1 Time and Space Hierarchies

The most fundamental results in complexity theory are the hierarchy theorems, which state that more problems can be solved given more resources. We will now prove hierarchy theorems for the two resources we have defined so far: time and space.

We first give a high-level overview of the proof ideas. The most important idea used in the proof is *diagonalization*, which was used by Cantor to show that the cardinality of the reals is greater than the cardinality of the natural numbers. Diagonalization is also widely used in computability theory, eg., for showing that the Halting Problem is uncomputable.

We will explain the proof idea for the time hierarchy; the proof of the space hierarchy is very similar. Suppose we want to show that there are languages decidable in deterministic time T but not in time t, where t and T are time bounds where T is time-constructible and "sufficiently larger" than t. To establish this, we need to show both a complexity upper bound and a complexity lower bound - we need to define a language L for which we can establish that $L \in \mathsf{DTIME}(T)$ but $L \notin DTIME(t)$. In general, it's hard to show complexity lower bounds for a problem because we need to prove that no *conceivable* Turing machine operating within the stated resource bounds can solve the problem.

Diagonalization is one of the few techniques which is known to work in establishing lower bounds. Intuitively, diagonalization is a way of proving the existence of an object that is not in a given countably infinite set of objects. In Cantor's proof, the objects are real numbers - the proof shows that for any countabe list of real numbers, one can find a real number not in the list. In our proof, the objects will be languages in $\mathsf{DTIME}(t)$, described by their characteristic function. Imagine an infinite table A where $A(i, j)$ is 1 iff the j'th bit of the binary description of the i'th object is 1. Now, "flip the diagonal", i.e., consider the object X such that the i'th bit in the description of X is 1 iff $A(i, i)$ is 0. X cannot be in the list, because it differs in the i'th position of its representation from the i'th element in the list, for any i.

As mentioned above, we apply diagonalization to some effective listing of the languages in $\mathsf{DTIME}(t)$ - this listing can be done using the facts about encodings of Turing machines in the notes on Lecture 2. This gives us a language L not in $\mathsf{DTIME}(t)$, i.e., the lower bound. We still need to show that L is in $\mathsf{DTIME}(T)$ for some T sufficiently larger than t. We will show this using the fact that there is a time-efficient universal Turing machine, which enables us to "flip the diagonal" efficiently. The formal proof is below, and incorporates some technical details which we've ignored in our high-level description, such as the dependence of T on t and the need for time-constructibility.

Theorem 1 *Let $t : \mathbb{N} \to \mathbb{N}$ and $T : \mathbb{N} \to \mathbb{N}$ be functions such that T is time-constructible and $t \log(t) = o(T)$. Then* $\mathsf{DTIME}(t) \subset \mathsf{DTIME}(T)$.

Proof. We define a machine M running in time T such that $L(M) \notin \mathsf{DTIME}(t)$.Given an input x, M operates as follows. It first computes $T(|x|)$ and stores this number in unary on one of its work tapes. This can be done in time $O(T(|x|))$ by time-constructibility of T. It then simulates the Turing machine M_x encoded by x on input x, by running the time-efficient universal Turing machine U whose existence is stated in Theorem 1 of the previous Lecture Notes on input $< x, x >$. This simulation makes sense by Property 1 of the encodings we use, mentioned in the previous Lecture Notes. For each time step of U it simulates, it deletes a 1 from the unary representation of $T(|x|)$ it has on a worktape. This worktape is used as a "time clock" - if all the 1's are deleted, then this means that there is zero time left for the simulation, and M halts and rejects. If, on the contrary, M is able to complete its simulation of U before the counter reaches zero, there are two cases. If U rejects, then M halts and accepts. If U accepts, M halts and rejects. (Intuitively, this behaviour corresponds to "flipping the diagonal").

Clearly, M runs in time T, so $L(M) \in \mathsf{DTIME}(T)$. Now suppose that $L(M) \in \mathsf{DTIME}(t)$. We will derive a contradiction. Let M' be a Turing machine halting within cn time steps on any input of length n, where c is a constant, and deciding $L(M)$. By Property 2 of the encodings of Turing machines we use, mentioned in the previous Lecture Notes, for any integer m, there is an encoding y of M' such that $|y| > m$. Now since $t \log(t) = o(T)$ there must be an integer m such that $cd_{M'}t(m)\log(t(m)) < T(m)$, where $d_{M'}$ is the constant in Theorem 1 of the previous Lecture Notes. Choose an encoding y of M' such that $|y| > m$. Now consider the behaviour of M and M' on y. M on input y has enough time to complete the simulation of U before its counter hits zero, by the assumption on length of y. Therefore M accepts y iff U rejects on $< y, y >$, which happens iff M' rejects on y. Thus $L(M)! = L(M')$, which contradicts the assumption on M'. □

The proof of the space hierarchy is exactly analogous, except that the parameters of the result are slightly stronger, since we use Theorem 2 from the previous Lecture Notes rather than Theorem 1, and the counter is used to record the space used by the simulation rather than the time.

