IAS/PCMI Summer Session 2000
Clay Mathematics Undergraduate Program
Basic Course on Computational Complexity

Lecture 9: Hierarchy Theorems

David Mix Barrington and Alexis Maciel
July 27, 2000

Most of this lecture will be devoted to hierarchy theorems that will allow us to separate some of our complexity classes. At the end of the lecture, we present an example of a probabilistic argument. This type of argument will be used in the next lecture to separate two more classes.

1. The Space Hierarchy Theorem

If a machine is allowed to use a significantly larger amount of resources, it would seem like it should be able to decide more languages. In this lecture, we show that this intuition is right. For example, we show that in space $O(s(n))$ we can decide a language that cannot be decided in space $o(s(n))$, provided that s is space constructible. (Recall that a function s is space constructible if given 1^n, the binary representation of $s(n)$ can be computed in space $O(s(n))$.) One consequence is that $\mathrm{L} \subset \mathrm{DSPACE}(n)$, which implies that one of the following inclusions must be strict: $\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{AC}^1 \subseteq \mathrm{NC} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$. Note, however, that we currently do not know which inclusion that might be. (It is generally believed that all these inclusions are strict.)

Theorem 1 *If s is space constructible, then there is a language that is decided in space $O(s(n))$ but not in space $o(s(n))$.*

Proof What we are going to do is design a machine D that runs in space $O(s(n))$ and we are going to make sure that if M is any machine that runs in space $o(s(n))$, then D and M disagree on some input. The language decided by D will then satisfy the conditions in the statement of the theorem.

To design D we will use the important technique of *diagonalization*. First, fix some encoding of machines over some alphabet and let $\langle M\rangle$ denote the encoding of

M. Given the encoding of some machine M, D will simulate M on the input string $\langle M \rangle$ and accept if and only if M rejects. Then if M is any machine running in space $o(s(n))$, M and D will disagree on $\langle M \rangle$.

More precisely, D first verifies that its input is the encoding of some machine M. If not, it rejects. Second, D computes the space bound $s(n)$. Third, D simulates M on $\langle M \rangle$. During this simulation, D will store in its own memory the contents of M's memory. If at any moment the amount of space needed to store M's memory contents exceeds $s(n)$, the simulation is halted and D rejects. Otherwise, if the simulation completes with M having either accepted or rejected, then D will do the opposite.

Now if M is any machine running in space $o(s(n))$, we would like to claim that D disagrees with M on $\langle M \rangle$. To do this, we run D on $\langle M \rangle$, which causes D to simulate M on $\langle M \rangle$ and output the opposite. If M uses space $f(n)$ and if storing each memory symbol of M requires b symbols in D's memory, then we need to ensure that $bf(n) \leq s(n)$ so the simulation can be guaranteed to find whether M accepts or rejects. However, we only know that $f(n) = o(s(n))$, which implies that $bf(n) \leq s(n)$ but only for sufficiently large n, not for all n. For small inputs, simulating M could use too much space.

To fix this, we modify D so that whenever it gets an input of the form $\langle M \rangle 1^k$ for some k, D simulates M on $\langle M \rangle 1^k$. When k is sufficiently large, we will have $bf(n) \leq s(n)$ and D will have enough space to simulate M and output the opposite. So we have established that D disagrees on some input with any machine M running in space $o(s(n))$.

It is clear that D runs in space $O(s(n))$. But we also want D to decide a language, so we need to make sure that D always halts. If M is a machine deciding a language in space $o(s(n))$, then we know that M always halts, so the simulation will always terminate. But D's input can be any machine whatsoever, so we need to take care of the possibility that M uses a small amount of space but does not halt. Now any machine running in space $o(s(n))$ also runs in time $2^{o(s(n))}$, which is less than $2^{s(n)}$ when n is large enough. So we will add to the simulation a counter and stop the simulation and reject if we find that M's computation has taken more than $2^{s(n)}$ steps. This will ensure that D always halts while still allowing the complete simulation of any machine that runs in space $o(s(n))$. $\square$

An immediate corollary of this result is that if $s_1 = o(s_2)$ and s_2 is space constructible, then $\mathrm{DSPACE}(s_1) \subset \mathrm{DSPACE}(s_2)$. In particular, $\mathrm{L} \subset \mathrm{DSPACE}(n)$, which implies that $\mathrm{L} \subset \mathrm{PSPACE}$. But we can say more, by using Savitch's Theorem.

Corollary 2 NL $\subset$ PSPACE.

