1 Hierarchy Theorems

Reading: Arora-Barak 3.1, 3.2.

“More (of the same) resources ⇒ More Power”

Theorem 1 (Time Hierarchy) If \( f, g \) are nice (“time-constructible”) functions and \( f(n) \log f(n) = o(g(n)) \) (e.g. \( f(n) = n^2, g(n) = n^3 \)), then \( \text{DTIME}(f(n)) \subsetneq \text{DTIME}(g(n)) \).

Nice functions: \( f \) is time-constructible if:

1. \( 1^n \rightarrow 1^{f(n)} \) can be computed in time \( O(f(n)) \)
2. \( f(n) \geq n \)
3. \( f \) is nondecreasing \( (f(n + 1) \geq f(n)) \)

The proof will be by diagonalization, like what is used to prove the undecidability of the Halting Problem. Specifically, we want to find TM \( D \) such that:

1. \( D \) runs in time \( O(g(n)) \)
2. \( L(D) \neq L(M) \) for every TM \( M \) that runs in time \( f(n) \).

First recall how (in cs121) an undecidable problem is obtained via diagonalization.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M_2 )</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Index \( (i, j) \) of the array is the result of \( M_i \) on \( x_j \), where \( M_1, M_2, \ldots \) is an enumeration of all TMs and \( x_1, x_2, \ldots \) is an enumeration of all strings. Our undecidable problem \( D \) is the complement of the diagonal, i.e. \( D(x_i) = \neg M_i(x_i) \). However, now we want \( D \) to be decidable, and even in \( \text{DTIME}(g(n)) \). This is possible because we only need to differ from TMs \( M_i \) that run in time \( f(n) \).

First Suggestion: Set \( D(x_i) = \neg M_i(x_i) \), where \( M_1, M_2, \ldots \) is an enumeration of all time \( f(n) \) TMs.

Problem: There is no way for \( D \) to tell whether a given TM \( M_i \) actually is a time \( f(n) \) algorithm. (Indeed, this is an undecidable problem...)

Better Suggestion: Enumerate all TMs, but only simulate for \( g(n) \) steps.

Proof:
D(x):

1. Compute \( g(n) \), where \( n = |x| \).

2. Run \( U([M_x], x) \) for at most \( g(n) \) steps and output opposite. If it does not complete, output anything.

By time-constructibility, \( D \) runs in time \( O(g(n)) \).

Suppose for contradiction \( \exists \) time \( f(n) \) TM \( M \) s.t. \( L(M) = L(D) \). \( M \) is described by some string \( x \). On \( x \), \( D \) does the opposite of \( M_x = M \) on \( x \), provided that the simulation \( U([M_x], x) \) completes. Time to complete simulation is \( \text{poly}(|[M]|) \cdot f(n) \log f(n) \). We want that to be less than \( g(n) \). We have \( f(n) \log f(n) = o(g(n)) \), but the \( \text{poly}([M]) \) factor is \( \text{poly}(n) \) and will give a coarser hierarchy theorem than we want. To deal with this, we only use a small portion of \( x \) as the description of the TM, so its size is arbitrarily small relative to input length. We want to be able to fix \( [M] \) while taking \( n \to \infty \).

We do this by padding our encodings of TMs. Specifically, for a string \( x \), we define \( M_x \) to be the TM \( M \) s.t. \( x = [M]1000 \cdots 0 \). This way we can take \( n = |x| \) to be arbitrarily large while keeping the TM \( M \) fixed. Now for sufficiently large \( n \), we will have \( \text{poly}([M]) \cdot f(n) \log f(n) < g(n) \).

By the time hierarchy theorem, all the following time classes are distinct:

\[
\text{DTIME}(n) \subset \text{DTIME}(n \log^2) \subset \text{DTIME}(n^2) \subset P \subset \tilde{P} \subset \text{SUBEXP} \subset \text{EXP} \subset \text{EEXP},
\]

where

\[
\tilde{P} = \bigcup_c \text{DTIME}(2^{\log^c n}) \text{ ("quasipolynomial time")}
\]

\[
\text{SUBEXP} = \bigcap_\varepsilon \text{DTIME}(2^{n^\varepsilon}) \text{ ("subexponential time")}
\]

\[
\text{EEXP} = \bigcup_\varepsilon \text{DTIME}(2^{2^{n^\varepsilon}}) \text{ ("double-exponential time")}
\]

We remark that we restrict to time bounds \( f(n) \geq n \) because this much time is needed to even read the input. However, if one allows randomization and certain notions of approximation, then sublinear-time algorithms become quite nontrivial, and indeed there is now a large literature on this subject, motivated by massive data sets (which we won’t have time to address in this course, except in the context of proof-verification — PCPs).

Another remark is that if we remove the time-constructibility condition and allow pathological bounds \( f(n) \), the hierarchy theorem becomes false and we can even have \( \text{DTIME}(f(n)) = \text{DTIME}(2^{2^{f(n)}}) \).

The proof of the Time Hierarchy Theorem shows that there is this contrived diagonal problem in \( \text{DTIME}(g(n)) \setminus \text{DTIME}(f(n)) \):

\[
L(D) = \{ x : U([M_x], x) \text{ accepts in } \leq g(|x|) \text{ steps} \}
\]

What about more natural problems?

