

COMPLEXITY THEORY

Lecture 23: Probabilistic Complexity Classes (2)

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 18th Jan 2021

More recent versions of this slide deck might be available.
For the most current version of this course, see
https://iccl.inf.tu-dresden.de/web/Complexity_Theory/en

Review: PP and BPP

Definition 21.4: A language \mathbf{L} is in **Polynomial Probabilistic Time (PP)** if there is a PTM \mathcal{M} such that:

- there is a polynomial function f such that \mathcal{M} will always halt after $f(|w|)$ steps on all input words w ,
- if $w \in \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] > \frac{1}{2}$,
- if $w \notin \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] \leq \frac{1}{2}$.

Definition 21.11: A language \mathbf{L} is in **Bounded-Error Polynomial Probabilistic Time (BPP)** if there is a PTM \mathcal{M} such that:

- there is a polynomial function f such that \mathcal{M} will always halt after $f(|w|)$ steps on all input words w ,
- if $w \in \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] \geq \frac{2}{3}$,
- if $w \notin \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] \leq \frac{1}{3}$.

Review: Polynomial Identity Testing in BPP

Algorithm: For a polynomial $p(x_1, \dots, x_m)$

- Randomly select a number $k \in \{1, \dots, 2^{2n}\}$
- Randomly select $a_1, \dots, a_n \in \{1, \dots, 10 \cdot 2^n\}$ (a total of $O(n \cdot m)$ random bits)
- Evaluate the circuit modulo k to compute $p(a_1, \dots, a_m) \pmod k$
- Repeat this experiment for $4n$ times and accept if and only if the outcome is 0 in all cases

This leads to a constant error probability of < 0.5 for polynomials that are non-zero (which can be amplified to be $\leq \frac{1}{3}$), and an error probability of 0 for polynomials that are.

BPP and other classes

The neighbours of BPP

We have already observed that $P \subseteq BPP$.

Moreover, since PP used less strict conditions on probabilities, we immediately get

$$BPP \subseteq PP \subseteq PSpace$$

Another interesting result is the following:

Theorem 23.1 (Adleman's¹ Theorem): $BPP \subseteq P_{/poly}$

(remember that we also knew that $P \subseteq P_{/poly}$ but not whether $NP \subseteq P_{/poly}$)

¹) Adleman is the A in RSA.

Proving Adleman's Theorem

Theorem 23.1 (Adleman's Theorem): $BPP \subseteq P_{/poly}$

Proof: By Theorem 21.12, any language in BPP is recognised by a PTM \mathcal{M} with error probability $\leq \frac{1}{2^{n+1}}$, for an input of size n . Moreover, \mathcal{M} uses a polynomial (in n) number m of random bits $r \in \{0, 1\}^m$ (verifier perspective on PTMs).

- r is **bad** for input $w \in \{0, 1\}^n$ if \mathcal{M} returns the wrong answer on w for random bits r ; otherwise r is **good** for w
- Because of the error probability, there are $\leq \frac{2^m}{2^{n+1}}$ bad strings for any w
- In total, for all 2^n inputs, there are $\leq 2^n \frac{2^m}{2^{n+1}} = \frac{2^m}{2}$ bad strings
- Therefore, there are strings r that are good for all inputs

Take one such universally good string \hat{r} ; build a circuit for a deterministic verifier TM of inputs $w\#r$ as in Theorem 19.7; hardwire \hat{r} as input for the certificate. \square

BPP and the Polynomial Hierarchy

Recall: We have defined the polynomial hierarchy in two ways:

- Polytime ATMs with number of alternations bounded by a constant
- Oracle (N)TMs that use oracles for lower levels of the hierarchy

For example, $\Sigma_2^P = \text{NP}^{\text{NP}} = \text{NP}^{\text{coNP}}$, the languages recognised by polytime ATMs that begin their runs in an existential state and may alternate to a universal state later on

One would not immediately expect that these classes are related to BPP, yet we have:

Theorem 23.2 (Sipser-Gács-Lautemann Theorem): $\text{BPP} \subseteq \Sigma_2^P \cap \Pi_2^P$

Notes:

- Michael Sipser first showed that $\text{BPP} \subseteq \text{PH}$; Peter Gács then showed the theorem; Clemens Lautemann then gave the readable proof we will show – all in 1983
- The result has been further strengthened since, suggesting that BPP strictly smaller, but no relation to any other class we covered so far is known

Proving Sipser-Gács-Lautemann (1)

Theorem 23.2 (Sipser-Gács-Lautemann Theorem): $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$

Proof: Overall proof outline:

- We will show that $BPP \subseteq \Sigma_2^P$. This implies $\text{coBPP} \subseteq \Pi_2^P$, and hence $BPP \subseteq \Pi_2^P$ since BPP is closed under complement.
- We will show the inclusion for an arbitrary language $L \in BPP$.
- Then there is a PTM M with the following features:
 - M runs in time $p(n)$ for some polynomial p , using $p(n)$ random bits
 - M accepts L with error probability $\leq 2^{-n}$
(using probability amplification as in Theorem 21.14)

We can view the computation of M as a deterministic polytime computation over an input of length n and an additional string of $p(n)$ random bits, as before.

