The nondeterministic time hierarchy theorem is even "tighter" than the deterministic hierarchy theorem.

(Cook 1972) Nondeterministic Hierarchy Theorem

If f & g are time constructible functions satisfying $f(n+1) = o(g(n))$ then

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

Comments:

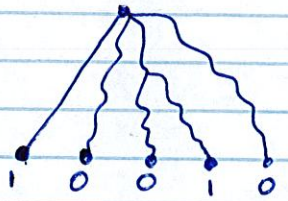
(1) Say $f(n) = n^2$. Then $f(n+1) = n^2 + 2n + 1 = \Theta(n^2)$. For $f(n+1) = o(g(n))$, we can let $g(n)$ be $n^2 \log n$ or $n^2 \cdot \log\log n$, etc. However, $g(n)$ cannot be $10n^2$ or $n^2 + 20n$, etc.

(2) The "tightness" of this result relative to the Deterministic Time Hierarchy Theorem is due to the fact that there is a Universal NDTM that can simulate an arbitrary given NDTM very efficiently. Specifically, there is a Universal NDTM NU that takes input $\langle \alpha, x \rangle$ and if M_α halts on x in time T then NU simulates M_α on x in $C \cdot T$ steps for some const. C (that depends only on α, but not on x).

[See Problem 2.6]

(3) The proof of the Deterministic Hierarchy Theorem constructs a DTM that simulates a machine and "flips" the output of the simulated machine. For NDTMs there seems to be no easy way to "flip" the output. The simple approach of requiring that if the NDTM is in a state q_{accept} it should move to q_{reject} & vice versa does not work.

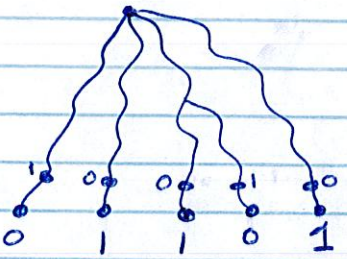
M's execution on input x



Since some branch accepts x, the NDTM accepts x.

flipping bits →

M's execution on input x with bits flipped at the end



There are branches that still accept x & hence x is accepted by M. Hence, flipping bits in this manner did not cause x to be rejected.

PROOF:

We'll specify f later.

- Let $f: \mathbb{N} \to \mathbb{N}$ be a rapidly growing function. In fact, f grows so fast that $f(i+1)$ is exponentially larger than $f(i)$ (i.e., $f(i+1) = 2^{f(i)}$ or something like that).

- We will construct a NDTM D that takes an input $x \in \{0,1\}^*$. D rejects all inputs that are not of the form 1^n.

- On an input 1^n, D first computes i such that

$$f(i) < n \leq f(i+1).$$

Note that since f is a rapidly growing function, i is quite small relative to n.