The Bounded Halting Problem

\[
H_f = \{ ([M], x) : M \text{ accepts } x \text{ within } f(|x|) \text{ steps} \}
\]
can be shown to not be in $\text{DTIME}(f(n))$. (If it were easy, then $L(D)$ would be too.) However, due to the $\text{poly}(|\lfloor M \rfloor|)$ factor in the running time of the universal TM, we can only put $H_f$ in $\text{DTIME}(\text{poly}(n) \cdot f(n) \log f(n))$ (rather than $\text{DTIME}(g(n))$ for any $g(n) = \omega(f(n) \log f(n))$). Eventually, we will use reductions and completeness to get even more natural problems of high time complexity (just as done in computability theory).

For space, we have an even finer hierarchy:

**Theorem 2 (Space Hierarchy)** If $g$ is space-constructible ($1^n \to 1^g(n)$ can be computed in space $O(g(n)))$, $f(n) = o(g(n))$, then $\text{SPACE}(f(n)) \subsetneq \text{SPACE}(g(n))$.

**Proof:** Same as Time Hierarchy Theorem, but use the fact that Space $U(\lfloor M \rfloor, x) \leq C_M \cdot \text{Space}_M(x)$. □

As this illustrates, the proofs of the above hierarchy theorems are quite general. Informally speaking, to show a separation of complexity classes $C_1 \not\subseteq C_2$ with this approach, all we need is that algorithms from $C_1$ can (1) enumerate, (2) simulate, and (3) negate algorithms from $C_2$. How fine a hierarchy theorem you get comes from how tight the simulation is (in terms of resources). We have a log factor in the time hierarchy theorem because the universal TM pays a log factor; for other models of computation, a constant factor may suffice.

Later in the semester, we will see that such generic diagonalization arguments are insufficient to resolve the major open problems in complexity theory (like $\text{P} \vs \text{NP}$).

**Theorem 3 (Nondeterministic Time Hierarchy)** If $g(n)$ time constructible and $f(n+1) \log f(n+1) = o(g(n))$, then $\text{NTIME}(f(n)) \subsetneq \text{NTIME}(g(n))$.

Why can’t we just do the same thing as we did with the deterministic time hierarchy theorem? Nondeterministic computation does not seem to be closed under negation. Recall that the accepting condition for an NTM is that there exists at least one accepting computation. Negating this yields a “for all” condition.

**Solution ideas:**

1. We can negate if we have exponentially more time. But then you won’t get a fine hierarchy at all.

2. Only try to disagree at least once in an exponentially large interval. (“Lazy diagonalization”)

**Proof:** For the $k$’th non-deterministic TM $M_k$, associate an interval $I_k = (\ell_k, u_k]$ of positive integers. These should be disjoint but can be contiguous (we can take $\ell_{k+1} = u_k$). The upper bound $u_k$ should be exponentially larger than the lower bound $\ell_k$, e.g. $u_k = 2^{\ell_k^2}$.

Now we define our diagonalizing NTM $D$ (on unary inputs $1^n$) as follows.

$D(1^n)$:

1. Find $k$ such that $n \in I_k$

2. (a) If $n < u_k$: Simulate nondeterministic universal TM on $M_k$ and $1^{n+1}$ for up to $g(n)$ steps. (Note that here we are not negating, but we are doing the simulation on the input that is the successor of the input to $D$.)
(b) If \( n = u_k \): Deterministically try to decide if \( M_k \) accepts \( 1^{\ell_k+1} \) by trying all computation paths, for a total of at most \( g(n) \) steps, and and do the opposite. (Here we are negating, but we are doing so on an input much shorter than the input to \( D \).)

By construction, \( D \) runs in nondeterministic time \( g(n) \). Suppose for contradiction that there exists an NTM \( M \) that runs in non-det time \( f(n) \) s.t. it accepts the exact same language as \( D \) \( (L(M) = L(D)) \). We may assume that there are infinitely many \( k \) s.t. \( M_k = M \) by using the same kind of encoding trick as in the deterministic hierarchy theorem. In particular, \( L(M) = L(D) \) on some such interval \( I_k \) where \( M_k = M \) and \( k \) is arbitrarily large. We’ll just focus on this interval. For sufficiently large \( k \) and sufficiently large intervals (like \( u_k = 2^{\ell_k} \)), it can be verified that all of the simulations in step 2 will complete within \( g(n) \) steps.

Now, we know that \( M(1^{\ell_k+1}) = D(1^{\ell_k+1}), M(1^{\ell_k+2}) = D(1^{\ell_k+2}),..., M(1^{u_k+1}) = D(1^{u_k+1}) \). But, by the definition of \( D \), it is simulating \( M_k \) on the next input, so \( D(1^{\ell_k+1}) = M(1^{\ell_k+2}), D(1^{\ell_k+2}) = M(1^{\ell_k+3}),..., D(1^{u_k-1}) = M(1^{u_k}) \). So \( M \) and \( D \) must agree on all inputs in the interval \( I_k \). But when \( n = u_k \), we negate to get \( D(1^{u_k}) \neq M(1^{\ell_k+1}) \). Contradiction.