- The key to the proof is the extreme difference between acceptance and rejection:
 - either $\geq (1 - 2^{-n})2^{p(n)}$ of random vectors $r \in \{0, 1\}^{p(n)}$ lead to acceptance,
 - or only $\leq 2^{-n}2^{p(n)} = 2^{p(n)-n}$ of random vectors $r \in \{0, 1\}^{p(n)}$ lead to acceptance.
- \leadsto we want to tell the two situations apart in Σ_2^P

Proving Sipser-Gács-Lautemann (2)

Theorem 23.2 (Sipser-Gács-Lautemann Theorem): $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$

Proof (continued): Idea for telling apart acceptance and rejection:

- For input w , let $S_w \subseteq \{0, 1\}^{p(n)}$ be the set of all random vectors such that, for all $r \in S_w$, \mathcal{M} accepts w when using random numbers r
- $S_w \subseteq \{0, 1\}^{p(n)}$ is either almost all of $\{0, 1\}^{p(n)}$, or a tiny fraction thereof
- We consider “shifted copies” of S_w , created by some uniform bit-flipping S_w vectors:
 - If S_w is large, then polynomially many such copies can cover all of $\{0, 1\}^{p(n)}$
 - If S_w is small, then polynomially many copies are too small to cover $\{0, 1\}^{p(n)}$
- Making a “shifted copy”:
for some $u \in \{0, 1\}^{p(n)}$, set $S_w \oplus u = \{r \oplus u \mid r \in S_w\}$, where \oplus is XOR (sum mod 2)
- Number of shifted copies: we will use $k = \left\lceil \frac{p(n)}{n} \right\rceil + 1$ copies (a polynomial number)

We will show that k random shifts can cover $\{0, 1\}^{p(n)}$ if and only if S_w is “large”.

Proving Sipser-Gács-Lautemann (3)

Theorem 23.2 (Sipser-Gács-Lautemann Theorem): $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$

Proof (continued):

Claim 1: If $|S_w| \leq 2^{p(n)-n}$, then, for every set of $k = \left\lceil \frac{p(n)}{n} \right\rceil + 1$ vectors $u_1, \dots, u_k \in \{0, 1\}^{p(n)}$, we have $\bigcup_{i=1}^k (S_w \oplus u_i) \subsetneq \{0, 1\}^{p(n)}$.

The result follows from the cardinalities of the involved sets:

Using that $|S_w \oplus u_i| = |S_w|$, we obtain:

$$\left| \bigcup_{i=1}^k (S_w \oplus u_i) \right| \leq k|S_w| \leq \left(\left\lceil \frac{p(n)}{n} \right\rceil + 1 \right) 2^{p(n)-n} = \frac{\left(\left\lceil \frac{p(n)}{n} \right\rceil + 1 \right) 2^{p(n)}}{2^n} = o(2^{p(n)})$$

Therefore the claim holds for sufficiently large n .

This suffices, since inputs of shorter length can surely be decided in Σ_2^P as well.

Proving Sipser-Gács-Lautemann (4)

Theorem 23.2 (Sipser-Gács-Lautemann Theorem): $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$

Proof (continued):

Claim 2: If $|S_w| \geq (1 - 2^{-n})2^{p(n)}$, then there is a set of $k = \lceil \frac{p(n)}{n} \rceil + 1$ vectors $u_1, \dots, u_k \in \{0, 1\}^{p(n)}$, such that $\bigcup_{i=1}^k (S_w \oplus u_i) = \{0, 1\}^{p(n)}$.

We argue that, for independently and randomly chosen u_1, \dots, u_k , we have $\Pr \left[\bigcup_{i=1}^k (S_w \oplus u_i) = \{0, 1\}^{p(n)} \right] > 0$. The claim follows from this.

