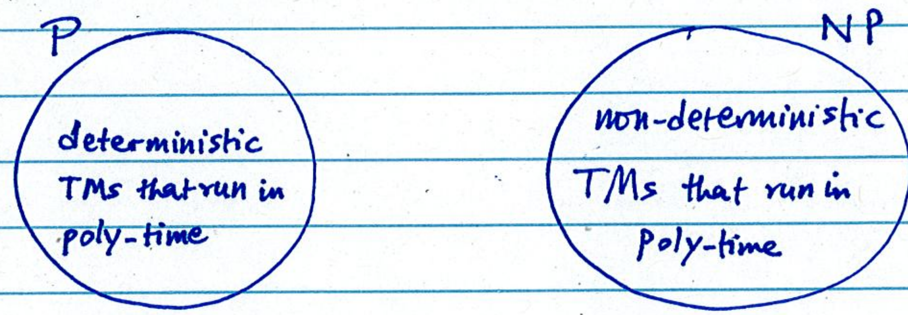
Diagonalization

Diagonalization was one of the earliest techniques employed (since 1970s) to resolve the $P \stackrel{?}{=} NP$ question. The general idea is the following:



We want to show the existance of a <u>non-det.</u> TM that runs in poly-time & differs from every det. poly-time TM in at least one input-output pair. At first glance, it seems like this is something we should be able to achieve using diagonalization.

Despite initial successes, it became clear in 70s that there were some serious obstacles to using diagonalization to resolve $P \stackrel{?}{=} NP$.

In this Chapter we'll study some early successes obtained by applying diagonalization:

- Time Hierarchy Theorem
- Nondeterministic Time Hierarchy Theorem
- Ladner's Theorem: If $P \neq NP$, then there exists a language $L \in NP \setminus P$ that is not NP-complete.

(Deterministic Time Hierarchy Theorem)

$\mathrm{DTIME}(n) \subsetneq \mathrm{DTIME}(n^2)$.

- In other words, $\mathrm{DTIME}(n)$ is a <u>strict</u> subset of $\mathrm{DTIME}(n^2)$.
- In other words, there is a language $L \in \mathrm{DTIME}(n^2) \setminus \mathrm{DTIME}(n)$.

PROOF (by diagonalization):

Consider a Turing machine D that is deterministic and works as follows:

Turing Machine D

INPUT: $x \in \{0,1\}^*$

ALGORITHM

1. Use the Universal Turing Machine $\mathcal{U}$ to simulate M_x (TM encoded by binary string x) for $n = |x|$ steps.

2. If $\mathcal{U}$ halts and outputs $b \in \{0,1\}$ then D outputs $1-b$.

3. Otherwise (if $\mathcal{U}$ does not halt within this time) output 0.

Note that the running of D is n^2 by using the "relaxed" version of the theorem on the existence of a Universal Turing Machine. This is because the theorem tells us that there is a Universal Turing Machine $\mathcal{U}$ that halts in T^2 steps on input $(\lfloor M \rfloor, x)$ if M halts on input x in T steps. Note that the fact that $\mathcal{U}$ uses a counter to abort if necessary adds very little overhead to the simulation time. (Check this!)

Let L be the language accepted by D (i.e., $L = \{x \in \{0,1\}^* \mid D(x) = 1\}$). Since D runs in time n^2, $L \in DTIME(n^2)$.

We will now show that $L \notin DTIME(n)$. Suppose (for the sake of obtaining a contradiction) that $L \in DTIME(n)$. Then there is a TM M that runs in time n such that $M(x) = D(x)$ $\forall x \in \{0,1\}^*$. Now Suppose we provide $\lfloor M \rfloor$ as input to D. Since M runs in time n, M will be simulated to completion (on input $\lfloor M \rfloor$) and D will output a bit that is distinct from what M would have output. In other words, $M(\lfloor M \rfloor) \neq D(\lfloor M \rfloor)$ contradicting our supposition that M & D accepted the same language. Thus $L \notin DTIME(n)$. $\square$

Core of the proof is that we have constructed a machine D that runs in n^2 time and differs from <u>every</u> machine that runs in n time on at least one bit.

In fact, if we use the stronger version of the Universal Turing Machine Theorem we get a stronger Time Hierarchy theorem.

<u>Time Hierarchy Theorem</u> (Hennie & Stearns 1965):
Suppose $f, g : \mathbb{N} \to \mathbb{N}$ are <u>time constructible</u> functions satisfying $f(n) \cdot \log(n) = o(g(n))$ then

$$DTIME(f(n)) \subsetneq DTIME(g(n)).$$

(Recall that a <u>time constrictible function</u> is a function $f : \mathbb{N} \to \mathbb{N}$ that can be computed in $O(f(n))$ time. Think about where this fact might be needed in the proof of the Time Hierarchy Theorem.)

<u>Non-deterministic Time Hierarchy Theorem</u> (Cook, 1972)
If f, g are time constructible functions satisfying $f(n+1) = o(g(n))$ then

$$NTIME(f(n)) \subsetneq NTIME(g(n)).$$

First we should ask if the earlier proof we used would work in the non-deterministic setting as well.