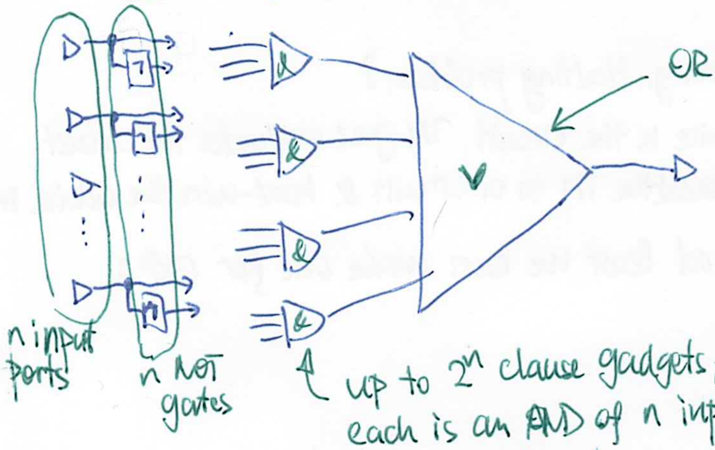
CIRCUIT COMPLEXITY

Back to (non-uniform) Boolean circuits.
We will use only AND, OR, NOT gates (other gates can be simulated with constant overhead).
Circuit size = #gates + #ports (input & output) ← size ≤ S is equivalent with size = S as we can pad circuits with redundant gates

Thm: For every function $f: \{0,1\}^n \to \{0,1\}$ there exists a circuit of size $\le 10n \cdot 2^n$ computing f.

Proof: Use DNF formula for f (see lecture on Cook-Levin thm.)



Note: There is a better construction with $O(2^n/n)$ gates. ← surprisingly, this is asymptotically optimal
Exercise: Achieve $O(2^n)$ gates.

Thm: For all sufficiently large n, there is $f: \{0,1\}^n \to \{0,1\}$ which is computed by no circuit of size at most $2^n/10n$.

Proof: There are 2^{2^n} functions from $\{0,1\}^n$ to $\{0,1\}$.
Let's count circuits of a given size s:

$$\#\text{circuits} \le 3^s \cdot s^{2s} = 2^{s \cdot \log_2 3} \cdot 2^{2s \log s} \le 2^{3s \log s}$$

↑ except for ports, each gate can be AND, OR, NOT
↑ # interconnections: each gate has at most 2 inputs, which are connected to a port or output of another gate

Now for $s = 2^n/10n$:

$$\#\text{circuits} \le 2^{\frac{3 \cdot 2^n}{10n} \cdot n} < 2^{2^n} \text{ for } n \text{ large enough.}$$ ← in fact, the majority of functions has no small circuits

Notes: All problems in P have polynomially large circuits.
Hypothesis (Kolmogorov): $O(n)$ is enough.
Surprisingly, the best lower bound so far is $5n$.

Idea: If we found $L \in NP$ with super-polynomial lower bound for circuit size, then $P \ne NP$.
But so far, we failed completely...

Df: For $S: \mathbb{N} \to \mathbb{N}$ we define $SIZE(S(n))$ as the class of languages, which are computable by a (non-uniform) family $\{C_n\}_{n=0}^{\infty}$ of circuits s.t. size of $C_n \le S(n)$. ← beware, no O here

We know: $P \subseteq \bigcup_k SIZE(n^k + k)$ ← this is to overcome finitely many exceptions in O

Df: Computation with advice: the TM gets an extra input (advice), which depends only on the size of the main input.

$DTIME(f(n))/g(n)$: the class of languages L s.t. $\exists M$ Turing Machine and $\exists a: \mathbb{N} \to \{0,1\}^*$
where $\forall \alpha \in \{0,1\}^*$ with $n = |\alpha|$: $M(\langle \alpha, a(n) \rangle)$ halts within $O(f(n))$ steps, accepts iff $\alpha \in L$ and $|a(n)| \le g(n)$.

($f(n)$: time bound; $g(n)$: size of advice)

Then: $P/g(n) := DTIME(poly(n))/g(n)$

- $P/0 = P$
- $P/1 \supsetneq P$, because $P/1$ contains the unary Halting problem!
- $P/poly = \bigcup_k SIZE(n^k + k)$ … $\supseteq$ the advice is the circuit, TM just evaluates the circuit
  $\subseteq$ we translate the TM to a circuit & hard-wire the advice in it

Circuit lower bounds for NP are hard, but at least we can make one for EXP:

Theorem: $\forall k \ge 0 \ \exists L \in EXP \setminus SIZE(n^k + k)$.