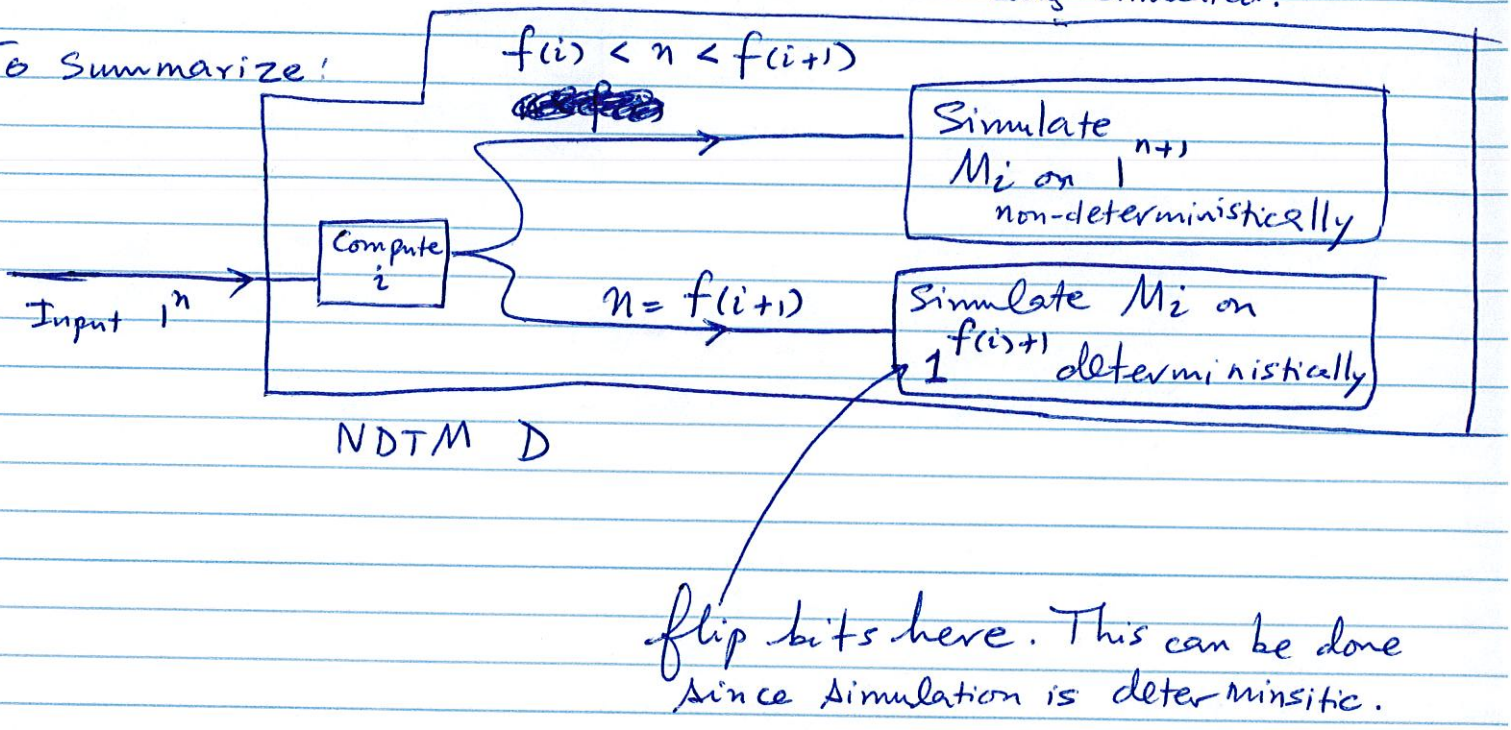
- D then simulates the machine M_i (i.e., the NDTM whose binary encoding is equivalent to the decimal i).

- If $f(i) < n < f(i+1)$, then D simulates M_i in a non-deterministic manner on the input 1^{n+1}. D lets the simulation go for $n^{1.1}$ steps. D outputs M_i's answer if it halts; otherwise D halts & accepts.

- If $n = f(i+1)$, then D behaves differently. D simulates M_i on input $1^{f(i)+1}$ deterministically. In otherwords D explicitly simulates all branches of M_i's execution. D accepts iff M_i rejects $1^{f(i)+1}$ in time $(f(i)+1)^{1.1}$.

This refers to the length of each path in M_i's execution being simulated.

To Summarize:



flip bits here. This can be done since simulation is deterministic.

What is the running time of D?

- Time to compute i
- $n^{1.1}$ steps for non-deterministic simulation
- $2^{(f(i)+1)^{1.1}}$ for deterministic simulation.

If we assume that f grows rapidly enough then $n = f(i+1) > 2^{(f(i)+1)^{1.1}}$. Even if we assume a logarithmic overhead for deterministic simulation, the running time is at most $n^{1.1}$.

