Cryptocomplexity II

Kryptokomplexität II

Sommersemester 2024

Chapter 8: Randomized Algorithms and Complexity Classes

Dozent: Prof. Dr. J. Rothe

Outline of this Chapter

- PP, RP, and ZPP: Monte Carlo and Las Vegas Algorithms
- BPP: Bounded-Error Probabilistic Polynomial Time
- Quantifiers and BPP
- Graph Isomorphism and the Arthur-Merlin Hierarchy

Probabilistic Turing Machines: Syntax

- The probabilistic Turing machine model can be described by nondeterministic Turing machines (NTMs).
- Syntactically, a *probabilistic Turing machine* simply is an NTM N.
- **Convention:**
  - The nondeterministic branching degree of N is at most two.
  - We require that NTMs be *normalized*, i.e., for each input x, all computations paths in $N(x)$ have the same number of nondeterministic branching points and can thus be described by binary strings of the same length.

  In other words, normalized machines always have a *full binary computation tree* (not allowing for the deterministic steps of the machine).

Probabilistic Turing Machines: Semantics

- To define the semantics of probabilistic Turing machines, we specify an acceptance behavior of NTMs suitable for randomization.
- To this end, for any given NTM N and any input x, we define a *probability measure μ_T on the set of computation paths in the tree $N(x)$*, whose vertices represent the configurations of N on input x.
- Note that there is a one-to-one correspondence between the computation paths and the leaves in the tree $N(x)$.

Probabilistic Turing Machines: Semantics

Definition

For any NTM N and any input x, consider any subtree T of the computation tree $N(x)$ such that the root of T is the root of $N(x)$.
The probability measure μ_T on the set of leaves of T is defined inductively by:

- Initially, if T consists of only the root r of $N(x)$ (i.e., r is the start configuration of $N(x)$), then set $\mu_T(r) = 1$.
- While $T \neq N(x)$, fix some leaf ℓ of T that is not a leaf of $N(x)$, and consider the new subtree T_ℓ of $N(x)$ that is obtained from T by adding the immediate successor configuration(s) of ℓ. From the probability measure μ_T on the leaves of T, define the probability measure on the leaves of T_ℓ by:

$$\mu_{T_\ell}(c) = \begin{cases} \mu_T(\ell)/2 & \text{if } c \text{ is one out of two successor configurations of } \ell \\ \mu_T(\ell) & \text{if } c \text{ is the only successor configuration of } \ell \\ \mu_T(c) & \text{if } c \text{ is not a successor configuration of } \ell. \end{cases}$$

Probabilistic Turing Machines

- We consider only polynomial-time probabilistic Turing machines.
- Every computation path of a given NPTM N running on some input x is represented by a binary string α of length $p(|x|)$ for some $p \in \mathbb{P}\mathrm{ol}$, where the i^{th} bit of α corresponds to the i^{th} nondeterministic branching of $N(x)$ along α.
- If α leads to some leaf ℓ of $T = N(x)$, we write $\mu_T(\alpha) = \mu_T(\ell)$.
- Note that μ_T indeed is a probability measure, since for each finite tree T:

$$\sum_{\alpha \text{ is some path of } T} \mu_T(\alpha) \quad = \quad 1.$$

Probabilistic Turing Machines

- Every subset of paths of T is an event. The probability that some event E occurs with respect to μ_T is given by:

$$\Pr(E) = \sum_{\alpha \in E} \mu_T(\alpha) = \sum_{\alpha \in E} 2^{-|\alpha|}.$$

- Using the above notation, we can now define the first two probabilistic complexity classes:
  - the class PP, "probabilistic polynomial time," and
  - the class RP, "random polynomial time."

Probabilistic Polynomial Time & Random Polynomial Time

Definition

1. *Probabilistic polynomial time* is defined by

$$\mathrm{PP} = \left\{ A \;\middle|\; \begin{array}{l} \text{there is some NPTM } M \text{ such that for each input } x, \\ x \in A \iff \Pr(\{\alpha \,|\, M \text{ accepts } x \text{ on path } \alpha\}) \geq \frac{1}{2} \end{array} \right\}.$$

2. *Random polynomial time* is defined by

$$\mathrm{RP} = \left\{ A \;\middle|\; \begin{array}{l} \text{there is some NPTM } M \text{ such that for each input } x, \\ x \in A \implies \Pr(\{\alpha \,|\, M \text{ accepts } x \text{ on path } \alpha\}) \geq \frac{1}{2}; \\ x \notin A \implies \Pr(\{\alpha \,|\, M \text{ accepts } x \text{ on path } \alpha\}) = 0 \end{array} \right\}.$$

Normalized Turing Machines and Threshold Computation

Remark:

1. Every NPTM whose acceptance criterion is based on probability weights can easily be normalized:
   - Just extend every computation path up to a fixed polynomial length by appending a full binary subtree to it;
   - on each path thus obtained,
     accept if and only if the original path was accepting.

   The modified normalized machine has the same acceptance probability as the original machine.

2. The acceptance criterion of the probabilistic machines for PP and RP is determined by the probability weight of accepting paths.

Normalized Turing Machines and Threshold Computation

Alternatively, the acceptance criterion for PP and RP can be based on the number of accepting paths of NPTMs. For any NPTM M, define the function $\mathrm{acc}_M : \Sigma^* \to \mathbb{N}$ by $\mathrm{acc}_M(x) = \|\{\alpha \,\big|\, M \text{ accepts } x \text{ on path } \alpha\}\|$. Then, the following characterizations of PP and RP can be shown:

1. A is in PP if and only if there exists some normalized NPTM M and some polynomial p such that for each x:

$$x \in A \iff \mathrm{acc}_M(x) \geq 2^{p(|x|)-1}.$$

2. A is in RP if and only if there exists some normalized NPTM M and some polynomial p such that for each x:

$$x \in A \implies \mathrm{acc}_M(x) \geq 2^{p(|x|)-1};$$
$$x \notin A \implies \mathrm{acc}_M(x) = 0.$$

RP is a Promise Class

- Observe that this equivalence of the probability weight interpretation and the acc_M interpretation uses normalized machines.
- In fact, for PP, the normalization requirement may be dropped.
- However, for RP, having normalized machines appears to be a crucial requirement. Unlike PP, the class RP is a so-called *"promise class"*:
  - RP has a rejection criterion that is *more restrictive* than the logical negation of the acceptance criterion, which leaves open the possibility that for some inputs none of the two criteria applies.
  - The burdon to avoid this obstacle is shouldered by the machines representing the promise class: Every machine *"promises"* that for each input exactly one of the two criteria holds.
  - Promise classes appear to have different properties than non-promise complexity classes (e.g., lack of complete sets).

RP and Monte Carlo Algorithms

- RP and coRP algorithms both have a *one-sided error probability*. Such algorithms are also called *Monte Carlo algorithms*.

- Let L be a set in RP, and let A be an RP algorithm for L. By definition,
  - A may make errors for instances x in L, but
  - A never lies for instances x not in L.

- Thus,
  - the answer "yes" (i.e., an accepting path) of A on input x can occur only if x is in L and is thus always correct,
  - whereas the answer "no" (i.e., a rejecting path) can occur both erroneously (if x is in L) and correctly (if x is not in L).

RP and Monte Carlo Algorithms

- Therefore, RP algorithms are sometimes called *no-biased Monte Carlo algorithms*.
- Similarly, coRP algorithms always give reliable "no" answers, but perhaps erroneous "yes" answers.
- Therefore, coRP algorithms are sometimes called *yes-biased Monte Carlo algorithms*.
- Example: $\textsc{Primes}$ is in coRP, since the Miller–Rabin test is a yes-biased Monte Carlo algorithm for $\textsc{Primes}$.
- Unlike PP, the class RP seems to be not closed under complementation (unless, of course, it turns out that $\mathrm{RP} = \mathrm{P}$).

Inclusion Relations and Properties of PP and RP

Theorem

1. $\mathrm{P} \subseteq \mathrm{RP} \subseteq \mathrm{NP} \subseteq \mathrm{PP} \subseteq \mathrm{PSPACE}$.
2. PP *is closed under complementation.*

Proof:

1. The first two inclusions, $\mathrm{P} \subseteq \mathrm{RP} \subseteq \mathrm{NP}$, follow immediately from the definitions.

   The inclusion $\mathrm{PP} \subseteq \mathrm{PSPACE}$ can be proven similar to the inclusion $\mathrm{NP} \subseteq \mathrm{PSPACE}$: Given a PP machine M running on input x, the simulating PSPACE machine performs a depth-first search through the computation tree of $M(x)$.

