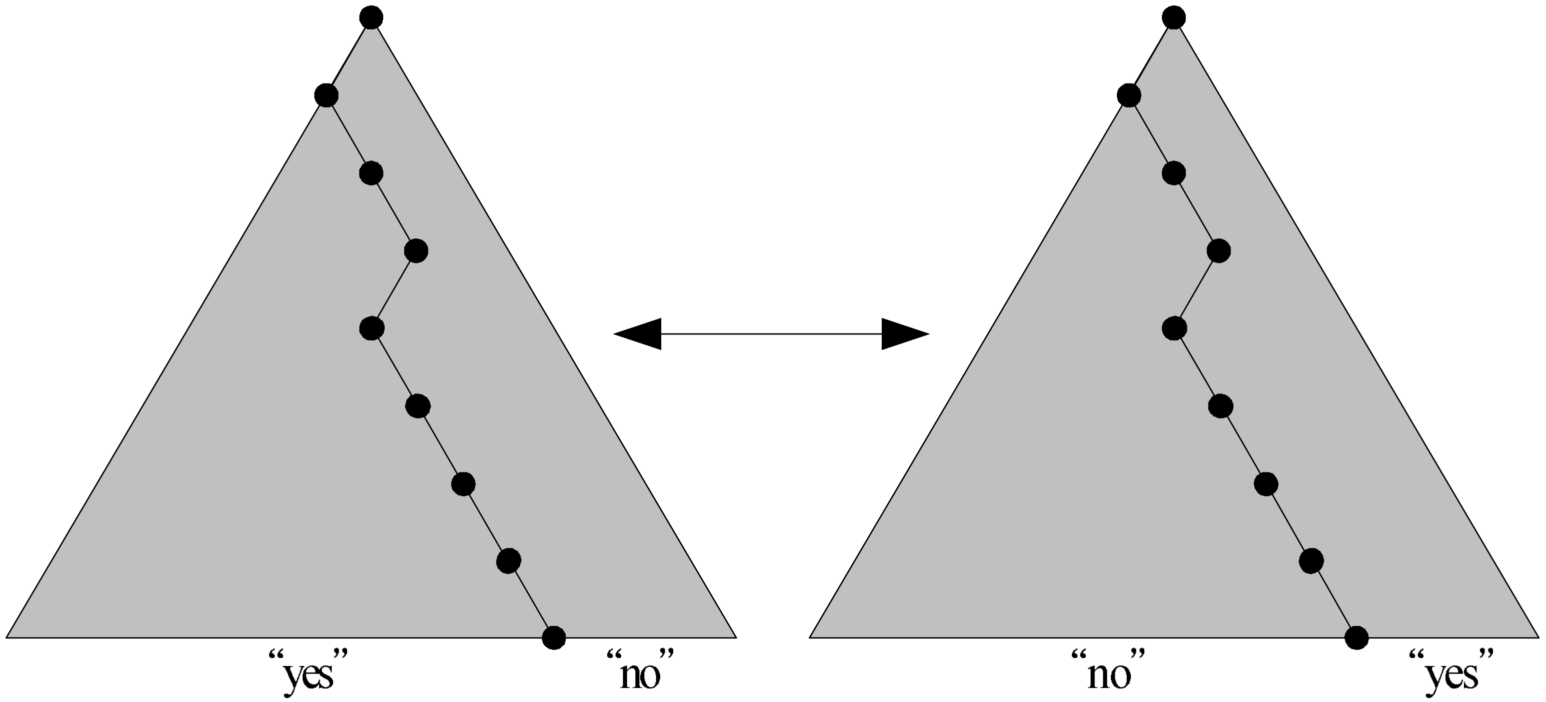
- BPP is the most comprehensive yet plausible notion of efficient computation.
  - If a problem is in BPP, we take it to mean that the problem can be solved efficiently.
  - In this aspect, BPP has effectively replaced P.
- $(\text{RP} \cup \text{coRP}) \subseteq (\text{NP} \cup \text{coNP})$.
- $(\text{RP} \cup \text{coRP}) \subseteq \text{BPP}$.
- Whether $\text{BPP} \subseteq (\text{NP} \cup \text{coNP})$ is unknown.
- But it is unlikely that $\text{NP} \subseteq \text{BPP}$ (see p. 614 and p. 615).

coBPP

- The definition of BPP is symmetric: acceptance by clear majority and rejection by clear majority.
- An algorithm for $L \in \text{BPP}$ becomes one for $\bar{L}$ by reversing the answer.
- So $\bar{L} \in \text{BPP}$ and $\text{BPP} \subseteq \text{coBPP}$.
- Similarly $\text{coBPP} \subseteq \text{BPP}$.
- Hence $\text{BPP} = \text{coBPP}$.
- This approach does not work for RP.[a]

[a]It did not work for NP either.