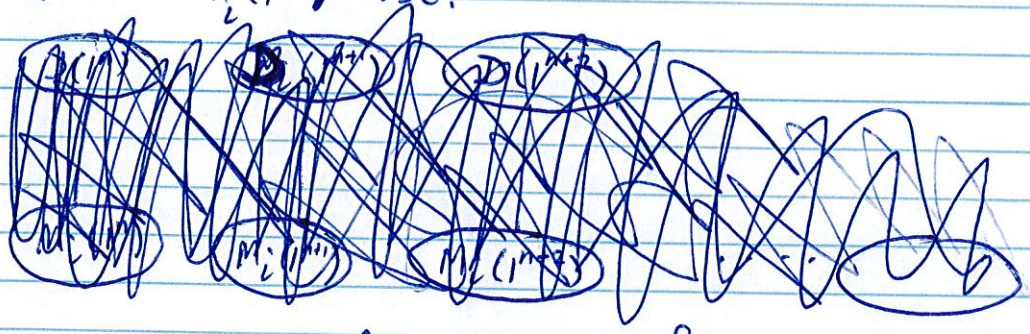
We can let $f(1) = 2$, $f(i+1) = 2^{f(i)^2}$. We can then check that i can be computed in $n^{1.5}$ time.

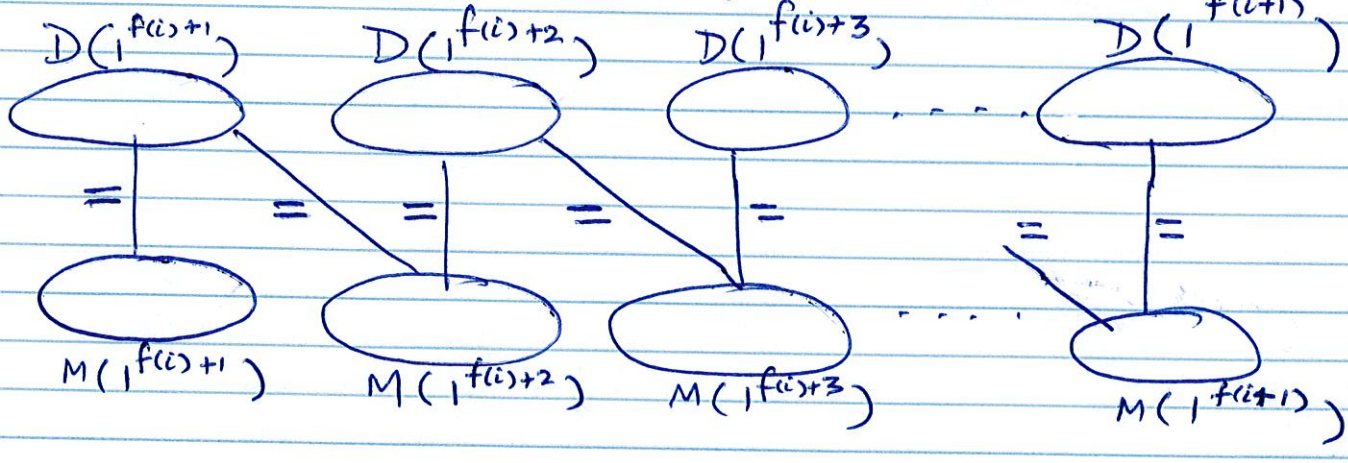
Let $L = L(D)$, i.e., the language decided by D. Then $L \in NTIME(n^{1.5})$.

We now show that $L \notin NTIME(n)$. To obtain a contradiction suppose that $L \in NTIME(n)$ & there is an NDTM M such that $L = L(M)$. We know that $M = M_i$ for some natural number i. Let us now think about how D behaves on input 1^n where $f(i) < n \leq f(i+1)$.

On input 1^n, $f(i) < n < f(i+1)$, D simulates M_i on 1^{n+1}. Note that M_i runs in $n+1$ time (non-deterministically) & so M_i is simulated to completion by D, since $n+1 < n^{1.1}$. Hence, $D(1^n) = M_i(1^{n+1})$.

Note that by our assumption $L(D) = L(M_{i+1})$ and this implies that $D(1^n) = M_i(1^n)$ also.





On input 1^n, $n = f(i+1)$, D simulates M_i on $1^{f(i)+1}$ and "flips" bits. Again, since M_i runs in linear time, every path in M_i's computation is at most $f(i)+1$ long. Hence, D simulates M_i to completion and correctly flips its bits.

In other words, $D(1^{f(i+1)}) \neq M(1^{f(i)+1})$. This contradicts the fact $M(1^{f(i)+1}) = D(1^{f(i+1)})$ - by transitivity from the previous figure

□