Inclusion Relations and Properties of PP and RP

However, rather than searching for *some* accepting path as in the proof of $\mathrm{NP} \subseteq \mathrm{PSPACE}$, the PSPACE machine now counts *all* accepting paths of $M(x)$, and it accepts the input if and only if this number is at least half of the total number of paths.

For the inclusion $\mathrm{NP} \subseteq \mathrm{PP}$, let A be any set in NP, and let M be a given NP machine accepting A.

Suppose that M is normalized so that on each input x, the computation tree $M(x)$ is a complete binary tree of depth $p(|x|)$ for some $p \in \mathbb{P}\mathrm{ol}$.

Construct a new NPTM N accepting A in the sense of PP as follows.

On input x, N branches nondeterministically.

Inclusion Relations and Properties of PP and RP

On the left branch, N simulates the computation of $M(x)$.

On the right branch, N creates a complete binary tree of depth $p(|x|)$, accepts on $2^{p(|x|)} - 1$ of its paths, and rejects on one of its paths.
If $x \in A$, then $M(x)$ accepts on at least one of its $2^{p(|x|)}$ paths.

Thus, $N(x)$ accepts on at least $2^{p(|x|)}$ of its $2^{p(|x|)+1}$ paths, i.e., N accepts x with probability at least one half.

If $x \notin A$, then all $2^{p(|x|)}$ paths of $M(x)$ are rejecting.

Thus, $N(x)$ accepts on at most $2^{p(|x|)} - 1$ of its $2^{p(|x|)+1}$ paths, i.e., N accepts x with probability less than one half.

It follows that A is in PP.

2. Exercise. ❑

Changing the Acceptance Threshold of $1/2$ in RP

Definition

Let q be a fixed polynomial. Define the class

$$\mathrm{RP}_q = \left\{ A \;\middle|\; \begin{array}{l} \text{there is some NPTM } M \text{ such that for each input } x, \\ x \in A \Longrightarrow \Pr(\{\alpha \mid M \text{ accepts } x \text{ on path } \alpha\}) \geq \frac{1}{q(|x|)}; \\ x \notin A \Longrightarrow \Pr(\{\alpha \mid M \text{ accepts } x \text{ on path } \alpha\}) = 0 \end{array} \right\}.$$

Theorem

Let q be a nondecreasing polynomial such that for each n, $q(n) \geq 2$.

Then, $\mathrm{RP}_q = \mathrm{RP}$.

Changing the Acceptance Threshold of $1/2$ in RP

Proof: Since $q(n) \geq 2$ for each n, $\mathrm{RP} \subseteq \mathrm{RP}_q$ holds by definition.

Conversely, to prove $\mathrm{RP}_q \subseteq \mathrm{RP}$, let A be any set in RP_q for some fixed polynomial q, and let M be some NPTM for A witnessing $A \in \mathrm{RP}_q$.

Construct an NPTM N that accepts A in the sense of RP as follows.

On input x of length n, N successively simulates the computation of $M(x)$ in $q = q(n)$ independent trials.

Thus, every path $\alpha = \alpha_1 \alpha_2 \cdots \alpha_q$ of $N(x)$ consists of a sequence of q paths α_i of $M(x)$, and α is defined to accept if and only if at least one of its subpaths α_i accepts.

Since q is a polynomial and M runs in polynomial time, so does N.

Changing the Acceptance Threshold of $1/2$ in RP

It remains to show that N witnesses membership of A in RP.

Let x be the given input string.

If $x \notin A$, no path of $M(x)$ accepts.

Thus, $N(x)$ has no accepting paths and the acceptance probability is zero.

We now estimate the error probability $E_N(x)$ of N for $x \in A$:

$$E_N(x) \quad = \quad \Pr(\{\alpha \,\big|\, N \text{ rejects } x \text{ on path } \alpha\}).$$

The error probability of $M(x)$ is bounded above by $1 - \frac{1}{q}$.

For each path α of $N(x)$, all subpaths α_i of α are independently chosen.

Changing the Acceptance Threshold of 1/2 in RP

This implies for the error probability of N that

$$E_N(x) < \left(1 - \frac{1}{q(n)}\right)^{q(n)} < \frac{1}{2}. \tag{1}$$

The latter inequality of (1) follows from $\lim_{k\to\infty}(1+\frac{a}{k})^k = \mathrm{e}^a$, where $\mathrm{e} = 2.71828\cdots$ is the base of the natural logarithm.

Thus, for $a = -1$, we have that $\left(1 - \frac{1}{q(n)}\right)^{q(n)}$ is close to e^{-1}, which is less than one half.

Hence,

$$\Pr(\{\alpha \,|\, N \text{ accepts } x \text{ on path } \alpha\}) \geq \frac{1}{2},$$

which completes the proof. ❑

Closure Properties of RP and PP

Corollary

RP *is closed under union and intersection.*

Proof: Just think a little about it . . . ❑

Theorem (Beigel, Reingold, and Spielman (1991))

PP *is closed under union and intersection.* **without proof**

Remark: Whereas closure under complementation is easy to see for PP, the proof of closure under intersection is not at all trivial. Check it out!

Zero-Error Probabilistic Polynomial Time

- RP algorithms have a one-sided error, and this error can be made very small. This is a clear advantage of RP over PP algorithms.
- RP algorithms can give false "no" answers, whereas coRP algorithms always give reliable "no" answers.
- On the other hand, coRP algorithms can give false "yes" answers, whereas RP algorithms always give reliable "yes" answers.
- The class ZPP collects all problems solvable by polynomial-time randomized algorithms with *zero error probability*, combining the advantages of yes-biased and no-biased Monte Carlo algorithms.

Zero-Error Probabilistic Polynomial Time

Definition

Define the class *zero-error probabilistic polynomial time* by

$$\mathrm{ZPP} = \mathrm{RP} \cap \mathrm{coRP}.$$

Remark:

- Just like RP, the class ZPP is a promise class.
- ZPP algorithms, which are also dubbed *Las Vegas algorithms*, never give a wrong answer, although it might happen that they do not give any useful answer at all.
- This justifies the name *"zero-error" probabilistic polynomial time*.

Zero-Error Probabilistic Polynomial Time

- A ZPP algorithm can be viewed as an NPTM M with three types of final states:
  - an accepting state, s_a,
  - a rejecting state, s_r, and
  - a "don't know" state, $s_?$.

- Let A be any language in ZPP, and let M and N be NPTMs witnessing that $A \in \mathrm{RP}$ and $\overline{A} \in \mathrm{RP}$, respectively.

	path α of $M(x)$	path β of $N(x)$	path $\langle \alpha, \beta \rangle$ of $(M \circ N)(x)$
$x \in A$	$+$	$-$	$(+,-) = s_a$
	$-$	$-$	$(-,-) = s_?$
$x \notin A$	$-$	$+$	$(-,+) = s_r$
	$-$	$-$	$(-,-) = s_?$

Table: A ZPP-computation

Zero-Error Probabilistic Polynomial Time

Corollary

A is in ZPP *if and only if*

there exists an NPTM M with three types of final states:

- *an accepting state,* s_a,
- *a rejecting state,* s_r, *and*
- *a "don't know" state,* $s_?$

such that for each x,

$$
\begin{aligned}
x \in A \quad \Longrightarrow \quad & \Pr(\{\alpha \mid M \text{ accepts } x \text{ on path } \alpha\}) \geq 1/2 \text{ and} \\
& \Pr(\{\alpha \mid M \text{ rejects } x \text{ on path } \alpha\}) = 0; \\
x \notin A \quad \Longrightarrow \quad & \Pr(\{\alpha \mid M \text{ rejects } x \text{ on path } \alpha\}) \geq 1/2 \text{ and} \\
& \Pr(\{\alpha \mid M \text{ accepts } x \text{ on path } \alpha\}) = 0.
\end{aligned}
$$

Majority Satisfiability and Threshold Satisfiability

Definition

$$\textsc{Majority-SAT} = \left\{ \varphi \;\middle|\; \begin{array}{l} \varphi \text{ is a boolean formula with } n \text{ variables} \\ \text{and at least } 2^{n-1} \text{ satisfying assignments} \end{array} \right\};$$

$$\textsc{Threshold-SAT} = \left\{ \langle \varphi, i \rangle \;\middle|\; \begin{array}{l} \varphi \text{ is a boolean formula with at} \\ \text{least } i \text{ satisfying assignments} \end{array} \right\}.$$

Theorem

Both $\textsc{Majority-SAT}$ *and* $\textsc{Threshold-SAT}$ *are* $\leq_{\mathrm{m}}^{\mathrm{p}}$*-complete for* PP.

