Deterministic Time Hierarchy Theorem

Marco L Carmosino

September 25, 2023

1 Introduction

Strictly more computational resources can solve strictly more problems. Variations of this theorem initiated research in computational complexity theory [HS65]. We will prove this *hierarchy theorem* for deterministic time. Our quantitative statement will be weaker than even the original theorem of [HS65], but our proof should be easier to translate into formal systems. All theorem numbers refer to AB unless otherwise specified. We use the following facts and notation about Turing Machines (TMs).

- Every TM can be encoded by a bitstring of finite length.
- TMs process inputs of arbitrary finite length.
- Write $M(x)$ for "the output of TM M on input $x \in \{0,1\}^*$" after M halts
- Write $\mathcal{L}(x)$ for the characteristic function of a language $\mathcal{L} \subseteq \{0,1,\#\}^*$

$$\mathcal{L}(x) = \begin{cases} 1 \text{ if } x \in \mathcal{L} \\ 0 \text{ otherwise} \end{cases}$$

2 Definitions & Tools

We define complexity classes as sets of languages that can be decided with a given amount of resources.

Definition 1 (Deterministic Time Complexity)**.** Let $\mathcal{L} \subseteq \{0,1,\#\}^*$ be a language. TM M *decides* $\mathcal{L}$ iff

$$\forall x\ M(x) = \begin{cases} 1 \iff x \in \mathcal{L} \\ 0 \iff x \notin \mathcal{L} \end{cases}$$

Let $T : \mathbb{N} \to \mathbb{N}$ be a step-counting function. M *runs in* $T(n)$*-time* iff $\forall x\ M(x)$ halts in at most $T(|x|)$ steps.

$$\mathcal{L} \in \mathsf{DTIME}[T(n)]$$
$$\iff$$
$$\exists c \in \mathbb{N},\ \text{TM } M \text{ running in } c \cdot T(n) \text{ time and deciding } \mathcal{L}$$

Theorem 1 (Efficient Universal Turing Machine (UTM))**.**
There is a TM U such that for all inputs $M, t \in \{0,1\}^*$ and $x \in \{0,1,\#\}^*$
***If** $M(x)$ halts in t steps, then*
  $U(M,x,t)$ outputs $M(x)$ in $C' t \log t$ steps
***Otherwise**,*
  $U(M,x,t)$ outputs TIMEOUT in $C' t \log t$ steps
***Where** C' depends on the number of tapes, symbols, and states of M but **NOT** the input length $|x|$*

3 Deterministic Time Hierarchy Theorem

Theorem 2 (Simplified Deterministic Time Hierarchy Theorem)**.**

$$\forall b \in \mathbb{N}\ \mathsf{DTIME}[n^b] \subsetneq \mathsf{DTIME}[n^{5b}]$$

Proof idea. We have enough time — $O(n^{5b})$ steps — to simulate and then *disagree with* every deterministic TM that runs in only $O(n^b)$ steps. We will describe a language H_b where part of each input x is treated as the code of a TM M, and $H_b(x) = \neg M(x)$ when M runs in $O(n^b)$ time. This will make it impossible for such M to agree with H_b on all inputs. When the code of M appears in a long enough input x, it is guaranteed to disagree with H_b. □

Proof. We'll begin by bringing the theorem statement closer to a sentence of first-order logic, to structure a proof that will be straightforward to formalize.

$$\forall b \in \mathbb{N}\ \ \exists H_b \in \mathsf{DTIME}[n^{5b}]\ \ \forall \mathcal{L} \in \mathsf{DTIME}[n^b]\ \ \exists n_0 \in \mathbb{N}\ \ \exists x \in \{0,1,\#\}^* \tag{1}$$

$$H_b(x) \neq \mathcal{L}(x) \tag{2}$$

We will discharge each quantifier above in turn, while tracking the work required to construct intermediate objects. First, let $b \in \mathbb{N}$ be arbitrary. Next, we define our "diagonal" TM D.

Algorithm 1 Diagonal Machine $D(x)$

```
n ← |x|
if x matches M#0* where M ∈ {0,1}* then
    Run U(M, x, n^{3b}) for at most n^{5b} steps
    cases
        U TIMEOUT ↦ output REJ
        M TIMEOUT ↦ output REJ
        M ACC ↦ output REJ
        M REJ ↦ output ACC
else
    output REJ
```

Let H_b denote the language of D. We know $H_b \in \mathsf{DTIME}[n^{5b}]$ because $|x|$, pattern matching, and maintaining a step counter are in $\mathsf{DTIME}[n^2]$. Observing the clock on U concludes the runtime upper bound.

Now fix arbitrary $\mathcal{L} \in \mathsf{DTIME}[n^b]$ witnessed by M running in time $C_M n^b$ for some constant C_M and deciding $\mathcal{L}$. All that remains is to witness the last two quantifiers of (1): a sufficiently large input length and particular input where H_b and $\mathcal{L}$ must disagree. Our proof works by simulating the execution of D.

Consider "matching" inputs of the form $M\#0^*$. If we can ensure that at least one such input avoids the TIMEOUT cases, we are done.

