1 The time hierarchy theorem

Theorem 1. *There's a universal $c > 1$ such that for any time-constructible $T(n) \geq n$ we have $\mathcal{DTIME}[T^c] \not\subseteq \mathcal{DTIME}[T]$.*

Proof. Let U be the universal machine, and let $c_U > 1$ be the constant such that if $M(x)$ runs in time T then $U(\langle M \rangle, x)$ runs in time at most T^{c_U}.

Defining a language $L \subseteq \{0,1\}^*$. For convenience, we define L by defining an indicator function $L\colon \{0,1\}^* \to \{0,1\}$ (i.e., x is in the language iff the indicator function outputs 1 on x). For any $x \in \{0,1\}^*$, if x isn't a valid encoding $\langle M \rangle$ of a TM, then $L(x) = 0$. Otherwise, consider the computation of $U(\langle M \rangle, \langle M \rangle)$ truncated after $T(|\langle M \rangle|)^{c_U}$ steps. If U halted and returned a bit b at that point (or before), then $L(x) = 1 - b$. Otherwise (if U didn't halt), then $L(x) = 0$.

A lower bound: $L \notin \mathcal{DTIME}[T]$. Assume towards a contradiction that a T-time M decides L. Then $M(\langle M \rangle)$ stops after at most $T(|\langle M \rangle|)$ steps and outputs some bit b, in which case $U(\langle M \rangle, \langle M \rangle)$ stops after at most $T(|\langle M \rangle|)^{c_U}$ steps and outputs b. Hence $L(\langle M \rangle) = 1 - b$. Then, we have

$$b = M(\langle M \rangle) = L(\langle M \rangle) = 1 - b\,,$$

a contradiction (where the middle inequality is since M decides L).

An upper bound: $L \in \mathcal{DTIME}[T^c]$. We design a three-tape M_L that decides L in time $\mathrm{poly}(T)$ (and rely on the simulation of three-tape TMs by standard TMs).

If the input $x \in \{0,1\}^n$ isn't a valid encoding $\langle M \rangle$ of a TM, M_L rejects. Otherwise, M_L runs $U(\langle M \rangle, \langle M \rangle)$ while counting its steps up to $T(n)^{c_U}$. (Specifically, M_L first writes the binary representation of the integer $T(n)^{c_U}$ on the second tape, and initializes the third tape to the value zero. After simulating each step of U the machine M_L increments the counter on the third tape and compares it to the value on the second tape, stopping when the two values are equal.) At this point, if U outputted a bit b then M_L outputs $1 - b$, otherwise it outputs 0.

By definition, M_L decides L. To see that M_L runs in time $\mathrm{poly}(T)$, we check that each of its steps runs in such time:

- Checking if the input is a valid encoding of a TM can be done in time $O(n \cdot \log(n))$,[1] which is at most $O(T(n) \cdot \log(T(n)))$ because $T(n) \geq n$.

[1] Specifically, we first check whether the input parses correctly as Σ, S, δ, and then check that for every pair $s \in S, \sigma \in \Sigma$ there is a valid entry in the description of δ corresponding to input (s, σ). Naively this can be done in cubic time, and by requiring that Σ, S, δ will be written in a suitably ordered way this can be done in time $O(n \cdot \log(n))$. (E.g., we first check that the input length is at least $|S| \cdot |\Sigma|$, otherwise we reject. Then we check that S and Σ are written in lexicographic order, and that the entries in the description of δ are written in lexicographic order of the inputs to δ, otherwise we reject. The ordering facilitates scanning over all pairs (s, σ) in linear time in the input length, for each entry we check that the output of δ (whose length is logarithmic in the input length) is of valid format.)

- Computing $T(n)^{c_U}$ can be done in time $O(T(n)) + \text{polylog}(T(n))$, because T is time-constructible and raising the integer $T(n)$ to the power c_U can be done in polynomial time in the integer's description length, which is $O(\log(T(n)))$.
- The machine simulates U for at most $T(n)^{c_U}$ steps. For each step, it increments a counter (whose value is at most $T(n)^{c_U}$) and compares it to the integer $T(n)^{c_U}$. Both the counter and the latter integer have description length at most $O(\log(T(n)))$, and incrementing and comparing can be done in linear time $O(\log(T(n)))$. Hence, simulating U takes time $O(T(n)^{c_U} \cdot \log(T(n)))$.

Each of the three steps above runs in time at most $\text{poly}(T(n))$, and the final step involves erasing the main worktape (whose content has at most $\text{poly}(T)$ bits), writing a bit and halting. So the overall running time is at most $\text{poly}(T(n))$. ■