Majority Satisfiability and Threshold Satisfiability

Proof: It is easy to see that MAJORITY-SAT belongs to PP.

We now prove that

1. THRESHOLD-SAT is $\leq_{\mathrm{m}}^{\mathrm{p}}$-hard for PP, and
2. THRESHOLD-SAT $\leq_{\mathrm{m}}^{\mathrm{p}}$ MAJORITY-SAT.

Since PP is closed under $\leq_{\mathrm{m}}^{\mathrm{p}}$-reductions (see exercises), both statements of the theorem follow.

1. THRESHOLD-SAT is $\leq_{\mathrm{m}}^{\mathrm{p}}$-hard for PP:
   Let A be any set in PP, and let M be some NPTM accepting A in the sense of PP. We may assume that M is a normalized machine and there is a polynomial p such that for each x of length n:

$$x \in A \iff \mathrm{acc}_M(x) \geq 2^{p(n)-1}.$$

Majority Satisfiability and Threshold Satisfiability

Let f_M be the Cook reduction and let $\varphi_{M,x} = f_M(x)$ be the corresponding boolean formula.

Since the Cook reduction is "parsimonious,"

$$\mathrm{acc}_M(x) = \|\{\beta \,\big|\, \beta \text{ is a satisfying assignment for } \varphi_{M,x}\}\|.$$

Thus, the reduction

$$g(x) = \langle \varphi_{M,x}, 2^{p(|x|)-1} \rangle$$

shows that $A \leq_{\mathrm{m}}^{\mathrm{p}} \textsc{Threshold-SAT}$.

Majority Satisfiability and Threshold Satisfiability

2. $\text{THRESHOLD-SAT} \leq_m^p \text{MAJORITY-SAT}$:

   Let $\langle \varphi, i \rangle$ be any given THRESHOLD-SAT instance, where φ is a boolean formula in the variables $x_1, x_2, \ldots, x_m$.

   Construct a formula $\psi = \psi(x_1, x_2, \ldots, x_m)$ such that ψ has *exactly* $j = 2^m - i$ satisfying assignments, where we assume that $i \leq 2^m$.

   Consider the binary representation of

   $$j = 2^{m-s_1} + 2^{m-s_2} + \cdots + 2^{m-s_k},$$

   where $0 \leq s_1 < s_2 < \cdots < s_k \leq m$.

   For example, if $m = 3$,

   $$j = 7 = 2^{3-1} + 2^{3-2} + 2^{3-3}.$$

Majority Satisfiability and Threshold Satisfiability

Define

$$\begin{aligned}\psi \quad = \quad & (x_1 \wedge \cdots \wedge x_{s_1-1} \wedge x_{s_1}) \vee \\ & (x_1 \wedge \cdots \wedge x_{s_1-1} \wedge \neg x_{s_1} \wedge x_{s_1+1} \wedge \cdots \wedge x_{s_2-1} \wedge x_{s_2}) \vee \\ & \qquad \vdots \\ & (x_1 \wedge \cdots \wedge x_{s_1-1} \wedge \neg x_{s_1} \wedge x_{s_1+1} \wedge \cdots \wedge x_{s_2-1} \wedge \neg x_{s_2} \wedge x_{s_2+1} \\ & \wedge \cdots \wedge x_{s_{k-1}-1} \wedge \neg x_{s_{k-1}} \wedge x_{s_{k-1}+1} \wedge \cdots \wedge x_{s_k}).\end{aligned}$$

Observe that

- the ℓ^{th} *implicant in* ψ (i.e., a term in ψ that, if true, makes the formula ψ true) contributes exactly 2^{m-s_ℓ} satisfying assignments, and
- due to the negations in ψ no assignment satisfying one implicant can also satisfy another. Thus, none of the satisfying assignments is counted twice.

Majority Satisfiability and Threshold Satisfiability

Hence, the number of assignments satisfying ψ adds up to:

$$2^{m-s_1} + 2^{m-s_2} + \cdots + 2^{m-s_k} = j$$

as desired.

Now, fix some formula $\gamma \notin \text{MAJORITY-SAT}$; for example, pick

$$\gamma = x \wedge y.$$

Define the reduction $\text{THRESHOLD-SAT} \leq_{\text{m}}^{\text{p}} \text{MAJORITY-SAT}$ by

$$f(\langle \varphi, i \rangle) = \begin{cases} \gamma & \text{if } i > 2^m \\ (x_0 \wedge \varphi(x_1, \ldots, x_m)) \vee (\neg x_0 \wedge \psi(x_1, \ldots, x_m)) & \text{if } i \leq 2^m. \end{cases}$$

Majority Satisfiability and Threshold Satisfiability

It is easy to see that

- if $i > 2^m$, then
  - both $\langle \varphi, i \rangle \notin \text{THRESHOLD-SAT}$
  - and $f(\langle \varphi, i \rangle) = \gamma \notin \text{MAJORITY-SAT}$.
- On the other hand, if $i \leq 2^m$, then

$\langle \varphi, i \rangle \in \text{THRESHOLD-SAT} \iff$ $f(\langle \varphi, i \rangle)$ is satisfied by at least $i + 2^m - i = 2^m$ out of the 2^{m+1} possible assignments. ❑

Bounded-Error Probabilistic Polynomial Time

- The acceptance criterion for PP machines is not very robust: Adding or deleting just one accepting path may result in a different outcome with regard to accepting or rejecting the input.
- In other words, as the input size grows to infinity, the error probability asymptotically can go to one half.
- Our goal now is to bound the error probability away from one half. To this end, we introduce the complexity class BPP.
- In a BPP computation either accepts or rejects its input with high probability, leaving a proper gap around the value of one half that is strictly avoided by the acceptance or rejection probability.
- That is why BPP is a promise class as well.

Bounded-Error Probabilistic Polynomial Time

Definition

The class *bounded-error probabilistic polynomial time* is defined by

$$\mathrm{BPP} = \left\{ A \;\middle|\; \begin{array}{l} \text{there is some NPTM } M \text{ and a constant } c,\ 0 < c \leq 1/2, \\ \text{such that for each input } x, \\ x \in A \Longrightarrow \Pr(\{\alpha \,|\, M \text{ accepts } x \text{ on path } \alpha\}) \geq \frac{1}{2} + c; \\ x \notin A \Longrightarrow \Pr(\{\alpha \,|\, M \text{ accepts } x \text{ on path } \alpha\}) \leq \frac{1}{2} - c \end{array} \right\}.$$

Bounded-Error Probabilistic Polynomial Time

- Let r be a function from $\mathbb{N}$ to the real interval $[0,1]$.
- We say that *an NPTM M accepts a set A in the sense of* BPP *with error probability at most r* if and only if

$$\Pr(\{\alpha \mid M(x) = \chi_A(x) \text{ on path } \alpha\}) \geq 1 - r(|x|),$$

  where χ_A denotes the characteristic function of A.
- We now show that the error probability of BPP computations can be made exponentially small in the input size. Thus, the error in such a BPP computation is "negligibly small."

Bounded-Error Probabilistic Polynomial Time

Theorem

Let p be some fixed polynomial, and let A be any set in BPP*. Then, there exists an NPTM N accepting A in the sense of* BPP *with error probability at most $2^{-p(n)}$.*

Proof: Fix a polynomial p. Given any set A in BPP, let M be some NPTM and c with $0 < c \leq 1/2$ be some constant such that:

$$\Pr(\{\alpha \mid M(x) = \chi_A(x) \text{ on path } \alpha\}) \geq \frac{1}{2} + c.$$

For any given input x of length n and for some polynomial q to be specified below, set $k = 2q(n) + 1$.

As in the proof of $\mathrm{RP}_q \subseteq \mathrm{RP}$, construct an NPTM N that, on input x, simulates the computation of $M(x)$ in k successive, independent trials.

Bounded-Error Probabilistic Polynomial Time

Thus, every path $\alpha = \alpha_1 \alpha_2 \cdots \alpha_k$ of $N(x)$ consists of a sequence of k paths α_i of $M(x)$.

However, now we define α to accept if and only if a majority (i.e., at least $q(n)+1$) of the paths α_i of $M(x)$ along α accept.

We have to show that we can find a polynomial q such that the error probability of $N(x)$, which is given by

$$E_N(x) \quad = \quad \Pr(\{\alpha \,\big|\, N(x) \neq \chi_A(x) \text{ on path } \alpha\})$$

is bounded above by $2^{-p(n)}$.

