The Time Hierarchy Theorem

Does more time mean more language recognition power? That is, if $g(n)$ grows faster than $f(n)$, does DTIME($g(n)$) contain languages that are not in DTIME($f(n)$)? Here, DTIME($f(n)$) denotes the set of decision problems that can be solved in $O(f(n))$ time by a deterministic Turing machine. The Time Hierarchy Theorem shows that the answer to this question is "yes" if g grows sufficiently faster than f.

Theorem 1 *(Time Hierarchy Theorem) Let $g(n)$ be a time constructible function and let $f(n)$ be such that $f(n) \log f(n) = o(g(n))$. Then*

$$\textit{DTIME}(g(n)) - \textit{DTIME}(f(n)) \neq \emptyset.$$

Here by "time constructible" we mean that there is a Turing machine that, on input 1^n, writes $1^{g(n)}$ on its tape in $O(g(n))$ time.

Proof: We will construct a DTM M_g that runs in $O(g(n))$ time, such that if $L(M_g)$ is the language accepted by M_g (set of binary strings on which M_g outputs "yes") then $L(M_g) \notin$ DTIME($f(n)$). We'll use a *diagonalization* argument: we construct M_g so that for every TM M_x that runs in $O(f(n))$ time, M_g *does the opposite* of M_x on some input w. That is, M_g outputs "yes" on w if and only if M_x outputs "no" w.

To do this, M_g will need to be able to simulate other TMs. We'll use the Hennie-Stearns univeral TM simulation for this purpose. This universal TM takes as input a description x of a Turing machine M_x, plus another input w, and simulates M_x on w. If M_x runs in $cf(|w|)$ time on input w then the simulation takes at most $c' f(|w|) \log f(|w|))$ steps for some other constant c' that depends on c (see Theorem 1.13 of Arora and Barak).

A simple approach to diagonalization is for M_g to do the opposite of M_x on input x. That won't work in this proof because the proof details are sensitive to time. Instead, M_g simulates M_x on inputs of the form $w = 0^i 1x$. M_g is guaranteed to complete the simulation (and thus is able to do the opposite of M_x) for sufficiently large i. (The technique of concatenating 0^i to the input is called "padding" and is handy in other situations too.)

M_g does the following on input w.

- If $w = 0^i 1x$ for some $i \geq 0$, where x is the binary encoding of any Turing machine:
  1. Mark off $g(|w|)$ cells on a tape. This is possible since g is time constructible.
  2. Let M_x be the TM encoded by x. Using the universal TM of Hennie and Stearns, simulate M_x on w until one (or both) of the following conditions is satisfied:
     (a) M_x halts, or
     (b) The number of simulation steps has reached the limit $g(|w|)$ (use the tape marked in step 1 to check this).
  3. Output "no" if M_x output "yes" in step 2 (a)
- Output "yes"

We first show that M_g runs in $O(g(n))$ time. Checking whether x is a valid encoding of a TM can be done in $O(|x|) = O(|w|)$ time, since a TM encoding is simply a list of transition triples in a fixed format. Step 1 can be done $O(g(|w|))$ time because we assume that g is time-constructible, and Step 2 stops after $g(|w|)$ steps. The remaining steps take $O(1)$ time. So the overall time is $O(g(n))$.

To complete the proof, we show that $L(M_g) \not\subseteq$ DTIME($f(n)$). Let x be the encoding of a TM M_x that runs in time $O(f(n))$. Then there is some constant c such that for sufficiently large i, M_x on input $w = 0^i 1x$ takes at most $cf(|w|)$ steps. The Hennie-Stearns universal TM can simulate $cf(|w|)$ steps of M in at most $c' f(|w|) \log f(|w|))$ steps, for some other constant c' that depends on c. Since $f(n) \log(f(n)) = o(g(n))$, for sufficiently large i we know that $c' f(|w|) \log(f(|w|)) \leq g(|w|)$.

Therefore, for sufficiently large i, the simulation of M_x on input $w = 0^i 1x$ halts due to condition 2 (a), before condition 2 (b) is reached. (The 0^i "padding" of the input ensures that M_g can complete the simulation of M_x the within the time limit of $g(|w|)$.)

In this case, M_x outputs "yes" on w if and only if M_g outputs "no" on w. Therefore, $L(M_g) \neq L(M_x)$. Since M_x is an arbitrary TM whose running time is $O(f(n))$, it must be that $L(M_g) \notin$ DTIME($f(n)$), completing the proof.