BPP and coBPP

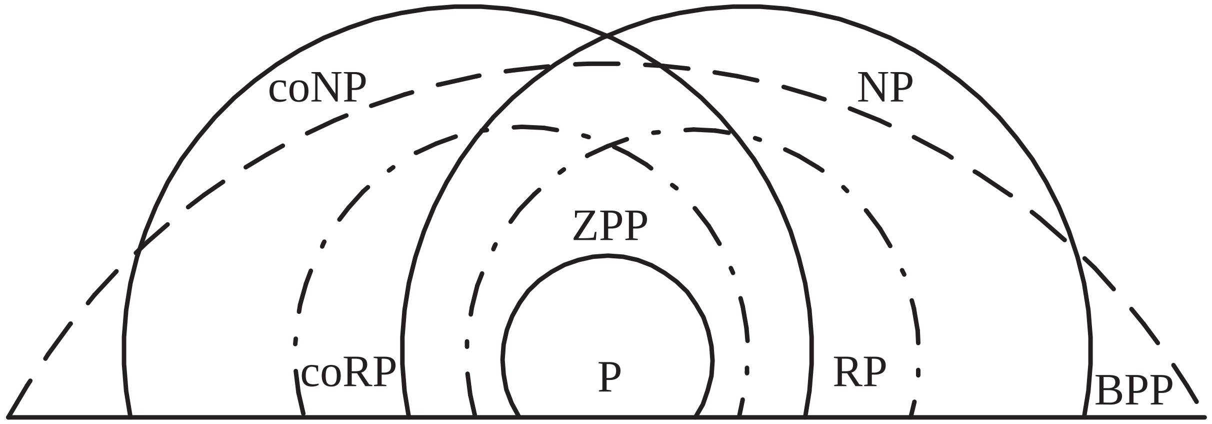


BPP and P

Theorem 71 (Nisan and Wigderson (1994)) *If every language in BPP only needs a pseudorandom generator which stretches a random seed of logarithmic length, then BPP = P.*

- We only need to show BPP $\subseteq$ P.
- Run the BPP algorithm for each of the seeds.
  - There are only $2^{O(\log n)} = O(n^c)$ seeds, a polynomial
- Accept if and only if at least 3/4 of the outcomes is a "yes."
- The running time is clearly deterministically polynomial.

"The Good, the Bad, and the Ugly"



Circuit Complexity

- Circuit complexity is based on boolean circuits instead of Turing machines.
- A boolean circuit with n inputs computes a boolean function of n variables.
- Now, identify `true`/1 with "yes" and `false`/0 with "no."
- Then a boolean circuit with n inputs accepts certain strings in $\{0, 1\}^n$.
- To relate circuits with an arbitrary language, we need one circuit for each possible input length n.

Formal Definitions

- The **size** of a circuit is the number of *gates* in it.
- A **family of circuits** is an infinite sequence $\mathcal{C} = (C_0, C_1, \ldots)$ of boolean circuits, where C_n has n boolean inputs.
- For input $x \in \{0,1\}^*$, $C_{|\,x\,|}$ outputs 1 if and only if $x \in L$.
- In other words,

$$C_n \text{ accepts } L \cap \{0,1\}^n.$$

Formal Definitions (concluded)

- $L \subseteq \{0,1\}^*$ has **polynomial circuits** if there is a family of circuits $\mathcal{C}$ such that:
  - The size of C_n is at most $p(n)$ for some fixed polynomial p.
  - C_n accepts $L \cap \{0,1\}^n$.

Exponential Circuits Suffice for All Languages

- Theorem 16 (p. 211) implies that there are languages that cannot be solved by circuits of size $2^n/(2n)$.
- But exponential circuits can solve *all* problems, decidable or otherwise!

Exponential Circuits Suffice for All Languages (continued)

Proposition 72 *All decision problems (decidable or otherwise) can be solved by a circuit of size* 2^{n+2}.