Bounded-Error Probabilistic Polynomial Time

By the way acceptance of a path $\alpha = \alpha_1\alpha_2\cdots\alpha_k$ is defined, for α to be such an erroneous computation, there must exist some j satisfying that:

- $j \leq q(n)$,
- j subpaths α_i along α are correct, i.e., $M(x) = \chi_A(x)$, and
- the remaining $k-j$ subpaths α_i along α are incorrect, i.e., $M(x) \neq \chi_A(x)$.

Denote

- the success probability of $M(x)$ by $\sigma = \frac{1}{2} + c$ and
- the error probability of $M(x)$ by $\varepsilon = \frac{1}{2} - c$.

Bounded-Error Probabilistic Polynomial Time

Choosing among the k possible subpaths α_i of α the j correct ones and summing over all possible j, we can estimate the error probability of $N(x)$ as follows:

$$E_N(x) \leq \sum_{j=0}^{q(n)} \binom{k}{j} \sigma^j \varepsilon^{k-j}. \tag{2}$$

Let $m > 0$ be chosen such that

$$j = \frac{k}{2} - m \quad \text{and} \quad k - j = \frac{k}{2} + m.$$

Since $\varepsilon < \sigma$, it follows that:

$$\sigma^j \varepsilon^{k-j} = (\sigma \cdot \varepsilon)^{\frac{k}{2}} \cdot \sigma^{-m} \cdot \varepsilon^m = (\sigma \cdot \varepsilon)^{\frac{k}{2}} \cdot \left(\frac{\varepsilon}{\sigma}\right)^m < (\sigma \cdot \varepsilon)^{\frac{k}{2}}. \tag{3}$$

Bounded-Error Probabilistic Polynomial Time

The Binomial Theorem, which says that

$$(a+b)^k = \sum_{j=0}^{k} \binom{k}{j} a^j b^{k-j},$$

implies for the special case of $a = b = 1$ that:

$$\sum_{j=0}^{k} \binom{k}{j} = 2^k. \tag{4}$$

Substituting (3) and (4) in (2) gives:

$$\begin{aligned} E_N(x) &< (\sigma \cdot \varepsilon)^{\frac{k}{2}} \cdot 2^k \\ &= (4\sigma\varepsilon)^{\frac{k}{2}} \\ &= \left(1-4c^2\right)^{\frac{k}{2}}, \quad \text{since } \sigma\varepsilon = (\tfrac{1}{2}+c)(\tfrac{1}{2}-c) = \tfrac{1}{4}-c^2 \\ &\leq \left(1-4c^2\right)^{q(n)}, \end{aligned}$$

Bounded-Error Probabilistic Polynomial Time

where the latter inequality follows from

$$1-4c^2<1 \quad \text{and} \quad \frac{k}{2}=q(n)+\frac{1}{2}>q(n).$$

Since $1-4c^2<1$, we have $\left(1-4c^2\right)^t \leq 1/2$ for some integer t.

Now, setting $q(n)=t\cdot p(n)$ gives

$$E_N(x) \quad \leq \quad \left(1-4c^2\right)^{t\cdot p(n)} \leq \left(\frac{1}{2}\right)^{p(n)} \leq 2^{-p(n)},$$

as desired. ❑

Fact

$\mathrm{RP} \subseteq \mathrm{BPP} = \mathrm{coBPP} \subseteq \mathrm{PP}$.

Bounded-Error Probabilistic Polynomial Time

Theorem

If $\mathrm{NP} \subseteq \mathrm{BPP}$ *then* $\mathrm{NP} = \mathrm{RP}$.

Proof: Suppose that $\mathrm{NP} \subseteq \mathrm{BPP}$.

By the previous theorem, there is some NPTM M accepting SAT in the sense of BPP with error probability at most 2^{-n}:

$$\Pr(\{\alpha \mid M(\varphi) = \chi_{\mathrm{SAT}}(\varphi) \text{ on path } \alpha\}) \quad \geq \quad 1 - 2^{-n}, \tag{5}$$

where $n = |\varphi|$ is the length of the boolean formula φ in some suitable encoding.

Goal: to show that $\mathrm{SAT} \in \mathrm{RP}$ by constructing an RP machine for SAT.

Since SAT is $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete in NP and since RP is $\leq_{\mathrm{m}}^{\mathrm{p}}$-closed, $\mathrm{NP} = \mathrm{RP}$ follows.

Bounded-Error Probabilistic Polynomial Time

For any given boolean formula $\varphi = \varphi(x_1, x_2, \ldots, x_m)$ and for any bit string $s \in \{0,1\}^*$, $|s| \leq m$, define the formula φ_s in $m - |s|$ variables that is obtained from φ by substituting the i^{th} bit of s as the truth value of the i^{th} variable in φ:

$$\begin{aligned}
\varphi_0(x_2, x_3, \ldots x_m) &= \varphi(0, x_2, x_3, \ldots, x_m) \\
\varphi_1(x_2, x_3, \ldots x_m) &= \varphi(1, x_2, x_3, \ldots, x_m) \\
\varphi_{00}(x_3, x_4, \ldots x_m) &= \varphi(0, 0, x_3, x_4, \ldots, x_m) \\
&\vdots
\end{aligned}$$

Depending on the encoding used, simplifying φ to some formula φ_s may result in a shorter encoding string, so $|\varphi_s| \leq |\varphi|$.

Bounded-Error Probabilistic Polynomial Time

However, since the error probability of M depends on the input size, we want to make sure that $|\varphi_s| \geq |\varphi|$.

To this end, we pad φ_s with a sufficient number of new variables $v_1, v_2, \ldots, v_{k(s)}$.

For each φ_s, where $s \in \{0,1\}^*$ and $|s| \leq m$, define the padded formula ψ_s in the variables $x_{m-|s|+1}, \ldots, x_m, v_1, \ldots, v_{k(s)}$ by

$$\psi_s \quad = \quad \varphi_s(x_{m-|s|+1}, \ldots, x_m) \wedge v_1 \wedge \ldots \wedge v_{k(s)},$$

where $k(s)$ is chosen large enough to ensure $|\psi_s| \geq |\varphi|$.

Note that ψ_s is satisfiable if and only if φ_s is satisfiable.

Bounded-Error Probabilistic Polynomial Time

To show that $\mathrm{SAT} \in \mathbf{RP}$, we describe an NPTM N accepting SAT in the sense of $\mathbf{RP}$.

On input $\varphi(x_1, x_2, \ldots, x_m)$ of length n, N seeks to find a satisfying assignment for φ, if one exists.

To this end, N first nondeterministically branches and uses M to construct, step by step, candidates of satisfying assignments for φ on each of its nondeterministic computation paths.

Then, on each such path, N verifies deterministically whether or not the candidate constructed indeed satisfies φ.

Bounded-Error Probabilistic Polynomial Time

In more detail, N on input $\varphi(x_1, x_2, \ldots, x_m)$ works as follows:

Step 1: Simulate $M(\psi_0)$. Since $|\psi_0| \geq |\varphi| = n$, Equation (5) implies that:

$$\begin{aligned} \Pr(\{\alpha \,|\, M(\psi_0) = \chi_{\mathrm{SAT}}(\psi_0) \text{ on path } \alpha\}) &\geq 1 - 2^{-|\psi_0|} \\ &\geq 1 - 2^{-n}. \end{aligned} \tag{6}$$

- On the accepting paths α of $M(\psi_0)$, N stores the assignment 0 for the variable x_1 by setting the first bit of s_α to 0, and continues recursively by simulating $M(\psi_{00})$.
- On the rejecting paths α of $M(\psi_0)$, N stores the assignment 1 for the variable x_1 by setting the first bit of s_α to 1, and continues recursively by simulating $M(\psi_{10})$.

After m such steps, N has found on each path α a candidate s_α of a satisfying assignment for φ.

Bounded-Error Probabilistic Polynomial Time

Step 2: On each path α, N checks deterministically whether s_α indeed satisfies φ.

If so, N accepts on α; otherwise, N rejects on α.

If $\varphi \notin \mathrm{SAT}$, $N(\varphi)$ rejects on all paths, due to the checking in Step 2. Thus,

$$\Pr(\{\alpha \,|\, N \text{ accepts } \varphi \text{ on path } \alpha\}) = 0.$$

Bounded-Error Probabilistic Polynomial Time

Suppose now that $\varphi \in \mathrm{SAT}$.

By (6), the error probability of M is at most 2^{-n} in each of the m simulations in Step 1.