For a particular $r \in \{0, 1\}^{p(n)}$, we compute

$$\Pr \left[r \notin \bigcup_{i=1}^k (S_w \oplus u_i) \right] \stackrel{(a)}{=} \prod_{i=1}^k \Pr [r \notin (S_w \oplus u_i)] \stackrel{(b)}{\leq} \prod_{i=1}^k 2^{-n} = 2^{-nk} = 2^{-n(\lceil \frac{p(n)}{n} \rceil + 1)} < 2^{-p(n)}$$

since: (a) u_i are selected independently; (b) $\Pr [r \notin (S_w \oplus u_i)] = \Pr [r \oplus u_i \notin S_w] \leq 2^{-n}$

Therefore: $\Pr \left[\text{there is } r \in \{0, 1\}^{p(n)} \setminus \bigcup_{i=1}^k (S_w \oplus u_i) \right] < 2^{p(n)} \cdot 2^{-p(n)} = 1$. In particular, there is at least one choice of u_1, \dots, u_k where this event does not occur, i.e., where all r are in $\bigcup_{i=1}^k (S_w \oplus u_i)$.

Proving Sipser-Gács-Lautemann (5)

Theorem 23.2 (Sipser-Gács-Lautemann Theorem): $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$

Proof (continued): In summary, we have shown:

- If S_w is “small,” then there are no vectors u_1, \dots, u_k such that $\bigcup_{i=1}^k (S_w \oplus u_i) = \{0, 1\}^{p(n)}$
- If S_w is “large,” then there are vectors u_1, \dots, u_k such that $\bigcup_{i=1}^k (S_w \oplus u_i) = \{0, 1\}^{p(n)}$

Hence, we can check the acceptance of \mathcal{M} by computing if the following holds true:

$$\exists u_1, \dots, u_k. \forall r \in \{0, 1\}^{p(n)}. r \in \bigcup_{i=1}^k (S_w \oplus u_i)$$

Using the DTM version of PTMs, this becomes:

$$\exists u_1, \dots, u_k. \forall r \in \{0, 1\}^{p(n)}. \bigvee_{i=1}^k \mathcal{M} \text{ accepts } w \text{ for random vector } r \oplus u_i$$

This is a Σ_2^P computation. □

Hierarchy Theorems for BPP

The **Time Hierarchy Theorems** for deterministic and non-deterministic Turing machines show that, when given (sufficiently) more time, such TMs can solve more problems. In particular:

- $P \neq \text{ExpTime}$
- $NP \neq \text{NExpTime}$

The proofs were based on **diagonalisation arguments** that enabled TMs with more time to deliberately differ from all TMs with less time.

Unfortunately, no such arguments are known for BPP:

- The difficulty of applying diagonalisation arguments is related to the semantic definition of BPP.
- Currently, we don't even know if $BPP \neq \text{NExpTime}$!

Relationship of BPP and P

We know $P \subseteq BPP \subseteq PP \subseteq PSpace$ but not even if $BPP \neq NExpTime$.

However, most experts expect that ...

BPP is equal to P!

- Many BPP algorithms have been de-randomised successfully
- $BPP = P$ is equivalent to the existence of strong pseudo-random number generators, which many experts consider likely

Further probabilistic classes

Types of errors

We have defined BPP by restricting the probability of error to $\leq \frac{1}{3}$.

However, there are two types of errors:

- **False positives:** the PTM accepts a word that is not in the language
- **False negatives:** the PTM rejects a word that is in the language

Common BPP algorithms can often avoid one of these errors:

Example 23.3: Our previous algorithm for polynomial identity testing aimed to decide **ZEROP**. For inputs $w \in \mathbf{ZEROP}$, the algorithm accepted with probability 1 (no false negatives). Uncertainty only occurred for inputs $w \notin \mathbf{ZEROP}$ (false positives were possible, though unlikely).

Randomised Polynomial Time

Excluding false positives/negatives from BPP leads to classes with one-sided error:

Definition 23.4: A language \mathbf{L} is in **Randomised Polynomial Time (RP)** if there is a PTM \mathcal{M} such that:

- there is a polynomial function f such that \mathcal{M} will always halt after $f(|w|)$ steps on all input words w ,
- if $w \in \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] \geq \frac{2}{3}$,
- if $w \notin \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] = 0$.

Definition 23.5: A language \mathbf{L} is in **coRP** if its complement is in RP, i.e., if there is a polynomially time-bounded PTM \mathcal{M} such that:

- if $w \in \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] = 1$,
- if $w \notin \mathbf{L}$, then $\Pr[\mathcal{M} \text{ accepts } w] \leq \frac{1}{3}$.

Example 23.6: $\mathbf{ZERO}P \in \mathbf{coRP}$.

Probability amplification for RP and coRP

It is clear from the definitions that $\text{RP} \subseteq \text{BPP}$ and $\text{coRP} \subseteq \text{BPP}$.

Hence, we can apply Theorem 21.14 to amplify the output probability.