Theorem 2 *Let $s : \mathbb{N} \to \mathbb{N}$ and $S : \mathbb{N} \to \mathbb{N}$ be functions such that S is space-constructible. Then* $\mathsf{DSPACE}(s) \subset \mathsf{DSPACE}(S)$.

2 Relationships between Time and Space, and Closure Properties of Complexity Classes

There are some straightforward relationships which hold between time and space complexity classes, which we describe next.

Proposition 3 *Let T be any time bound.* $\mathsf{DTIME}(T) \subseteq \mathsf{DSPACE}(T)$.

The reason is that any computation halting after at most t steps cannot use space more than t.

The best known simulation of space by time is much weaker.

Theorem 4 *Let S be any space bound such that S is space-constructible and $S = \Omega(\log(n))$. Then* $\mathsf{DSPACE}(S) \subseteq \bigcup_{c>1} \mathsf{DTIME}(c^S)$.

Proof. Let L be any language in $\mathsf{DSPACE}(S)$ and M be a machine with read-only input tape deciding L in space $O(S)$. The key observation is that there are at most $(n+1)2^{O(S)}$ possible configurations of M on any input of length n, since a configuration of M is completely described by the position of the input tape head, the positions of the work tape heads, the contents of the work tapes and the state. There are at most $(n+1)$ possible positions of the input tape head, at most $poly(S)$ possible positions of work tape heads, at most $2^{O(S)}$ possible contents of work tapes, and at most constantly many possible states of the machine.

Now if any configuration repeats in the computation of M on x, the computation cannot halt, since M is a deterministic machine. So any accepting computation must terminate within $(n+1)2^{O(S)}$ steps. This is at most $2^{O(S)}$ if $S = \Omega(\log(n))$. We define a deterministic Turing machine M' accepting L and halting within time c^S for some constant $c > 1$. M' first uses the space-constructibility of S to obtain the unary representation of d^S in unary within time d^S, where d is a constant such that the number of possible configurations of M on input x is bounded above by d^S. There is a subtlety here, in that space-constructibility doesn't immediately imply time-constructibility, but here we can again use the fact that configurations of the space transducer cannot repeat to obtain the time bound for the first part of the computation of M'. M' then uses the number it has computed as a "clock" while it simulates M on x. If the clock runs out, M' rejects. If the simulation of M completes before this happens, M' accepts if and only if M accepts.

Clearly M' runs in time $2^{O(S)}$, and decides L correctly using our upper bound on the termination time of M on x. $\square$

Also of interest are closure properties of complexity classes. We would like complexity classes to be *robust*, i.e., closed under simple operations like union and intersection. We next observe that deterministic time and space complexity classes satisfy this condition.

Proposition 5 *For any time bound t, $\mathsf{DTIME}(t)$ is closed under union and intersection. Namely, given $L_1 \in \mathsf{DTIME}(t)$ and $L_2 \in \mathsf{DTIME}(t)$, $L_1 \cup L_2 \in \mathsf{DTIME}(t)$ and $L_1 \cap L_2 \in \mathsf{DTIME}(t)$. The same closure results hold for space-bounded classes.*

The proof of the above proposition is simple. To observe closure under union, let M_1 be a machine deciding L_1 in time $O(t)$ and M_2 a machine deciding L_2 in time $O(t)$. Then a machine M which accepts on x iff either M_1 or M_2 accepts on x halts in time $O(t)$ and decides $L_1 \cup L_2$. Similarly, a machine M' which accepts on x iff both M_1 and M_2 accept on x halts in time $O(t)$ and decides $L_1 \cap L_2$.

Closure under complementation also holds, with just a slightly more involved argument.

Proposition 6 *If $L \in \mathsf{DTIME}(t)$, then $\bar{L} \in \mathsf{DTIME}(t)$. Similarly for $\mathsf{DSPACE}(s)$.*

The proof is just by interchanging accepting and rejecting states. Note that a time-bounded machine halts on all inputs. So the only ways it could reject are by falling off the end of a tape, or by reaching a configuration where there is no next transition. We could modify the machine so that it enters a rejecting state q_r in either of these cases. Now the Turing machine where the accepting state q_f and the rejecting state q_r of this modified machine are swapped halts in time t and decides the complement of L.