Proof: L is defined by the following algorithm:

1. Let α be an input of size n.
2. Let $\beta_0, \dots, \beta_{2^n-1}$ denote possible inputs for an n-input circuit: $\beta_j :=$ j written in binary.
3. $C_0 \leftarrow$ {all circuits of size $n^k + k$ with n inputs}
4. $i \leftarrow 0$
5. While $C_i \ne \emptyset$ & $i < 2^n$:
6. Simulate all circuits in C_i on input β_i, let t_i be the minority answer.
7. $C_{i+1} \leftarrow$ those circuits from C_i which gave output t_i
8. $i \leftarrow i+1$
9. If $\alpha = \beta_j$ for some $j < i$: answer t_j
   Else: answer NO ← arbitrary

Now: $C_{i+1} \le \frac{1}{2}|C_i| \Rightarrow C_i \le |C_0|/2^i$
$|C_0| \le 2^{n^{k+1}}$ ← see calculations in circuit lower bound thm. $\Rightarrow$ after less than 2^n steps, we get $C_i = \emptyset$ — works for n large enough (i.e., we don't run out of β_j's)

So the whole algorithm runs in time $O(2^{n^{k+2}})$, so $L \in EXP$.
But no circuit in $SIZE(n^k + k)$ can agree with L on n large enough.

Improvement: Choose $k := \lfloor \log n \rfloor$ … then run time is in $O(2^{n^{\log n + 2}}) \subseteq \underbrace{O(2^{2^n})}_{\text{this is called EEXP or 2-EXP}}$

$n^{\log n + 2} = 2^{\log^2 n + 2}$

So L is in EEXP, but not in $SIZE(n^k + k)$ for any k.

So $L \in EEXP \setminus P/poly$.

Therefore $EEXP \not\subseteq P/poly$.

Theorem: If $NP \subseteq P/poly$, then $PH = \Sigma_2^P$. ← generally, it's believed that the PH does not collapse, so there should be languages in NP with no poly-size circuits

Proof: (not shown at the lecture)

- we want to show that $\Sigma_2^P = \Pi_2^P$
- so $\Pi_2^P \subseteq \Sigma_2^P$ suffices (the other inclusion by taking complements)
- so Π_2-SAT $\in \Sigma_2^P$ suffices
  - this is $\{ \langle\psi\rangle \mid \psi$ is a true formula of the form $\forall\alpha \in \{0,1\}^n\ \exists\beta \in \{0,1\}^n\ \varphi(\alpha,\beta) \}$
  - unquantified fla of size $O(n)$
- If $NP \subseteq P/poly$, there is a family of poly-size circuits $\{C_n\}_{n=0}^{\infty}$ solving: given $\langle\varphi\rangle$ and α, is there β s.t. $\varphi(\alpha,\beta)$ is true?
  - ↳ we can convert this to $\langle\varphi\rangle, \alpha \mapsto$ find that β ... still within polynomial size → circuits C'_n
  - the exercise with SAT oracle earlier...
- We don't know how C'_n looks, but we can guess it and verify:
  $\exists \langle C'_n \rangle\ \forall\alpha\ \varphi(\alpha, C'_n(\alpha))$ ... this solves Π_2-SAT, but it is in Σ_2^P.

PROBABILISTIC ALGORITHMS (a.k.a. randomized)

Define the Probabilistic TM (PTM): random states & exactly 2 possible instructions, the TM decides by flipping a fair coin (i.e., generates an uniformly random bit, independent of all the other random bits) ← another form of non-determinism like ∃ & ∀ states

↓

We have a probability distribution on computations

↳ $P(M \text{ accepts } \alpha)$

Df:
- $BPTIME(f(n))$:= class of all languages L s.t. $\exists M$ PTM: all computations halt within $O(f(n))$ steps and $P(M(\alpha) = L(\alpha)) \geq 2/3$. (2-sided error)
  - $L(\alpha)$: indicator of $\alpha \in L$