Since each of these m trials are independent, the acceptance probability of $N(\varphi)$ can be estimated as follows:

$$\begin{aligned} \Pr(\{\alpha \mid N \text{ accepts } \varphi \text{ on path } \alpha\}) &\geq \left(1-2^{-n}\right)^m \\ &\geq \left(1-m2^{-n}\right) \\ &\geq \frac{1}{2}, \end{aligned}$$

where the latter inequality follows from the obvious fact that $m \leq n$, which implies $m2^{-n} \leq 2^{-1} = 1/2$. ❑

Reminder: Quantifier Representation of Complexity Classes

Recall that NP can be characterized by existential polynomially length-bounded quantifiers: A is in NP if and only if there is some set $B \in \mathrm{P}$ and a polynomial p such that for each $x \in \Sigma^*$,

$$x \in A \implies (\exists^p w)\,[\langle x, w\rangle \in B];$$
$$x \notin A \implies (\forall^p w)\,[\langle x, w\rangle \notin B].$$

Definition

1. Let $\mathfrak{Q}_1$ and $\mathfrak{Q}_2$ be two strings of n quantifiers each. The pair $(\mathfrak{Q}_1, \mathfrak{Q}_2)$ is *sensible* if and only if for each $(n+1)$-ary predicate B, for each x, and for each $\vec{y} = (y_1, y_2, \ldots, y_n)$,

$$(\mathfrak{Q}_1 \vec{y})\,[B(x, \vec{y})] \wedge (\mathfrak{Q}_2 \vec{y})\,[\neg B(x, \vec{y})]$$

is a contradiction. Here, y_i is the variable quantified by the i^{th} quantifier in $\mathfrak{Q}_1$ and $\mathfrak{Q}_2$, respectively.

Reminder: Quantifier Representation of Complexity Classes

Definition

2. Let $(\mathfrak{Q}_1, \mathfrak{Q}_2)$ be a sensible pair of strings consisting of n (polynomially length-bounded) quantifiers each. Define the complexity class $(\mathfrak{Q}_1 \,|\, \mathfrak{Q}_2)$ as follows: L belongs to $(\mathfrak{Q}_1 \,|\, \mathfrak{Q}_2)$ if and only if there exists an $(n+1)$-ary predicate $B \in \mathrm{P}$ such that for each $x \in \Sigma^*$:

$$x \in L \implies (\mathfrak{Q}_1 \vec{y})\,[B(x, \vec{y})];$$
$$x \notin L \implies (\mathfrak{Q}_2 \vec{y})\,[\neg B(x, \vec{y})],$$

where $\vec{y} = (y_1, y_2, \ldots, y_n)$ and y_i is the variable quantified by the i^{th} quantifier in $\mathfrak{Q}_1$ and $\mathfrak{Q}_2$, respectively, and $|y_i| \leq p(|x|)$ for some suitable polynomial p.

Example: $\mathrm{NP} = (\exists \,|\, \forall)$.

Polynomially Length-Bounded Majority Quantifier

Definition

Let B be a predicate, and let p be a given polynomial. For each fixed string x, define $(\exists^+ y)\,[B(x,y)]$ to be true if and only if at least three-quarters of all strings y with $|y| \leq p(|x|)$ satisfy $B(x,y)$.

Example: $\mathrm{BPP} = (\exists^+ \,|\, \exists^+)$ and $\mathrm{RP} = (\exists^+ \,|\, \forall)$.

Fact

For each sensible pair $(\mathfrak{Q}_1, \mathfrak{Q}_2)$ of quantifier strings,

$$(\mathfrak{Q}_1 \,|\, \mathfrak{Q}_2) = \mathrm{co}(\mathfrak{Q}_2 \,|\, \mathfrak{Q}_1).$$

Corollary

BPP *is closed under complementation.*

Quantifiers and BPP

Lemma

Let B be any predicate in P, let x be any string, and suppose that $(\forall y)(\exists^{+} z)[B(x,y,z)]$. Then, the following two statements are true:

1. $(\exists^{+} Z)(\forall y)(\exists z \in Z)[B(x,y,z)]$.
2. $(\forall Y)(\exists^{+} z)(\forall y \in Y)[B(x,y,z)]$.

- *In the first part, if $|z| \leq p(n)$ for some $p \in \mathbb{P}\mathrm{ol}$, then Z is viewed as a variable ranging over sets of strings of length at most $p(n)$. To ensure that Z itself can be represented by a string of length polynomially in n, we require Z to satisfy $\|Z\| = q(n)$ for some $q \in \mathbb{P}\mathrm{ol}$.*
- *An analogous comment applies to the set variable Y in the second part.* **without proof**

Quantifiers and BPP

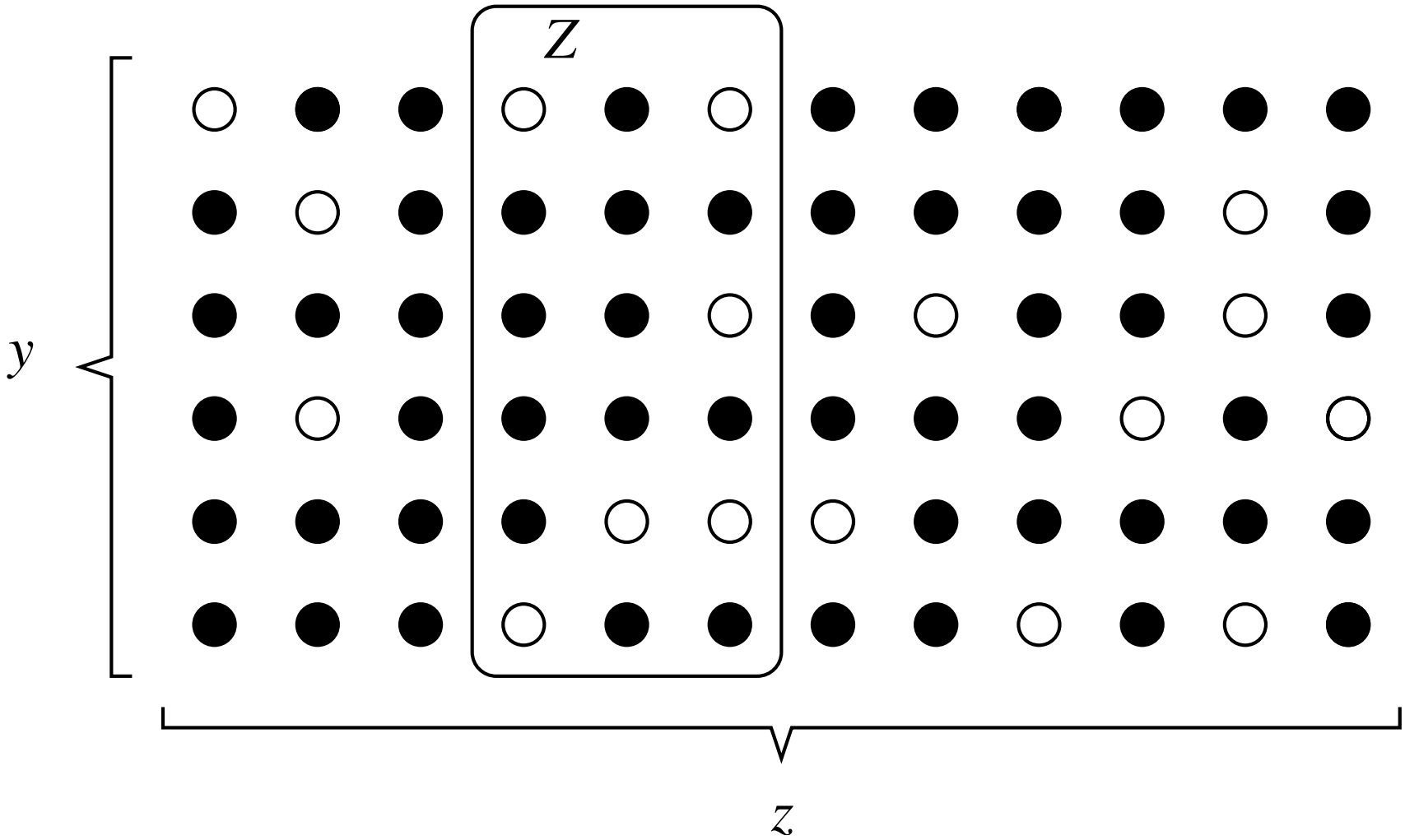


Figure: Illustration of the first statement of this lemma

Quantifiers and BPP

Lemma (Swapping Quantifiers)

$(\exists\forall|\forall\exists^+) \subseteq (\forall\exists|\exists^+\forall)$.