- We will show that for any language $L \subseteq \{0,1\}^*$, $L \cap \{0,1\}^n$ can be decided by a circuit of size 2^{n+2}.
- Define boolean function $f : \{0,1\}^n \to \{0,1\}$, where

$$f(x_1 x_2 \cdots x_n) = \begin{cases} 1 & x_1 x_2 \cdots x_n \in L, \\ 0 & x_1 x_2 \cdots x_n \notin L. \end{cases}$$

The Proof (concluded)

- Clearly, any circuit that implements f decides $L \cap \{0,1\}^n$.
- Now,

$$f(x_1 x_2 \cdots x_n) = (x_1 \wedge f(1 x_2 \cdots x_n)) \vee (\neg x_1 \wedge f(0 x_2 \cdots x_n)).$$

- The circuit size $s(n)$ for $f(x_1 x_2 \cdots x_n)$ hence satisfies

$$s(n) = 4 + 2s(n-1)$$

with $s(1) = 1$.

- Solve it to obtain $s(n) = 5 \times 2^{n-1} - 4 \leq 2^{n+2}$.

The Circuit Complexity of P

Proposition 73 *All languages in P have polynomial circuits.*

- Let $L \in \mathrm{P}$ be decided by a TM in time $p(n)$.
- By Corollary 33 (p. 312), there is a circuit with $O(p(n)^2)$ gates that accepts $L \cap \{0,1\}^n$.
- The size of the circuit depends only on L and the length of the input.
- The size of the circuit is polynomial in n.

Polynomial Circuits vs. P

- Is the converse of Proposition 73 true?
  - Do polynomial circuits accept only languages in P?
- No.
- Polynomial circuits can accept *undecidable* languages!

Languages That Polynomial Circuits Accept

- Let $L \subseteq \{0,1\}^*$ be an undecidable language.
- Let $U = \{1^n : \text{the binary expansion of } n \text{ is in } L\}$.[a]
  - For example, $11111_1 \in U$ if $101_2 \in L$.
- U is also undecidable (prove it).
- $U \cap \{1\}^n$ can be accepted by the trivial circuit C_n that outputs 1 if $1^n \in U$ and outputs 0 if $1^n \notin U$.[b]
- The family of circuits $(C_0, C_1, \ldots)$ is polynomial in size.

[a] Assume n's leading bit is always 1 without loss of generality.

[b] We may not know which is the case for *general* n.

A Patch

- Despite the simplicity of a circuit, the previous discussions imply the following:
  - Circuits are *not* a realistic model of computation.
  - Polynomial circuits are *not* a plausible notion of efficient computation.
- What is missing?
- The *effective and efficient constructibility* of

$$C_0, C_1, \ldots.$$

Uniformity

- A family $(C_0, C_1, \ldots)$ of circuits is **uniform** if there is a $\log n$-space bounded TM which on input 1^n outputs C_n.
  - Note that n is the length of the input to C_n.
  - Circuits now cannot accept undecidable languages (why?).
  - The circuit family on p. 610 is not constructible by a *single* Turing machine (algorithm).
- A language has **uniformly polynomial circuits** if there is a *uniform* family of polynomial circuits that decide it.

Uniformly Polynomial Circuits and P

Theorem 74 *$L \in P$ if and only if L has uniformly polynomial circuits.*

- One direction was proved in Proposition 73 (p. 608).
- Now suppose L has uniformly polynomial circuits.
- A TM decides $x \in L$ in polynomial time as follows:
  - Calculate $n = |\, x \,|$.
  - Generate C_n in $\log n$ space, hence polynomial time.
  - Evaluate the circuit with input x in polynomial time.
- Therefore $L \in$ P.

Relation to P vs. NP

- Theorem 74 implies that P $\neq$ NP if and only if NP-complete problems have no *uniformly* polynomial circuits.
- A stronger conjecture: NP-complete problems have no polynomial circuits, *uniformly or not.*
- The above is currently the preferred approach to proving the P $\neq$ NP conjecture—without success so far.

BPP's Circuit Complexity

Theorem 75 (Adleman (1978)) *All languages in BPP have polynomial circuits.*

- Our proof will be *nonconstructive* in that only the existence of the desired circuits is shown.
  - Recall our proof of Theorem 16 (p. 211).
  - Something exists if its probability of existence is nonzero.
- It is not known how to efficiently generate circuit C_n.
  - If the construction of C_n can be made efficient, then P = BPP, an unlikely result.
- This result answers the question on p. 520 with a "yes."