1. Substituting into the efficient UTM theorem, there exists C'_M depending on M but not |x| such that $U(M, x, n^{3b})$ runs in time $C'_M \times C_M \times n^{3b} \times \log(C_M n^{3b})$. Manipulating inequalities over $\mathbb{N}$, we have

   $$\exists n_1 \in \mathbb{N}\ \forall n > n_1\ n^{5b} > C'_M \times C_M \times n^{3b} \times \log(C_M n^{3b}).$$

   Therefore, on input lengths greater than n_1, U will not TIMEOUT.

2. From the definition of "runtime" M takes $C_M n^b$ steps to halt on *any* input x. Again manipulating inequalities over $\mathbb{N}$, we have

   $$\exists n_2 \in \mathbb{N}\ \forall n > n_1\ n^{3b} > C_M n^b$$

   Therefore, on input lengths greater than n_2, simulating M for n^{3b} steps will not TIMEOUT.

Let $n_0 = \max\{n_1,\ n_2\}$ and consider the matching input $x = M\#0^{n_0}$ so that $|x| > n_0$ and M is the bitstring encoding TM M. $D(x)$ will match and run U. By the two claims above, $D(x)$ does not TIMEOUT. Consider the remaining cases.

Suppose $M(x) = \text{ACC}$. Then $D(x) = \text{REJ}$ so $\mathcal{L}(x) \neq H_b(x)$.

Suppose $M(x) = \text{REJ}$. Then $D(x) = \text{ACC}$ so $\mathcal{L}(x) \neq H_b(x)$. □

4 How "Constructive" is the DTIME Hierarchy Theorem?

Observe that this proof gives us more than was promised. Let x match $M\#0^\ell$ where $\ell \geq n_0 - |M|$. Any such input disagrees with H_b, if M runs in $O(n^b)$ time. This means we can produce mistakes computing H_b at *every* sufficiently long input length.

Furthermore, mistakes are easy to find: we need only print the encoding of M and then append enough padding to reach the desired input length. This error-printer only depends on a constant amount of information about each language in $\mathsf{DTIME}[n^b]$, and correctness follows from our proof of the deterministic time hierarchy theorem.

This suggests the definition of an algorithmic task that could *accompany* or be *extracted from* the proof of a lower bound. Such efficient witnessing of the errors guaranteed to exist by a separation (Equation 1) was originally introduced by [Kab01] and is key to many results in (meta-)complexity. We give a simplified variant of Definition 1.1 in [Che+22].

Definition 2 (Refuter)**.** For a language $\mathcal{L}$ and a TM M, a *refuter* for $\mathcal{L}$ against M is a polynomial time algorithm R that, given 1^n as input, prints an n-bit string witnessing that M fails to decide $\mathcal{L}$ almost everywhere. Formally,

$$\exists n_0 \in \mathbb{N}\ \forall n > n_0 \quad \mathcal{L}(R(1^n)) \neq M(R(1^n))$$

Definition 3 (Constructive Separation)**.** For a complexity class $\mathcal{C}$ and a languge $\mathcal{L}$, a *constructive separation* of $\mathcal{L} \notin \mathcal{C}$ means that, for every $\mathcal{C}$-algorithm M attempting to decide $\mathcal{L}$, there is a refuter for $\mathcal{L}$ against M.

Notice how a refuter is allowed to depend on the target machine. We need not refute every $\mathcal{C}$-machine at once, but are allowed to depend on the code of each individual machine. Inspecting the proof of Theorem 2, we have

Corollary 1 (Refuters for the Diagonal Language)**.**
There is a constructive separation of $H_b \notin \mathsf{DTIME}[n^b]$ for every b.

Introducing non-uniformity complicates matters. Is a refuter still meaningful? Yes — if we give the refuter access to the *same* advice sequence as the target machine, this ensures a "fair fight" between resource bounds and still distinguishes between proofs of separations that can be transformed into efficient algorithms, and those that cannot.

Definition 4 (Refuter Against Non-Uniform Machines)**.** For a language $\mathcal{L}$ and advice-taking TM M with advice sequence $\alpha_1, \alpha_2, \ldots \alpha_n \ldots$ a *refuter* for $\mathcal{L}$ against M with advice sequence α is a polynomial time algorithm R that, given $\langle 1^n, \alpha_n \rangle$ as input, prints an n-bit string witnessing that M fails to decide $\mathcal{L}$ almost everywhere. Formally,

$$\exists n_0 \in \mathbb{N}\ \forall n > n_0 \quad \mathcal{L}(R(1^n, \alpha_n)) \neq M(R(1^n, \alpha_n))$$

Now, the refuter is allowed to depend on both the code of M and the sequence of advice that M sees at each input length.

References

[Che+22] Lijie Chen et al. "Constructive Separations and Their Consequences". In: *CoRR* abs/2203.14379 (2022). DOI: 10.48550/arXiv.2203.14379. arXiv: 2203.14379. URL: https://doi.org/10.48550/arXiv.2203.14379.

[HS65] J. Hartmanis and R. E. Stearns. "On the computational complexity of algorithms". In: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306.

[Kab01] Valentine Kabanets. "Easiness Assumptions and Hardness Tests: Trading Time for Zero Error". In: *J. Comput. Syst. Sci.* 63.2 (2001), pp. 236–252. DOI: 10.1006/jcss.2001.1763. URL: https://doi.org/10.1006/jcss.2001.1763.