Circuit Complexity Classes

- Analogous to $\mathrm{Dtime}(T(n))$ we have,
  <u>Size(s(n))</u> $:= \{ L \subseteq \{0,1\}^* \mid \exists\, O(s(n))$-sized boolean circuits $\{C_n\}$ solving $L \}$.

$$\underline{P/poly} := \bigcup_{c \in \mathbb{R}_{>0}} \mathrm{Size}(n^c).$$

- Analogously for arithmetic circuits we can define:
  $\underline{\text{Alg-size}(s(n))} = \{ \{f_n\}_n \mid n \in \mathbb{N},\ f_n : \mathbb{F}^n \to \mathbb{F},$ $\exists\, O(s(n))$-sized arithmetic circuit C_n computing $f_n \}$

$$\underline{\text{Alg-}P/poly} := \bigcup_{c \in \mathbb{R}_{>0}} \text{Alg-size}(n^c)$$

<u>Proposition</u>: $P \subsetneq P/poly \subseteq \mathrm{Alg}P/poly$ $(\mathbb{F} = \mathbb{F}_2)$.

Derandomization

- I.e. solving a problem without using randomness.

<u>Qn</u>: $BPP = P$?

- We will now show that this question is intimately connected to proving "lower bounds" or "hardness"!

<u>Theorem</u> (Impagliazzo & Kabanets '03): $PIT \in P \Rightarrow$ $NEXP \not\subseteq P/poly$ or $per \notin AlgP/poly$.

<u>Remark</u>: • <u>per</u> is the functional problem of computing the <u>permanent</u> of a given matrix $A_{n \times n}$.

• $\underline{per(A)} := \sum_{\sigma \in Sym(n)} A_{1\sigma(1)} \cdots\cdots A_{n\sigma(n)}$

• Here we are interested in $\mathbb{F} = \mathbb{Q}$.

• The theorem connects "algorithm" with "nonexistence" of one.

- The involved proof depends on several older results.

per as oracle

Lemma 1: PIT $\in$ P & per $\in$ AlgP/poly $\Rightarrow P^{per} \subseteq NP$.

Proof:

- Idea — "Guess" the poly-sized circuit for per. (& verify via self-reducibility)
- We can expand the permanent of an $n \times n$ matrix, $per_n(A)$, by the first row:

$$per_n(A) = \sum_{i \in [n]} A_{1i} \cdot per_{n-1}(A'_{1i}) \;,$$

where A'_{1i} is the submatrix of A after removing the first row & the i-th column.

- Given a circuit $C_{(n-1)^2}$ for per_{n-1}, we can "guess" a circuit C_{n^2} for $per_n(A)$, & using PIT algorithm verify

$$C_{n^2}(A) \stackrel{?}{=} \sum_{i \in [n]} A_{1i} \cdot C_{(n-1)^2}(A'_{1i}) .$$

- This guess-&-verify process proves $C_{n^2}(A) = per_n(A)$, by induction on n.

- $\Rightarrow$ For any $L \in P^{per}$ we can first guess the poly-sized circuit C_{n^2} for per_n, verify using PIT, & use C_{n^2} instead of the oracle.

$\Rightarrow L \in NP \Rightarrow P^{per} \subseteq NP$. $\square$

- The "strange" assumption of $per \in AlgP/poly$ gives the strange conclusion $P^{per} \subseteq NP$.

- We now intend to make another strange assumption ($NEXP \subseteq P/poly$) & deduce that $NEXP \subseteq P^{per} \subseteq NP$. (nondet. time hierarchy)

This contradiction will prove the main theorem.

- This will require a crash course in

quantifier-based & interaction-based complexity classes.

Definition: • $\Sigma_0 := P$, $\Sigma_1 := NP$, $\Sigma_2 := NP^{NP}$, $\Sigma_3 := NP^{\Sigma_2}$, ...

oracle-based

• Formally, $L \in \Sigma_2$ if $\exists$ poly-time NDTM "solving" L using SAT as oracle.

• As having SAT as an oracle also means having $\overline{SAT}$ as an oracle, it could be shown that Σ_2 has the following

$\exists, \forall \rightarrow$ quantifier-based definition:

$L \in \Sigma_2$ if $\exists$ poly-time TM N s.t. $\forall x$,
$x \in L$ iff $\exists y_1 \forall y_2$, $N(x, y_1, y_2) = 1$.

size = poly(|x|)

• Similarly, Σ_3 can be defined via an alternating sequence of 3 quantifiers:
$\exists y_1 \forall y_2 \exists y_3$.

• Polynomial hierarchy $PH := \bigcup_{i \in \mathbb{N}} \Sigma_i$.

▷ $PH \subseteq Pspace$.

Pf: Systematically go through all the possibilities of the quantified strings. □

OPEN: $\Sigma_0 \subsetneq \Sigma_1 \subsetneq \Sigma_2 \subsetneq \ldots\ldots \subsetneq PH \subsetneq Pspace$?

– Instead of $\exists, \forall$ we could use other types of quantifiers.

Eg. '$\mathcal{M}$' -quantifier for most.

TM

Definition: • The statement "$\mathcal{M} y \in \{0,1\}^n, N(y)=1$" is true iff $\Pr_{y \in \{0,1\}^n} [N(y)=1] \geq 3/4$.

• k-alternations of $\exists$ & $\mathcal{M}$ give us the class AM[k]: $L \in AM[k]$ if $\forall x$,

$$x \in L \text{ iff } \underbrace{\mathcal{M} y_1 \exists y_2 \mathcal{M} y_3 \cdots\cdots}_{k}, N(x, y_1, \ldots, y_k) = 1$$

Arthur (verifier) –

Merlin (prover)

• Starting the alternations with '$\exists$' gives us the class MA[k].