CS6160 Complexity

Topic: BPP in P/Poly
Lecturer: Wei-Kai Lin

Date: Oct 12, 2026
Scriber: Andrew

---

1 Recalling Definitions

Def: BPP. A language L is in BPP if there exists a polynomial-time Turing machine M with access to random bits r such that

$$x \in L \implies \Pr_r[M(x;r) = 1] \geq 2/3,$$
$$x \notin L \implies \Pr_r[M(x;r) = 0] \geq 2/3.$$

(Two-sided error.)

Def: RP. A language L is in RP if there exists a polynomial-time Turing machine M with access to random bits r such that

$$x \in L \implies \Pr_r[M(x;r) = 1] \geq 2/3,$$
$$x \notin L \implies \Pr_r[M(x;r) = 0] = 1.$$

(One-sided error: M never accepts a string outside L.)

2 Why Are NP Problems Hard?

Consider an NP-complete problem such as SAT:

$$\text{SAT}(\varphi) = 1 \iff \exists x \text{ s.t. } \varphi(x) = 1.$$

There are often many solutions to the same instance, and there can be many x that work. Now consider the unique-solution variant (a promise problem):

$$\text{USAT}(\varphi) = \begin{cases} 1, & \text{if } \exists\, x \text{ s.t. } \varphi(x) = 1, \\ 0, & \text{if } \varphi \notin \text{SAT (no satisfying assignment)}, \end{cases}$$

with no requirement on inputs having more than one solution. Here the promise is that the input does not have more than 1 solution. Is USAT easier or harder than SAT?

Theorem 1 (Valiant–Vazirani, '86)**.** *[VV86]* $\text{SAT} \leq_{rp} \text{USAT}$.

(This is a randomized reduction, not a Karp reduction.)

Poly-Randomized Reductions

- Karp: $A \leq_p B$ if there exists a polynomial-time M such that $A(x) = B(M(x))$ for all x.

- Poly-randomized: $A \leq_{rp} B$ if there exists a probabilistic polynomial-time Turing machine (PPTM) M such that

$$\Pr[B(M(x)) = 0 \mid A(x) = 0] = 1,$$
$$\Pr[B(M(x)) = 1 \mid A(x) = 1] \geq \frac{1}{\text{poly}(|x|)}.$$

Even though the probability of acceptance can be very low when it should accept, we can repeat to increase it as long as it is not negligible.

Transitivity. Karp reductions are transitive: $L \leq_p L'$ and $L' \leq_p L''$ imply $L \leq_p L''$. We can't make the same statements about randomized reductions due to the probabilities.

Closure. Suppose $A \leq_{rp} B$. Then

$$B \in \mathsf{RP} \implies A \in \mathsf{RP}, \qquad B \in \mathsf{BPP} \implies A \in \mathsf{BPP}.$$

(Repeat the reduction polynomially many times to amplify the 1/poly success probability to a constant, then run the algorithm for B.)

If SAT $\in$ RP, then RP = NP.

Remarks.

- Having a unique solution does not make the problem easier: finding or deciding with promise-unique solutions is still as hard as all of NP (under randomized reductions). This is also why counting the number of solutions is at least as hard as NP.
- The construction of the reduction uses hash functions.
- Important for cryptography.

---

3 Relationships Between BPP and Other Classes

- $\mathsf{BPP} \subseteq \mathsf{P/poly}$ (Adleman)[Adl78].
- $\mathsf{BPP} \subseteq \Sigma_2^P \cap \Pi_2^P$ (Sipser–Gács; Lautemann)[Sip83][Lau83].
- Is P = BPP? This is widely believed to be true (via derandomization), but it is open. In other words randomization doesn't help us very much.

Is SAT $\in$ BPP plausible? If SAT $\in$ BPP, then NP $\subseteq$ BPP $\subseteq$ P/poly. By Karp–Lipton[KL80], this implies $\mathsf{PH} = \Sigma_2^P$, i.e., the polynomial hierarchy collapses, which is not believed to be true. So it is unlikely that SAT $\in$ BPP.

Space-bounded analogues. Is L = RL = BPL? Open AI released a paper claiming to prove this conjecture[Ope26]. It is widely believed to be true but the paper is not very clear.

Intuition. The Sipser–Gács result, $\mathsf{BPP} \subseteq \Sigma_2^P \cap \Pi_2^P$, shows BPP sits low in the polynomial hierarchy, so randomness is not believed to be very powerful for time.

Proof of Adleman: $\mathsf{BPP} \subseteq \mathsf{P/poly}$

Let $L \in \mathsf{BPP}$. By amplification (repeat and take the majority; Chernoff bound), there is a polynomial-time probabilistic TM M such that

$$\forall x, \quad \Pr_r[M(x;r) = L(x)] \geq 1 - 2^{-2n},$$

where $n = |x|$ and $|r| = m = \text{poly}(n)$.

- Number of bad random strings r for a fixed x: $< 2^m \cdot 2^{-2n}$.
- Number of r that are bad for *some* x of length n (union bound over the 2^n inputs):

$$< 2^n \cdot 2^m \cdot 2^{-2n} = 2^{m-n}.$$

- Number of r that remain good for *all* x of length n:

$$> 2^m - 2^{m-n} \geq 2^{m-1} > 0.$$

Hence there exists r^* (depending only on n) such that $M(x;r^*) = L(x)$ for all $x \in \{0,1\}^n$. Hard-wiring r_n^* as the advice string for length n gives a $\mathsf{P/poly}$ algorithm for L. □

References

[Adl78] Leonard M. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pages 302–309, 1980.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[Ope26] OpenAI. Exact Derandomization of Logarithmic Space: $\mathsf{L} = \mathsf{RL} = \mathsf{BPL}$. OpenAI Math Release preprint OAI:Exact-Derandomization-of-Logarithmic-Space-L-equals-RL-equals-BPL-September-23-2026, 2026.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.