The Proof

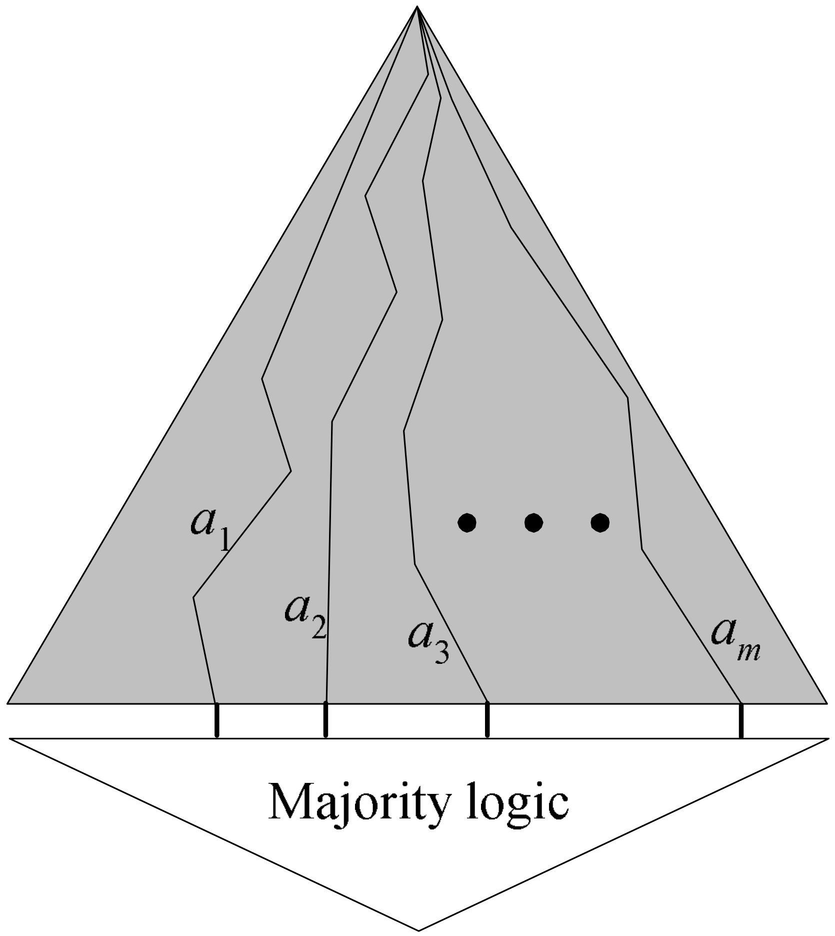
- Let $L \in$ BPP be decided by a precise polynomial-time NTM N by clear majority.
- We shall prove that L has polynomial circuits $C_0, C_1, \ldots$.
  - These *deterministic* circuits cannot make mistakes.
- Suppose N runs in time $p(n)$, where $p(n)$ is a polynomial.
- Let $A_n = \{a_1, a_2, \ldots, a_m\}$, where $a_i \in \{0,1\}^{p(n)}$.
- Each $a_i \in A_n$ represents a sequence of nondeterministic choices (i.e., a computation path) for N.
- Pick $m = 12(n+1)$.

The Proof (continued)

- Let x be an input with $|x| = n$.
- Circuit C_n simulates N on x with each sequence of choices in A_n and then takes the majority of the m outcomes.[a]
- As N with a_i is a polynomial-time deterministic TM, it can be simulated by polynomial circuits of size $O(p(n)^2)$.
  - See the proof of Proposition 73 (p. 608).
- The size of C_n is therefore $O(mp(n)^2) = O(np(n)^2)$.
  - This is a polynomial.

[a]As m is even, there may be no clear majority. Still, the probability of that happening is very small and does not materially affect our general conclusion. Thanks to a lively class discussion on December 14, 2010.

The Circuit



The Proof (continued)

- We now confirm the existence of an A_n making C_n correct on *all* n-bit inputs.
- Call a_i **bad** if it leads N to an error (a false positive or a false negative).
- Select A_n *uniformly randomly.*
- For each $x \in \{0,1\}^n$, $1/4$ of the computations of N are erroneous.
- Because the sequences in A_n are chosen randomly and independently, the expected number of bad a_i's is $m/4$.[a]

[a]So the proof will not work for NP. Contributed by Mr. Ching-Hua Yu (`D00921025`) on December 11, 2012.

