CS221: Computational Complexity Prof. Salil Vadhan

Lecture 4: More Hierarchy Theorems

9/27 *Scribe: Sasha Schwartz*

Contents

Relevant reading: Papadimitriou, Sec 7.2. (Does not cover much of the lecture.)

1 Recap

Correction from last lecture: a proper complexity function requires that $f(n+1) \geq f(n)$ for all n.

Theorem 1 (Time Hierarchy Theorem) *If $f(n) \geq n$ is a proper complexity function, then* $\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(n^2 f(n)^2)$.

Proof: We construct a Turing Machine $M \in \mathbf{TIME}(n^2 f(n)^2)$ with $M \notin \mathbf{TIME}(f(n))$. For a string x, first compute $f(|x|)$, and simulate $M_x(x)$ for $f(|x|)$ steps, where M_x is the Turing Machine associated to the string x. If M_x halts on x in $f(|x|)$ steps, then output the opposite of $M_x(x)$. Otherwise output 0. The Turing Machine M which performs the above sequence of steps runs in time $O(|M_x|^2 f(|x|)^2) = O(|x|^2 f(|x|)^2)$, but by construction no Turing Machine running in time $f(n)$ agrees with M on every input. Thus M is as desired, and the proof is complete. ■

What makes this diagonalization work?

- Algorithms from the larger complexity class ($\mathbf{TIME}(n^2 f(n)^2)$) can simulate algorithms from the smaller class ($\mathbf{TIME}(f(n))$).
- Algorithms from the larger class can (effectively) enumerate algorithms from a small class (in this case by clocking the simulation for $f(n)$ steps).
- Closure under complement (i.e. we can "output the opposite").

2 Explicit Problems

Can we identify a *specific* problem (i.e. language) in $\mathbf{TIME}(n^2 f(n)^2) \setminus \mathbf{TIME}(f(n))$? One such language comes from the proof of Theorem 1. With the M defined in the proof we have

$$L(M) = L_f = \{x : M_x(x) = 0 \text{ within } f(|x|) \text{ steps}\}.$$

We can also deduce that the following BOUNDED HALTING problem is hard:

$$H_f = \{\langle M, x\rangle : M(x) \text{ halts within } f(|x|) \text{ steps}\}.$$

We show H_f is hard by relating it to L_f. Let M'_x be a Turing Machine that is identical to M_x except that it goes into an infinite loop when M_x is about to halt and output 1. Then $x \in L_f$ if and only if $\langle M'_x, x\rangle \in H_f$. Notice that if x is of length n, then $\langle M'_x, x\rangle$ is of length roughly $2n$. Therefore $H_f \in \mathbf{TIME}(f(n/2)) \Rightarrow L_f \in \mathbf{TIME}(f(n))$, and we conclude $H_f \notin \mathbf{TIME}(f(n/2))$.

While the languages L_f and H_f are reasonably interesting from a computer science perspective, we'd really like to identify *natural* intractable problems, e.g. ones that don't directly refer to computation.

3 A Refinement

The function $n^2 f(n)^2$ in Theorem 1, which comes from the running time of a universal Turing Machine, provides a fairly coarse separation. The $f(n)^2$ part can be improved but it is not so easy (uses a more efficient universal TM which simulates t steps in time $O(t \log t)$), but the n^2 can be easily replaced with any slowly growing function as follows.

Theorem 2 *If $f(n), g(n) \geq n$ are proper complexity functions and $g(n) = \omega(f(n)^2)$ (meaning that $\lim_{n\to\infty} g(n)/f(n)^2 = \infty$), then $\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(g(n))$.*

Proof: We describe the steps of a Turing Machine M on an input string x. First compute $f(|x|)$ and $g(|x|)$. Simulate $M_{x'}(x)$ for $f(|x|)$ steps where x' consists of the first $\lfloor \sqrt{g(|x|)/f(|x|)^2} \rfloor$ bits of x. The remainder of the construction and proof mirrors that of the proof of Theorem 1, except that the running time of M is $O(|x'|^2 f(|x|)^2) = O(g(|x|))$, as desired. ■

4 Almost Everywhere Hard

An important issue in complexity theory is that for a given "hard problem" L, how "rare" are the "hard instances"? This question can be interpreted in several ways. Consider the language L we constructed in $\mathbf{TIME}(n^2 f(n)^2) \setminus \mathbf{TIME}(f(n))$. We know there is no algorithm that *always* decides L correctly and *always* runs in time $f(n)$. But we can ask for more, for example:

1. Every algorithm that runs in time $f(n)$ outputs the wrong answer on infinitely many inputs.
2. Every algorithm that runs in time $f(n)$ outputs the wrong answer on all but finitely many inputs.
3. Every algorithm that decides L correctly runs in time $> f(n)$ on infinitely many inputs.

4. Every algorithm that decides L correctly runs in time $> f(n)$ on all but finitely many inputs.

Property 1 already follow from the fact that $L \notin \mathbf{TIME}(f(n))$:

Corollary 3 *There is a language $L \in \mathbf{TIME}(n^2 f(n)^2) \setminus \mathbf{TIME}(f(n))$ such that for any Turing Machine M' running in time $f(n)$ there exist infinitely many input strings x with $M'(x) \neq \chi_L(x)$ (where χ_L denotes the characteristic function).*

Proof: Let L be the language given by Theorem 1. If M' errs on only finitely many inputs x then these errors can be corrected with a lookup table and no slowdown. That is we may construct $M'' \in \mathbf{TIME}(f(n) + O(n)) = \mathbf{TIME}(f(n))$ with $L(M'') = L$, which is a contradiction. ■

Property 3 above follows by a similar argument. Property 2, however, is too strong: because either the all-0 or the all-1 Turing Machine will be correct infinitely many times. However, we can achieve Property 4 with a more complicated diagonalization:

Theorem 4 (Almost-Everywhere Hierarchy Theorem) *Let $f(n) \geq n$ be any proper complexity function. There exists $L \in \mathbf{TIME}(2^{O(n)} f(n)^2)$ such that for any Turing Machine M' with $L(M') = L$ we have that M' takes time at least $f(n)$ on all but finitely many inputs.*

Proof: Again we describe the execution of a Turing Machine M on an input string x. First compute $f(|x|)$, and then find the first $i \leq |x|$ such that

1. M_i halts within $f(|x|)$ steps on x, and
2. for all $y < x$ lexicographically either $M_i(y) = M(y)$ or $M_i(y)$ does not halt within $f(|y|)$ steps.

If no such i exists then output 0, and otherwise output the opposite of $M_i(x)$. (In the latter case M_i is said to have been *cancelled.*)

We see immediately than any M_i can be cancelled at most once, for once cancelled condition 2 with never again be satisfied for M_i.

Now suppose, for contradiction, that there exists some M_i with $L(M_i) = L(M)$ such that there exists an infinite set of inputs $x_1 < x_2 < \cdots$ with $M_i(x_k)$ running in time less than $f(|x_k|)$ for all $k \geq 1$. Then there exists some ℓ with $x_\ell > i$ such that for all $j < i$ either M_j has been cancelled for some string less than x_ℓ or M_j will never be cancelled. However, in the computation of $M(x_\ell)$, we see that condition 2 fails for all $j < i$, and conditions 1 and 2 both hold for M_i. Thus M_i will be cancelled as well, and $L(M_i) \neq L(M)$, providing the desired contradiction. ■