However, the situation for one-sided error classes is actually much simpler:

Theorem 23.7: Consider a language \mathbf{L} and a polynomially time-bounded PTM \mathcal{M} for which there is a constant $c > 0$ such that, for every word $w \in \Sigma^*$,

- if $w \in \mathbf{L}$ then $\Pr[\mathcal{M} \text{ accepts } w] \geq |w|^{-c}$
- if $w \notin \mathbf{L}$ then $\Pr[\mathcal{M} \text{ accepts } w] = 0$

Then, for every constant $d > 0$, there is a polynomially time-bounded PTM \mathcal{M}' such that

- if $w \in \mathbf{L}$ then $\Pr[\mathcal{M}' \text{ accepts } w] \geq 1 - 2^{-|w|^d}$
- if $w \notin \mathbf{L}$ then $\Pr[\mathcal{M}' \text{ accepts } w] = 0$.

Proof: Much simpler than for BPP (exercise). □

RP and NP

The asymmetric acceptance conditions of RP reminds us of NP, since already “some” accepting runs are enough to prove acceptance.

Indeed, we get:

Theorem 23.8: $RP \subseteq NP$

Proof: If \mathcal{M} satisfies the RP acceptance conditions for L , then \mathcal{M} can be considered as an NTM that accepts L with respect to the usual non-deterministic acceptance conditions. Indeed, \mathcal{M} has an accepting run on input $|w|$ if and only if $w \in L$. \square

Similarly, we find $coRP \subseteq coNP$.

Recall: While $RP \subseteq BPP$, we do not know whether $BPP \subseteq NP$.

Zero-sided error

Instead of admitting a possibly false answer (positive or negative), one can also require the correct answer while making some concessions on runtime:

Definition 23.9: A PTM \mathcal{M} has **expected runtime** $f : \mathbb{N} \rightarrow \mathbb{R}$ if, for any input w , the expectation $E[T_w]$ of the number T_w of steps taken by \mathcal{M} on input w is $T_w \leq f(|w|)$.

ZPP is the class of all languages for which there is a PTM \mathcal{M} that

- returns the correct answer whenever it halts,
- has expected runtime f for some polynomial function f .

ZPP is for **zero-error probabilistic polynomial time**.

Note: In general, algorithms that produce correct results while giving only probabilistic guarantees on resource usage are called **Las Vegas algorithms**, as opposed to **Monte Carlo algorithms**, which have guaranteed resource bounds but probabilistic correctness (as in the case of BPP).

Zero-sided vs. one-sided error

In spite of the different approaches of expected error vs. expected runtime, we find a close relation between ZPP, RP, and coRP:

Theorem 23.10: $ZPP = RP \cap \text{coRP}$

Proof: $ZPP \subseteq RP$: Given a ZPP algorithm \mathcal{M} , construct an RP algorithm by running \mathcal{M} for three times the expected (polynomial) runtime t . If it stops, return the same answer; if it times out, reject.

- For any random variable X and $c > 0$, Markov's inequality implies:
$$\Pr[X \geq cE[X]] \leq \frac{E[X]}{cE[X]} = \frac{1}{c}$$
- Hence the probability of \mathcal{M} running for $\geq 3t$ is $\leq \frac{1}{3}$
- Therefore, the probability of a false negative (due to a timeout) is $\leq \frac{1}{3}$

$ZPP \subseteq \text{coRP}$ is dual; we just have to accept after timeout.

Zero-sided vs. one-sided error

In spite of the different approaches of expected error vs. expected runtime, we find a close relation between ZPP, RP, and coRP:

Theorem 23.10: $ZPP = RP \cap \text{coRP}$

Proof: $ZPP \supseteq RP \cap \text{coRP}$: Assume we have an RP algorithm \mathcal{A} and a coRP algorithm \mathcal{B} for the same language L . To obtain a ZPP algorithm, we run \mathcal{A} and \mathcal{B} on input w :

- If \mathcal{A} accepts, accept
- If \mathcal{B} rejects, reject
- If \mathcal{A} rejects and \mathcal{B} accepts, repeat the experiment.

Since RP has no false positives and coRP has no false negatives, this can only return the correct answer.

The probability of repetition is $\leq \frac{1}{3}$, since it requires one of the algorithms to be in error.

Hence the probability of k repetitions is $\leq 3^{-k}$, for an expected runtime of $\leq \sum_{k \geq 0} \frac{(k+1)p}{3^k}$, where p is the combined (polynomial) runtime of \mathcal{A} and \mathcal{B} . This is polynomial. \square

Summary and Outlook

Complexity relationships: see board (or make your own drawing)

Probabilistic classes with ones-sided error – RP and coRP – are common.

ZPP defines random computations with zero-sided error, but probabilistic runtime.

Many experts believe that

$$P = ZPP = RP = \text{coRP} = \text{BPP} \subsetneq \text{PP}$$

What's next?

- Quantum computing
- Summary
- Examinations