  ↓

  $BPP := BPTIME(poly(n))$
- $RTIME(f(n))$:= class of all languages L s.t. $\exists M$ PTM: all computations halt within $O(f(n))$ steps, (1-sided error)
  - if $\alpha \in L$: $P(M(\alpha) \text{ accepts}) \geq 2/3$
  - if $\alpha \notin L$: $P(M(\alpha) \text{ accepts}) = 0$

  ↓

  $RP := RTIME(poly(n))$, co-RP (error at the opposite side)

Amplification of probability of success:

① For RP: Let $L \in RP$, M the corresponding PTM. Run $M(\alpha)$ t times independently, accept if at least one run accepted.
  - → this is still poly-time...
  - $\alpha \notin L$: always rejected
  - $\alpha \in L$: rejected with pr. $\leq (1 - 2/3)^t$

  symmetrically for co-RP

  Corollary: Iterating decreases pr. of error exponentially with #tries. This works whenever the original $P(\text{accepts}) \geq c$ for any $c > 0$. So the definition is robust wrt. change of the (arbitrary) 2/3.

② For BPP: Run t (odd) times independently, use majority answer.

Analysis: Let random variable X_i := indicator of correct answer in try #i, $E[X_i] \geq \frac{2}{3}$ (assume $= \frac{2}{3}$, more is obviously better)

Let $X := \sum_i X_i$ (# successful tries), then $\mathbb{E}[X] = \frac{2}{3} \cdot t$

$P(\text{majority is wrong}) = P(X < \frac{1}{2} t)$

this is Chernoff with $\mu = \frac{2}{3} t$, $(1-\delta) \cdot \frac{2}{3} = \frac{1}{2}$, so $\delta = \frac{1}{4}$.

Hence $P \le e^{-\frac{(\frac{1}{4})^2 \cdot \frac{2}{3} \cdot t}{2}} = e^{-\frac{2 \cdot t}{4^2 \cdot 3 \cdot 2}} = e^{-\Omega(t)}$

Tool: Chernoff's bound for the left tail: Let $X_1 - X_k$ be independent random vars with domain $\{0,1\}$, $X = \sum_i X_i$, $\mu = \mathbb{E}[X_i]$, $\delta \in (0,1)$. Then: $P(X < (1-\delta)\mu) \le e^{-\frac{\delta^2 \mu}{2}}$

So Pr. of error again decreases exponentially with # tries.
This works whenever the original machine answers correctly with $P \ge c$, where $c > \frac{1}{2}$.

Certificate-based definition: • BPP is the class of all languages L s.t. $\exists V \in P$ ← verifier and $\forall \alpha \in \{0,1\}^*$ $P_{\beta \in \{0,1\}^{poly(n)}}(V(\langle \alpha, \beta \rangle) = L(\alpha)) \ge \frac{2}{3}$.

↑ input (α) ↑ certificate (β)

→ Why this is the same BPP: old ⇒ this: the certificate is a sequence of all random bits generated by the TM, the rest can be simulated deterministically

padded to the same size for all computations

this ⇒ old: first generate random β, then run V.

• for RP: $L \in RP \Leftrightarrow \exists V \in P$ $\forall \alpha \in \{0,1\}^*$: $P_{\beta \in \{0,1\}^{poly(n)}}(V(\langle \alpha, \beta \rangle) = 1) \begin{cases} \ge 2/3 & \text{if } \alpha \in L \\ = 0 & \text{if } \alpha \notin L \end{cases}$

→ this implies $RP \subseteq NP$

"Zero-based errors": TWO definitions: ① TM runs in expected time $O(t(n))$, always answers correctly
② TM runs in worst-case time $O(t(n))$, can answer MAYBE. If answer is not MAYBE, it's correct. P(answers MAYBE) $\le 1/3$.

