CS 278: Computational Complexity Theory
Lecture Notes - September 2, 2025

Lecturer: Lijie Chen

Fall 2025

Theorem 1 (Non-deterministic Time Hierarchy Theorem)**.** *Let* $f, g : \mathbb{N} \to \mathbb{N}$ *be time-constructible functions such that* $f(n+1) = o(g(n))$*. Then*

$$\mathit{NTIME}[f(n)] \subsetneq \mathit{NTIME}[g(n)]$$

Proof. Let $M_1, M_2, M_3, \ldots$ be an enumeration of all non-deterministic Turing machines.

Let n_1 be a sufficiently large integer. For every $i \in \mathbb{N}$, we set $I_i = [n_i, 2^{f(n_i)^2}]$, and $n_{i+1} = 2^{f(n_i)^2} + 1$.

We will define our hard language $H \in \text{NTIME}[g(n)]$ such that for every $i \in \mathbb{N}$, if M_i runs in $\text{NTIME}[f(n)]$ time, there exists an $n \in I_i$ such that $H(1^n) \neq M_i(1^n)$, where 1^n denotes the all-1 string of length n.

For $n \in [n_i, 2^{f(n_i)^2})$, we set

$$H(1^n) = U(\langle M_i \rangle, 1^{n+1}, f(n+1))$$

That is, $H(1^n)$ simulates M_i on input 1^{n+1} for $f(n+1)$ steps.

Note that for the sake of contradiction, if we assume M_i runs in $\text{NTIME}[f(n)]$ time and M_i computes the same language as H, then we know that

$$H(1^n) = H(1^{n+1}) \text{ for } n \in [n_i, 2^{f(n_i)^2})$$

In particular, this means that

$$H(1^{n_i}) = H(1^{2^{f(n_i)^2}})$$

Note that we haven't defined $H(1^{2^{f(n_i)^2}})$ yet. We set $H(1^{2^{f(n_i)^2}}) = 1$ if and only if M_i rejects the input 1^{n_i}. Note that this is possible, since we can check whether M_i rejects the input 1^{n_i} in *deterministic* $2^{O(f(n_i))} \leq 2^{f(n_i)^2}$ time.

Therefore, we have

$$H(1^{2^{f(n_i)^2}}) = \neg M_i(1^{n_i}).$$

This implies $H(1^{n_i}) = \neg M_i(1^{n_i})$, which contradicts the assumption that M_i computes H. □

Remark. $2^{f(n_i)^2}$ is not important, anything a bit larger than $2^{f(n_i)}$ is enough.