Complexity Theory I Jan – May, 2023

Lecture 3: Time Hierarchy Theorem

Lecturer: Partha Mukhopadhyay *Scribe: Soham Chatterjee*

1 Diagonalization

TMs can be encoded efficiently

Theorem 1 Cantor's Idea

Reals in $(0, 1)$ are uncountable

Proof. Otherwise let $r_1, r_2, r_3, \dots$ be an enumeration of the reals in $(0, 1)$.

$$r_i = \sum_{j\geq 1} r_i[j]2^{-j}$$

where $r_i[j] \in \{0, 1\}$.

Define r such that $r[j] = 1 - r_j[j]$. So,

$$r = \sum_{j\geq 1} r[j]2^{-j}$$

r is not in the enumeration list. Otherwise let $r = r_k$ for some $k \in \mathbb{N}$. But by construction $r[k] = 1 - r_k[k]$. $\square$

2 Time Hierarchy Theorem

$\boldsymbol{TIME(n^3)}$:= Set of problems which can be solved by a DTM in time $O(n^3)$ where the input length $= n$.

Time Hierarchy Theorem says that

$$TIME(n^2) \subsetneq TIME(n^3)$$

$$TIME(g(n)) \subsetneq TIME(f(n))$$

where $g(n) \approx o(f(n))$

Definition 1: Time Constructible Function

Let $t : \mathbb{N} \to \mathbb{N}$ and $\exists\ n_0$ such that $t(n) \geq n \log n$ for $n \geq n_0$. Then we say that t is time constructible if on input 1^n the binary value of $t(n)$ can be computed in $O(t(n))$ time using a DTM

Example: $n \log n$, n^2, n^3, $n\sqrt{n}$, 2^n

T

Example 1 (Non-Time Constructible Function)

$f : \mathbb{N} \to \mathbb{N}$.

$$f(n) = \begin{cases} n^2 & \text{if } n \text{ encoded in binary a TMM which halts on all inputs} \\ n^2 + 1 & \text{otherwise} \end{cases}$$

Theorem 2 Time Hierarchy Theorem [Sip13]

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. Then there exists a language $L \in TIME(t(n))$ such that $L \notin TIME\left(o\left(\frac{t(n)}{\log(t(n))}\right)\right)$

Idea: We construct a TM D that decides a language A in time $O(t(n))$, whereby A cannot be decided in $o(t(n)/\log t(n))$ time. Here, D takes an input w of the form $\langle M\rangle 10^*$ and simulates M on input w, making sure not to use more than $t(n)$ time. If M halts within that much time, D gives the opposite output.

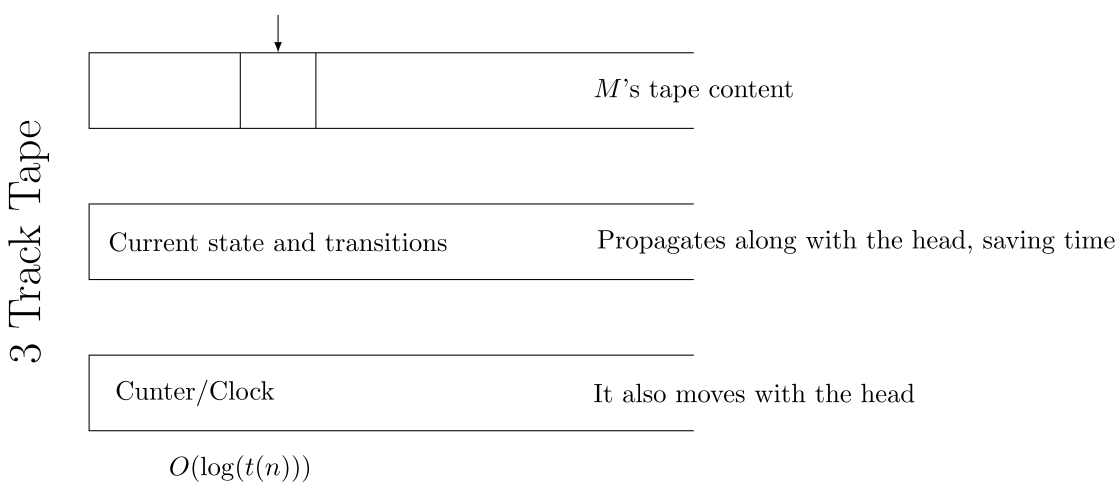
The important difference in the proof concerns the cost of simulating M while, at the same time, counting the number of steps that the simulation is using. Machine D must perform this timed simulation efficiently so that D runs in $O(t(n))$ time while accomplishing the goal of avoiding all languages decidable in $o(t(n)/\log t(n))$ time. For space complexity, the simulation introduced a constant factor overhead, as we observed in the proof of Theorem 9.3. For time complexity, the simulation introduces a logarithmic factor overhead. The larger overhead for time is the reason for the appearance of the $1/\log t(n)$ factor in the statement of this theorem. If we had a way of simulating a single-tape TM by another single-tape TM for a prespecified number of steps, using only a constant factor overhead in time, we would be able to strengthen this theorem by changing $o(t(n)/\log t(n))$ to $o(t(n))$. No such efficient simulation is known.

Proof. The following $O(t(n))$ time algorithm D decides a language A that is not decidable in $o(t(n)/\log t(n))$ time.

Turing Machine B

1. Input w of length $|w| = n$
2. Compute $\frac{t(n)}{\log n}$ and make a counter for $\frac{t(n)}{\log n}$ using $\log\left(\frac{t(n)}{\log n}\right) \approx \log(t(n))$ bits.

   Decrement the clock in every step
3. Check if $w = \langle M\rangle 10^*$ where M is an encoding of a Turing Machine, else reject.
4. Simulate M on w. If M halts within the clock, B does opposite to M
5. Halts and reject.



$\square$

References

[Sip13] Michael Sipser. *Introduction to the Theory of Computation.* Cengage India Private Limited, third edition, 2013.