ZTIME$(t(n))$
↓
ZPP := ZTIME(poly(n))

① ⇒ ② Run machine for $3 \cdot t(n)$ steps, if it times out, answer MAYBE.
$P(\text{MAYBE}) = P(\text{time} \ge 3 \cdot \mathbb{E}(\text{time})) \le \frac{1}{3}$ ← by Markov's inequality

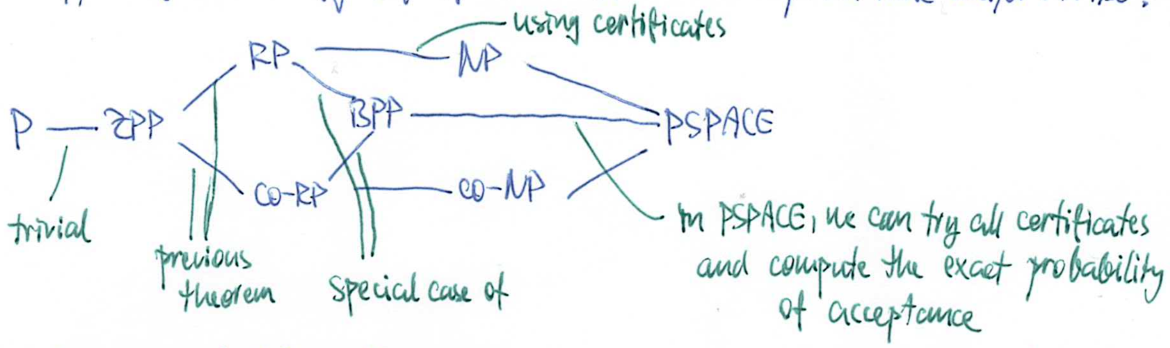
② ⇒ ① Run machine repeatedly as long as it returns MAYBE.
Tool: "Water jug lemma" (a.k.a. geometric distribution)
If $P(1 \text{ try succeeds}) = p$, then $\mathbb{E}(\#\text{tries until the 1st success}) = 1/p$.
$\mathbb{E}(\#\text{tries}) \le 3$, so $\mathbb{E}(\text{time}) \in O(t(n))$.

Theorem: ZPP = RP ∩ co-RP.

Proof: ① ZPP ⊆ RP: use worst-case def. of ZPP, translate MAYBE to NO.
② ZPP ⊆ co-RP: the same, but MAYBE → YES.
③ RP ∩ co-RP ⊆ ZPP: let M_1 be the machine witnessing $L \in RP$, M_2 for $L \in$ co-RP.
Run both. If they agree, use their answer. Otherwise return MAYBE.
(!) both cannot be wrong simultaneously → if they agree, the answer is right.
$P(\text{MAYBE}) \le \frac{1}{3}$ (if $\alpha \in L$, M_2 cannot fail, if $\alpha \notin L$, M_1 cannot fail)

Exercise: What happens if we modify def. of BPP or RP to use expected time and/or MAYBE?

Inclusions:



Excercise: Define class PP: $L \in PP \equiv \exists M$ PTM running in w.c. time $\mathcal{O}(poly(n))$ s.t. $P(L(\alpha) = M(\alpha)) > 1/2$ for all α.

beware: amplification doesn't work for PP (not exponentially!)

Show that: $NP \subseteq PP$, $co\text{-}NP \subseteq PP$, $BPP \subseteq PP$, $PP \subseteq PSPACE$.

Theorem: $BPP \subseteq P/poly$.

Proof: For inputs of size n: iterate (amplify) $O(n)$ times to get $P(\text{error}) \le \frac{1}{2} \cdot 2^{-n}$

Let $L \in BPP$

Let $r :=$ # random bits used by the machine (certificate size)

For a fixed input α: $P_{\beta \in \{0,1\}^r}(V(\langle \alpha, \beta \rangle) \neq L(\alpha)) \le \frac{1}{2} \cdot 2^{-n} \Rightarrow$ #"bad" certs for which this happens $\le 2^r \cdot \frac{1}{2} \cdot 2^{-n}$

↳ taking union over all α: # bad certs $\le 2^r \cdot \frac{1}{2} \cdot 2^{-n} \cdot 2^n = \frac{1}{2} \cdot 2^r < 2^r$

$\Rightarrow$ there exists a certificate which is good for all inputs: this will be the advice.

So our algorithm just calls V on ⟨input, advice⟩. This implies $L \in P/poly$.

Notes: It is known that $BPP \subseteq \Sigma_2^P \cap \Pi_2^P$ (Sipser-Gács theorem) ← this is stronger than $BPP \subseteq PSPACE$.

There are no known BPP-complete problems nor hierarchy theorems. ← BPP is a "semantic" class, so diagonalization doesn't work

It's believed that $BPP = P$ (otherwise hard-to-believe things happen)