The Proof (continued)

- By the Chernoff bound (p. 588), the probability that the number of bad a_i's is $m/2$ or more is at most

$$e^{-m/12} < 2^{-(n+1)}.$$

- The error probability of using majority rule is thus $< 2^{-(n+1)}$ for each $x \in \{0,1\}^n$.
- The probability that there is an x such that A_n results in an incorrect answer is $< 2^n 2^{-(n+1)} = 2^{-1}$.
  - Recall the union bound: $\text{prob}[\, A \cup B \cup \cdots] \leq \text{prob}[\, A\,] + \text{prob}[\, B\,] + \cdots$.
- Note that each A_n yields a circuit.

The Proof (concluded)

- We just showed that at least half of them are correct.
- So with probability ≥ 0.5, a random A_n produces a correct C_n for *all* inputs of length n.
- Because this probability exceeds 0, an A_n that makes majority vote work for all inputs of length n exists.
- Hence a correct C_n exists.[a]
- We have used the **probabilistic method**.[b]

[a]Quine (1948), "To be is to be the value of a bound variable."

[b]The proof is a counting argument phrased in the probabilistic language.

Leonard Adleman[a] (1945–)

[a]Turing Award (2002).

Cryptography

Whoever wishes to keep a secret
must hide the fact that he possesses one.
— Johann Wolfgang von Goethe (1749–1832)

Cryptography

- **Alice** (A) wants to send a message to **Bob** (B) over a channel monitored by **Eve** (eavesdropper).
- The protocol should be such that the message is known only to Alice and Bob.
- The art and science of keeping messages secure is **cryptography**.

$$\begin{tikzcd}\text{Alice} \arrow[r, "\text{Eve}"] & \text{Bob}\end{tikzcd}$$

Encryption and Decryption

- Alice and Bob agree on two algorithms E and D—the **encryption** and the **decryption algorithms**.
- Both E and D are known to the public in the analysis.
- Alice runs E and wants to send a message x to Bob.
- Bob operates D.
- Privacy is assured in terms of two numbers e, d, the **encryption** and **decryption keys**.
- Alice sends $y = E(e, x)$ to Bob, who then performs $D(d, y) = x$ to recover x.
- x is called **plaintext**, and y is called **ciphertext**.[a]

[a]Both “zero” and “cipher” come from the same Arab word.

Some Requirements

- D should be an inverse of E given e and d.
- D and E must both run in (probabilistic) polynomial time.
- Eve should not be able to recover x from y without knowing d.
  - As D is public, d must be kept secret.
  - e may or may not be a secret.

Degrees of Security

- **Perfect secrecy**: After a ciphertext is intercepted by the enemy, the a posteriori probabilities of the plaintext that this ciphertext represents are identical to the a priori probabilities of the same plaintext before the interception.
  - The probability that plaintext $\mathcal{P}$ occurs is independent of the ciphertext $\mathcal{C}$ being observed.
  - So knowing $\mathcal{C}$ yields no advantage in recovering $\mathcal{P}$.
- Such systems are said to be **informationally secure**.
- A system is **computationally secure** if breaking it is theoretically possible but computationally infeasible.

Conditions for Perfect Secrecy[a]

- Consider a cryptosystem where:
  - The space of ciphertext is as large as that of keys.
  - Every plaintext has a nonzero probability of being used.
- It is perfectly secure if and only if the following hold.
  - A key is chosen with uniform distribution.
  - For each plaintext x and ciphertext y, there exists a unique key e such that $E(e, x) = y$.

[a]Shannon (1949).

The One-Time Pad[a]

```
1: Alice generates a random string $r$ as long as $x$;
2: Alice sends $r$ to Bob over a secret channel;
3: Alice sends $x \oplus r$ to Bob over a public channel;
4: Bob receives $y$;
5: Bob recovers $x := y \oplus r$;
```

[a]Mauborgne and Vernam (1917); Shannon (1949). It was allegedly used for the hotline between Russia and U.S.

Analysis

- The one-time pad uses $e = d = r$.
- This is said to be a **private-key cryptosystem**.
- Knowing x and knowing r are equivalent.
- Because r is random and private, the one-time pad achieves perfect secrecy (see also p. 629).
- The random bit string must be new for each round of communication.
  - **Cryptographically strong pseudorandom generators** require exchanging only the seed once.
- But the assumption of a private channel is problematic.