Proof: Let A be any set in $(\exists\forall|\forall\exists^+)$. By definition, there exists a predicate $B \in \mathrm{P}$ such that for each x,

$$
\begin{aligned}
x \notin A &\implies (\forall y)(\exists^+ z)[\neg B(x,y,z)] && \text{by definition}\\
&\implies (\exists^+ Z)(\forall y)(\exists z \in Z)[\neg B(x,y,z)] && \text{by part 1 of lemma}\\
&\implies (\exists Z)(\forall y)(\exists z \in Z)[\neg B(x,y,z)]\\
&\implies (\forall y)(\exists z)[\neg B(x,y,z)]\\
&\implies x \notin A,
\end{aligned}
$$

where the last $\implies$ follows from "$x \in A \implies (\exists y)(\forall z)[B(x,y,z)]$."

Quantifiers and BPP

Now, defining the predicate C by

$$C(x, y, Z) \equiv (\forall z \in Z)\,[B(x, y, z)],$$

it follows that

$$x \notin A \iff (\exists^{+} Z)\,(\forall y)\,[\neg C(x, y, Z)]; \tag{7}$$

$$x \notin A \iff (\exists Z)\,(\forall y)\,[\neg C(x, y, Z)]. \tag{8}$$

Negating (8), we obtain

$$x \in A \iff (\forall Z)\,(\exists y)\,[C(x, y, Z)]. \tag{9}$$

Since Z contains only polynomially many elements, $B \in \mathrm{P}$ implies $C \in \mathrm{P}$.

By definition, it follows from (7) and (9) that A is a set in $(\forall\exists \,|\, \exists^{+}\forall)$. ❑

Quantifiers and BPP: BPP Theorem by Zachos and Heller

Theorem

$$\mathrm{BPP} = (\exists^+ \,|\, \exists^+) = (\forall\exists^+ \,|\, \exists^+\forall) = (\exists^+\forall \,|\, \forall\exists^+).$$

Proof: It is enough to prove the equality $(\exists^+ \,|\, \exists^+) = (\forall\exists^+ \,|\, \exists^+\forall)$. The other equality follows immediately from the above fact and corollary, since

$$(\forall\exists^+ \,|\, \exists^+\forall) = \mathrm{co}(\exists^+\forall \,|\, \forall\exists^+).$$

To prove the inclusion $(\forall\exists^+ \,|\, \exists^+\forall) \subseteq (\exists^+ \,|\, \exists^+)$, let A be any set in $(\forall\exists^+ \,|\, \exists^+\forall)$. By definition, there is a set $B \in \mathrm{P}$ such that for each x,

$$\begin{aligned} x \in A &\implies (\forall y)\,(\exists^+ z)\,[B(x,y,z)] \\ &\implies (\exists^+ \langle y,z\rangle)\,[C(x,\langle y,z\rangle)], \end{aligned}$$

where C, defined by $C(x,\langle y,z\rangle) \equiv B(x,y,z)$, is in P.

Quantifiers and BPP: BPP Theorem by Zachos and Heller

Furthermore, for each x,

$$\begin{aligned} x \notin A &\implies (\exists^+ y)(\forall z)[\neg B(x,y,z)] \\ &\implies (\exists^+ \langle y,z\rangle)[\neg C(x,\langle y,z\rangle)]. \end{aligned}$$

Thus, A is in $(\exists^+ \,|\, \exists^+)$.

Conversely, to prove the inclusion $(\exists^+ \,|\, \exists^+) \subseteq (\forall\exists^+ \,|\, \exists^+\forall)$, let A be any set in $(\exists^+ \,|\, \exists^+)$. Thus, there exists a set $B \in \mathrm{P}$ such that for each x,

$$x \in A \implies (\exists^+ y)[B(x,y)]; \tag{10}$$

$$x \notin A \implies (\exists^+ y)[\neg B(x,y)]. \tag{11}$$

Let p be some polynomial bounding the lengths of the variables y quantified in (10) and (11).

Quantifiers and BPP: BPP Theorem by Zachos and Heller

Let x be any fixed string of length n. Let

$$S = \{0,1\}^{\leq p(n)}$$

be the set of all binary strings of length at most $p(n)$.

Suppose that the strings in S are lexicographically ordered, i.e., $S = \{s_0, s_1, \ldots, s_{m-1}\}$, where $m = 2^{p(n)+1} - 1$.

For fixed x, define a predicate $C(x,y,z)$ as follows, where the variables y and z range over the strings in S:

$$C(x, s_i, s_j) \equiv B(x, s_{i+j \bmod m}),$$

where i and j are from the set $\mathbb{Z}_m = \{0,1,\ldots,m-1\}$, i.e., they are residues modulo m.

Quantifiers and BPP: BPP Theorem by Zachos and Heller

z \ y	s_0	s_1	s_2	$\cdots$	s_{m-2}	s_{m-1}
s_0	b_0	b_1	b_2	$\cdots$	b_{m-2}	b_{m-1}
s_1	b_1	b_2	b_3	$\cdots$	b_{m-1}	b_0
s_2	b_2	b_3	b_4	$\cdots$	b_0	b_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
s_{m-2}	b_{m-2}	b_{m-1}	b_0	$\cdots$	b_{m-4}	b_{m-3}
s_{m-1}	b_{m-1}	b_0	b_1	$\cdots$	b_{m-3}	b_{m-2}

Table: Definition of predicate C

Quantifiers and BPP: BPP Theorem by Zachos and Heller

This table illustrates the definition of C: Letting $b_i = B(x, s_i)$ for $i \in \mathbb{Z}_m$,

- the table's i^{th} row gives the values of $C(x, y, s_i)$ for varying y, and
- the table's j^{th} column gives the values of $C(x, s_j, z)$ for varying z.

Note that $C(x, y, z)$ is symmetric in its last two arguments.

Thus, the rows are cyclically shifted by one position, and so are the columns.

Consequently, every row and every column in this table has the same number of ones and the same number of zeros.

Note that $B \in \mathrm{P}$ implies $C \in \mathrm{P}$.

Quantifiers and BPP: BPP Theorem by Zachos and Heller

From (10) it follows that for each x,

$$x \in A \implies (\forall z)(\exists^+ y)[C(x,y,z)]$$ since each row has the same number of ones

$$\implies (\forall y)(\exists^+ z)[C(x,y,z)]$$ since C is symmetric in its last two arguments

$$\implies (\forall Y)(\exists^+ z)\underbrace{(\forall y \in Y)[C(x,y,z)]}_{D(x,z,Y)}$$ by part 2 of the lemma,

where the predicate D is defined by $D(x,z,Y) \equiv (\forall y \in Y)[C(x,y,z)]$.

Since the $\forall$ quantifier in D ranges over a domain of polynomial size, $C \in \mathrm{P}$ implies $D \in \mathrm{P}$.

Quantifiers and BPP: BPP Theorem by Zachos and Heller

Similarly, from (11) it follows that for each x,

$$
\begin{aligned}
x \notin A \quad &\Longrightarrow \quad (\forall z)(\exists^{+} y)[\neg C(x,y,z)] && \text{since each row has the same number of zeros} \\
&\Longrightarrow \quad (\forall y)(\exists^{+} z)[\neg C(x,y,z)] && \text{since } C \text{ is symmetric in its last two arguments} \\
&\Longrightarrow \quad (\exists^{+} Z)(\forall y)\underbrace{(\exists z \in Z)[\neg C(x,y,z)]}_{\neg D(x,y,Z)} && \text{by part 1 of the lemma} \\
&\Longrightarrow \quad (\exists^{+} Z)(\forall y)[\neg D(x,y,Z)].
\end{aligned}
$$

Thus, A is in $(\forall\exists^{+} \,|\, \exists^{+}\forall)$. ❑

Reminder: The Arthur-Merlin Hierarchy

Definition (Arthur-Merlin Hierarchy)

The *levels of the Arthur-Merlin hierarchy* are the following classes:

$$\begin{array}{llllll} \mathrm{A} = (\exists^+ | \exists^+), & \mathrm{AM} = (\exists^+\exists | \exists^+\forall), & \mathrm{AMA} = (\exists^+\exists\exists^+ | \exists^+\forall\exists^+), \\ \mathrm{M} = (\exists | \forall), & \mathrm{MA} = (\exists\exists^+ | \forall\exists^+), & \mathrm{MAM} = (\exists\exists^+\exists | \forall\exists^+\forall), \ldots \end{array}$$

Define the *Arthur-Merlin hierarchy*, AMH, as the union of all these classes.