Proof NL $\subseteq$ DSPACE($\log^2 n$) $\subset$ DSPACE(n). $\square$

2. The Time Hierarchy Theorem

We just proved that increasing the amount of space from $o(s(n))$ to $O(s(n))$ allows machines to decide more languages. The same is true for time, though we defer the proof to the exercises.

Theorem 3 *If t is time constructible, then there is a language that is decided in time $O(t(n))$ but not in time $o(t(n))$.*

Corollary 4 P $\subset$ EXP.

Proof For every k, DTIME(n^k) $\subseteq$ DTIME($n^{\log n}$), which implies that P $\subseteq$ DTIME($n^{\log n}$). The result follows since DTIME($n^{\log n}$) $\subset$ DTIME(2^n). $\square$

A consequence of this is that one of the following inclusions must be strict: P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXP. Once again, we do not know which one. For another consequence, the exercises ask you to show that there are problems that cannot be solved by any machine whatsoever. From the Time Hierarchy Theorem, we now know that there are languages that can be solved by machines but not by efficient machines (i.e., machines that run in polynomial time).

Note that the proof of the above theorem relies on the fact that our machines have a constant number of memory heads, not just one. For machines that have only one memory head, we could have proved a separation between time $O(t(n))$ and time $o(t(n)/\log t(n))$. It is not known if the separation holds with $o(t(n))$ for machines with a single memory head.

3. A Probabilistic Argument

We end this lecture with a result about graphs that will serve as our introduction to probabilistic arguments. An argument of this type will be used in the next lecture.

In an undirected graph, an *anticlique* or *independent set* is a set of nodes in which every two nodes are *not* connected by an edge. For example, a pentagon has anticliques of size 2 but not of size 3. It may seem reasonable to think that as the number of nodes in a graph becomes very large, the chances increase that it will contain either a large clique or a large anticlique. In fact, Ramsey's Theorem states that *every* graph of size at least 2^{2k} contains either a clique of size k or an anticlique of size k.

Let $R(k)$ be the minimum number such that every graph with $R(k)$ nodes is guaranteed to contain either a clique or an anticlique of size k. Ramsey's Theorem gives an upper bound on this number: $R(k) \leq 2^{2k}$. What about a lower bound? $R(k) \geq k$ is trivial and, for $k \geq 3$, $R(k) \geq 2k$ is easy: consider a polygon with $2k-1$ sides. However, these numbers are very far from the upper bound of 2^{2k}. Can we get closer? The following theorem, due to Erdös, shows that we can. Its proof is an example of a *probabilistic argument*: we show that an object of a certain type with a certain property exists by showing that the probability that a random object of the same type does not have the property is less than 1.

Theorem 5 $R(k) \geq 2^{k/2-1}$.

Proof Let n and k be arbitrary. Choose a graph with n nodes at random by deciding whether each of the possible $\binom{n}{2}$ edges is in or out. For any set A of k nodes, the probability that A is a clique is $2^{-\binom{k}{2}}$. The probability that A is an anticlique is the same. Therefore, the probability that there is a set of k nodes in the graph that is either a clique or an anticlique is no more than $2\binom{n}{k}2^{-\binom{k}{2}}$.

If $n \geq R(k)$, then every graph with n nodes has a clique or anticlique of size k, so this probability is 1 and this implies that $2\binom{n}{k}2^{-\binom{k}{2}} \geq 1$. Therefore, if $2\binom{n}{k}2^{-\binom{k}{2}} < 1$, then some graph with n nodes must have no cliques or anticliques of size k, which implies that $n < R(k)$. A straightforward calculation shows that n satisfies the condition if $n < 2^{k/2-1}$. The lower bound follows. $\square$

4. Exercises

1. Show that for every rational number r, the function n^r is space constructible.

2. Show that the following problem cannot be decided by any machine whatsoever: given a machine M and an input x, determine whether M halts on x. This problem is usually referred to as the *halting problem*. It is the typical example of an *undecidable* problem.

3. Show that the following problem is undecidable: given a machine M, does M halt on the empty input? (Hint: Given an input $\langle M, x \rangle$ to the halting problem, consider a machine M_x that ignores its input and simply simulates M on x.)

4. Show that NP $\neq$ DTIME(n).

5. Show that NP $\neq$ DSPACE(n).

6. Prove the Time Hierarchy Theorem.

7. Prove Ramsey's Theorem. That is, show that $R(k) \leq 2^{2k}$. (Hint: Let $R(i, j)$ be the minimum number such that every graph with $R(i, j)$ nodes is guaranteed to contain either a clique of size i or an anticlique of size j. Show that $R(i, j) \leq R(i, j-1) + R(i-1, j)$.)