Theorem

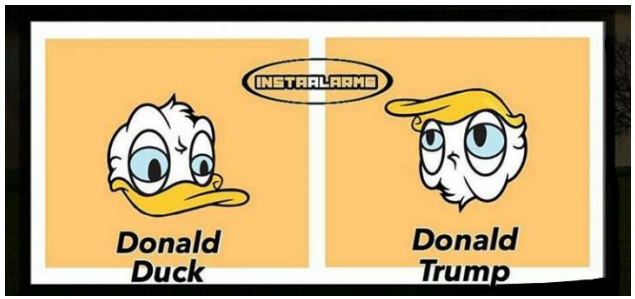
$$\mathrm{NP} \cup \mathrm{BPP} \subseteq \mathrm{MA} \subseteq \mathrm{AM} = \mathrm{AMA} = \mathrm{MAM} = \cdots = \mathrm{AMH}.$$

The Arthur-Merlin Hierarchy Collapses

Proof: By definition, $\mathrm{NP} \subseteq \mathrm{MA}$ and $\mathrm{BPP} \subseteq \mathrm{MA}$.

To prove the inclusion $\mathrm{MA} \subseteq \mathrm{AM}$, we

- first characterize these classes in terms of quantifier-based classes and
- then apply the swapping quantifiers lemma:



That is, we will prove that

$$\mathrm{MA} \stackrel{!}{=} (\exists\forall \,|\, \forall\exists^{+}) \subseteq (\forall\exists \,|\, \exists^{+}\forall) \stackrel{!}{=} \mathrm{AM}. \tag{12}$$

In what follows, we apply the swapping quantifiers lemma and the BPP Theorem in certain quantifier contexts, underlying the relevant quantifiers.

The Arthur-Merlin Hierarchy Collapses

The first equality in (12) can be seen as follows:

$$
\begin{aligned}
\mathrm{MA} &= (\exists\underline{\exists^+}\,|\,\forall\underline{\exists^+}) && \text{by definition}\\
&= (\exists\underline{\exists^+\forall}\,|\,\forall\underline{\forall\exists^+}) && \text{by BPP Theorem in quantifier context}\\
&\subseteq (\exists\exists\forall\,|\,\forall\forall\exists^+) && \text{since } (\exists^+ v)[\cdots] \text{ implies } (\exists v)[\cdots]\\
&= (\exists\forall\,|\,\forall\exists^+) && \text{by combining adjacent, equal quantifiers}\\
&\subseteq (\exists\exists^+\,|\,\forall\exists^+) && \text{since } (\forall v)[\cdots] \text{ implies } (\exists^+ v)[\cdots]\\
&= \mathrm{MA} && \text{by definition.}
\end{aligned}
$$

The Arthur-Merlin Hierarchy Collapses

The last equality in (12) can be seen as follows:

$$
\begin{aligned}
\mathrm{AM} &= (\underline{\exists^+}\exists \,|\, \underline{\exists^+}\forall) && \text{by definition} \\
&= (\underline{\forall\exists^+}\exists \,|\, \underline{\exists^+\forall}\forall) && \text{by BPP Theorem in quantifier context} \\
&\subseteq (\forall\exists\exists \,|\, \exists^+\forall\forall) && \text{since } (\exists^+ v)[\cdots] \text{ implies } (\exists v)[\cdots] \\
&= (\forall\exists \,|\, \exists^+\forall) && \text{by combining adjacent, equal quantifiers} \\
&\subseteq (\exists^+\exists \,|\, \exists^+\forall) && \text{since } (\forall v)[\cdots] \text{ implies } (\exists^+ v)[\cdots] \\
&= \mathrm{AM} && \text{by definition.}
\end{aligned}
$$

By the swapping quantifiers lemma, inclusion (12) is proven, so $\mathrm{MA} \subseteq \mathrm{AM}$.

The Arthur-Merlin Hierarchy Collapses

We now show that the entire Arthur-Merlin hierarchy collapses down to AM.

It is clear that we have the inclusions $\mathrm{AM} \subseteq \mathrm{MAM}$, $\mathrm{AM} \subseteq \mathrm{AMA}$, etc.

Conversely, applying the inclusion $\mathrm{MA} \subseteq \mathrm{AM}$ from (12) in a quantifier context implies

$$\mathrm{AMA} \subseteq \mathrm{AAM} \subseteq \mathrm{AM},$$

since two adjacent $\exists^+$ quantifiers can be combined to one $\exists^+$ quantifier the same way that this can be done for the $\exists$ or the $\forall$ quantifier.

Note that, for example, $(\exists^+\exists^+ \,|\, \exists^+\exists^+) = (\exists^+ \,|\, \exists^+) = \mathrm{BPP}$.
Equivalently, this can be written as $\mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$.

The Arthur-Merlin Hierarchy Collapses

The inclusion $\mathrm{MAM} \subseteq \mathrm{AM}$ can be seen as follows:

$$
\begin{aligned}
\mathrm{MAM} &= (\exists\underline{\exists^+}\exists \,|\, \forall\underline{\exists^+}\forall) && \text{by definition} \\
&= (\exists\underline{\exists^+\forall}\exists \,|\, \forall\underline{\forall\exists^+}\forall) && \text{by BPP Theorem in quantifier context} \\
&\subseteq (\exists\exists\forall\exists \,|\, \forall\forall\exists^+\forall) && \text{since } (\exists^+ v)[\cdots] \text{ implies } (\exists v)[\cdots] \\
&= (\underline{\exists\forall}\exists \,|\, \underline{\forall\exists^+}\forall) && \text{by combining adjacent, equal quantifiers} \\
&\subseteq (\underline{\forall\exists}\exists \,|\, \underline{\exists^+\forall}\forall) && \text{by swapping quantifiers lemma in context} \\
&= (\forall\exists \,|\, \exists^+\forall) && \text{by combining adjacent, equal quantifiers} \\
&\subseteq (\exists^+\exists \,|\, \exists^+\forall) && \text{since } (\forall v)[\cdots] \text{ implies } (\exists^+ v)[\cdots] \\
&= \mathrm{AM} && \text{by definition.}
\end{aligned}
$$

So $\mathrm{AM} = \mathrm{MAM} = \mathrm{AMA}$. The other equalities follow by induction. ❑

Universal Hashing

As a technical prerequisite to show that GI is in coAM, we need the so-called *hashing lemma*.

- Hashing is a method used in computer science for dynamic data management.
- Every data set is uniquely identified by some (short) key.
- The set of all potential keys, called the universe U, is usually very large, whereas the set $V \subseteq U$ of all keys actually used can be much smaller.
- A *hashing function* $h : U \to T$ maps the elements of U to the *hashing table* $T = \{0, 1, \ldots, k-1\}$.

Universal Hashing

- Hashing functions are many-to-one, which means that distinct keys from U can be mapped to the same address in T.
- If possible, however, any two distinct keys from V should be mapped to distinct addresses in T. That is, one seeks to avoid collisions on the set of actually used keys.
- In other words, a hashing function should, if possible, be injective on V.
- Hashing is also a very useful technique in cryptographic applications. Cryptographic hash functions usually map large keys to smaller keys in a "secure" manner.

Universal Hashing

- Among the various hashing techniques, *universal hashing* (which was invented by Carter and Wegman in 1979) is of particular interest for proving that GI is in coAM. The idea is to not focus on a particular, concrete hashing function, but rather to *randomly* select one from a suitable family of hashing functions.
- This hashing technique is universal in the sense that it no longer depends on a specific set V of keys that are actually used; instead, it seeks to avoid collisions on *all* sufficiently small sets V with high probability.
- The probability is taken over the random choice of hashing functions.
- In what follows, we think of keys as strings over the alphabet $\Sigma = \{0,1\}$, and we denote by Σ^n the set of all length n strings in Σ^*.

Universal Hashing

Definition (Universal Hashing)

Let $\Sigma = \{0,1\}$, and let m and t be integers with $t > m$.

A *hashing function* $h : \Sigma^t \to \Sigma^m$ is a linear mapping determined by a boolean $(m \times t)$ matrix $B_h = (b_{i,j})_{i,j}$, where $b_{i,j} \in \{0,1\}$.

For $\vec{x} \in \Sigma^t$ and for each i with $1 \leq i \leq m$, the i^{th} bit of $\vec{y} = h(\vec{x}) \in \Sigma^m$ is given by

$$y_i = (b_{i,1} \wedge x_1) \oplus (b_{i,2} \wedge x_2) \oplus \cdots \oplus (b_{i,t} \wedge x_t),$$

where $\oplus$ denotes the *exclusive-or* operation (a.k.a. the *parity* operation).

Note that $\oplus$ is associative. We can thus write:

$$a_1 \oplus a_2 \oplus \cdots \oplus a_n = 1 \iff \|\{i \,\big|\, a_i = 1\}\| \equiv 1 \bmod 2.$$

Universal Hashing

Definition (Universal Hashing, continued)

Let $\mathscr{H}_{t,m} = \{h : \Sigma^t \to \Sigma^m \mid B_h \text{ is a boolean } (m \times t) \text{ matrix}\}$
be a family of hashing functions for the parameters t and m.

We assume the uniform distribution on $\mathscr{H}_{t,m}$:
A hashing function h is chosen from $\mathscr{H}_{t,m}$ by picking the bits $b_{i,j}$ in B_h independently according to the uniform distribution.

Let $V \subseteq \Sigma^t$. For a subfamily $\widehat{\mathscr{H}}$ of $\mathscr{H}_{t,m}$, we say there is a *collision on V* if

$$(\exists \vec{v} \in V)(\forall h \in \widehat{\mathscr{H}})(\exists \vec{x} \in V)[\vec{v} \neq \vec{x} \wedge h(\vec{v}) = h(\vec{x})].$$

Otherwise, $\widehat{\mathscr{H}}$ is *collision-free on V*.

Universal Hashing

Lemma (Hashing Lemma)

Let $t, m \in \mathbb{N}$ be fixed parameters, let $V \subseteq \Sigma^t$, and let $\widehat{\mathscr{H}} = (h_1, h_2, \ldots, h_{m+1})$ be some family of hashing functions randomly selected from $\mathscr{H}_{t,m}$ under the uniform distribution.

Let the collision predicate *be*

$$\mathrm{Col}(V) = \{\widehat{\mathscr{H}} \mid (\exists \vec{v} \in V)(\forall h \in \widehat{\mathscr{H}})(\exists \vec{x} \in V)[\vec{v} \neq \vec{x} \wedge h(\vec{v}) = h(\vec{x})]\}.$$

That is, $\mathrm{Col}(V)$ is the event that, given $\widehat{\mathscr{H}}$, a collision occurs on V.

Then, the following two statements are true:

1. *If $\|V\| \leq 2^{m-1}$, then $\mathrm{Col}(V)$ occurs with probability at most $1/4$.*
2. *If $\|V\| > (m+1)2^m$, then $\mathrm{Col}(V)$ occurs with probability 1.*

without proof

Graph Isomorphism is in coAM

Theorem

GI *is in* coAM.

Proof: Let G and H be two graphs with n vertices each.

Reminder: Define the set

$$A(G,H) = \{\langle F, \varphi\rangle \,|\, F \cong G \wedge \varphi \in \mathrm{AUT}(F)\} \cup \{\langle F, \varphi\rangle \,|\, F \cong H \wedge \varphi \in \mathrm{AUT}(F)\}.$$

Lemma

For any two given graphs G and H with n vertices each, we have

$$\|A(G,H)\| = \begin{cases} n! & \text{if } G \cong H \\ 2n! & \text{if } G \ncong H. \end{cases}$$

Graph Isomorphism is in coAM

We want to apply the hashing lemma.

It seems to be reasonable to use as the set V from this lemma the set

$$A(G,H)=\{\langle F,\varphi\rangle \,|\, F\cong G\wedge\varphi\in \mathrm{AUT}(F)\}\cup\{\langle F,\varphi\rangle \,|\, F\cong H\wedge\varphi\in \mathrm{AUT}(F)\}.$$

We give an AM machine for GNI, the complement of GI, which, of course, must be polynomial-time bounded. This requires the parameters t and m from the hashing lemma to be polynomially in n. **However, we then would have to choose the polynomial $m=m(n)$ such that**

$$n!\leq 2^{m-1}<(m+1)2^m<2n!, \tag{13}$$

since otherwise the set $V=A(G,H)$ would not be large enough to tell two isomorphic graphs G and H apart from two nonisomorphic graphs, with sufficiently high probability as per the hashing lemma.

Graph Isomorphism is in coAM

Unfortunately, it is not possible to find a polynomial m satisfying (13).

That is why we choose, as our V from the hashing lemma, a set other than $A(G,H)$, one that creates a gap between the upper and the lower bound in (13) that is large enough so as to tell isomorphic graphs apart from nonisomorphic graphs. Define

$$V = A(G,H)^n = \underbrace{A(G,H) \times A(G,H) \times \cdots \times A(G,H)}_{n \text{ times}}.$$

Now, (13) becomes

$$(n!)^n \leq 2^{m-1} < (m+1)2^m < (2n!)^n, \tag{14}$$

and this inequality can be satisfied by setting $m = m(n) = 1 + \lceil n \log n! \rceil$, which is polynomially in n.

Graph Isomorphism is in coAM

Define an AM machine M for GNI as follows.

Given two graphs G and H with n vertices each, M starts by computing the parameter $m = m(n)$.

Note that the set $V = A(G, H)^n$ contains n-tuples of pairs of the form $\langle F, \varphi \rangle$, where F is a graph with n vertices, and φ is a permutation in the automorphism group $\mathrm{AUT}(F)$.

The elements of V can be suitably encoded as strings over the alphabet $\Sigma = \{0, 1\}$, where $t = t(n)$ is an appropriate polynomial.

So far, all computations are deterministic.

Graph Isomorphism is in coAM

M then makes a probabilistic move by Arthur: Randomly choose a family

$$\widehat{\mathscr{H}} = (h_1, h_2, \ldots, h_{m+1})$$

of hashing functions under the uniform distribution.

Each such hashing function $h_i \in \widehat{\mathscr{H}}$ is given by a boolean $(m \times t)$ matrix whose entries are chosen independently and uniformly distributed.

The $m+1$ hashing functions $h_i \in \widehat{\mathscr{H}}$ can thus be encoded by a string $r_{\widehat{\mathscr{H}}} \in \Sigma^*$ of length $p(n)$, for some suitable polynomial p.

Modify the collision predicate $\mathrm{Col}(V)$ from the hashing lemma as follows:

$$B = \left\{ \langle G, H, r_{\widehat{\mathscr{H}}} \rangle \;\middle|\; \begin{array}{l} (\exists \vec{v} \in V)\,(\forall i : 1 \leq i \leq m+1) \\ (\exists \vec{x} \in V)\,[\vec{v} \neq \vec{x} \wedge h_i(\vec{v}) = h_i(\vec{x})] \end{array} \right\}.$$

Graph Isomorphism is in coAM

Since the $\forall$ quantifier in B ranges over only polynomially many i, it can be evaluated deterministically in polynomial time.

Thus, the two $\exists$ quantifiers in B can be combined to just *one* polynomially length-bounded $\exists$ quantifier, which means that B is a set in NP.

Let N be an NPTM for B.

If $r_{\widehat{\mathscr{H}}}$ is the randomly chosen string encoding $m+1$ independently and uniformly distributed hashing functions from $\mathscr{H}_{t,m}$, then simulating the computation of $N(\langle G, H, r_{\widehat{\mathscr{H}}}\rangle)$ corresponds to Merlin's move.

This completes the description of M.

Graph Isomorphism is in coAM

Suppose that G and H are nonisomorphic.

By our previous lemma, $\|A(G,H)\| = 2n!$.

Inequality (14) implies

$$\|V\| = (2n!)^n > (m+1)2^m.$$

By the hashing lemma, the probability of $\langle G, H, r_{\widehat{\mathscr{H}}} \rangle$ being in B is 1, i.e., a collision occurs with certainty.

Thus, for each choice of $r_{\widehat{\mathscr{H}}}$, there exists an accepting computation path of $N(\langle G, H, r_{\widehat{\mathscr{H}}} \rangle)$.

Graph Isomorphism is in coAM

Now suppose that G and H are isomorphic.

By our previous lemma, $\|A(G,H)\| = n!$.

Inequality (14) implies

$$\|V\| = (n!)^n \leq 2^{m-1}.$$

By the hashing lemma, the probability of $\langle G, H, r_{\widehat{\mathscr{H}}} \rangle$ being in B is at most $1/4$.

Thus, for more than $3/4$ of the possible choices of $r_{\widehat{\mathscr{H}}}$, $N(\langle G, H, r_{\widehat{\mathscr{H}}} \rangle)$ has no accepting computation path.

It follows that GNI is in $(\exists^+\exists \,|\, \exists^+\forall) = \mathrm{AM}$;
or, equivalently, that GI is in $(\exists^+\forall \,|\, \exists^+\exists) = \mathrm{coAM}$. ❑