CSE596 Notes on Complexity Class Relations Fall 2018

These are lecture notes for Monday 11/5 and Wednesday 11/7.

The central mystery of computational complexity is: why do we know *tight* separations between classes defined for the *same* complexity measure but have a yawning exponential gap in our knowledge of inclusions of classes *between* the major complexity measures? The best inclusions we know are:

Theorem 1. *For any "reasonable" time measure $t(n) \geq n+1$ and space measure $s(n) \geq \log_2 n$,*

$$\mathsf{DSPACE}[s(n)] \subseteq \mathsf{NSPACE}[s(n)] \subseteq \mathsf{DTIME}[2^{O(s(n))}]$$
$$\mathsf{DTIME}[t(n)] \subseteq \mathsf{NTIME}[t(n)] \subseteq \mathsf{DSPACE}[O(t(n))] \subseteq \cdots$$

Proof. The first and third containments are immediate by definition. For the second, let N be an NTM with some number k of tapes and work alphabet Γ that runs in space $s(n)$, and consider any input x to N, putting $n = |x|$ as usual. The notion of "reasonable" allows us to lay out in advance $s(n)$ tape cells that N is allowed to change. Thus any configuration I has the form $I = \langle q, w, \vec{h} \rangle$ where q is the current state, $w \in \Gamma^{s(n)}$ represents the current content of the cells N can change, and $\vec{h}$ gives the head positions on all tapes, including the location of the input head reading x. Note that I does not need to give the parts that don't change—if all cells occupied by x are kept constant, w doesn't need to include any of them. So the total number of different possible IDs we need to consider on input x is at most

$$|Q| \cdot |\Gamma|^{s(n)} \cdot (n+2)(s(n)+2k-2)^{k-1}.$$

Since $s(n) \geq \log_2(n)$, $|\Gamma|^{s(n)}$ is at least $2^{\log_2(n)} = n$, so the third factor does not dominate the second factor and the whole size is bounded by $2^{O(s(n))}$. (The $+2$ and $2k-2$ allow the heads to occupy blanks to the left or right of x and the cells they can change, however they are laid out on the tapes; they don't really matter to the $2^{O(s(n))}$ size estimate.)

Now we define a directed graph G_x with the IDs $I, J, \ldots$ as its nodes and the relation $I \vdash_N J$ as its edge relation. Then N accepts x if and only if breadth-first search from the starting ID $I_0(x)$ finds an accepting ID. Since BFS runs in time polynomial in the size of the graph, and polynomial-in-$2^{O(s(n))}$ still gives $2^{O(s(n))}$, we obtain a deterministic algortihm that decides whether $x \in L(N)$ in time $2^{O(s(n))}$. This proves the second containment.

The fourth containment is (IMHO) best described as a *depth-first* search. Given a k-tape NTM N that runs in time $t(n)$, we may suppose N has binary nondeterminism, so that on any input x of length n there are at most $t(n)$ bits of nondeterminism that N can use. We can organize all the possible guesses y as branches of a binary tree T of depth $t(n)$ and allocate $t(n)$ cells to hold the current y we are trying. Since $N(x)$ cannot possibly use more than $kt(n)$ tape cells, we need only $t(n) + kt(n)$ space total to do a full transversal of T. We accept x if and only if an accepting branch is found. This simulation takes roughly $2^{t(n)}$ time but it all operates within $O(t(n))$ space, so $L(N) \in \mathsf{DSPACE}[O(t(n))]$. $\square$

For example with $s(n) = O(\log n)$ we get $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$. This brings us back full-circle to the deterministic space measure, and we can ratchet up to the next level: $\mathsf{PSPACE} \subseteq \mathsf{NPSPACE}$ (actually, these two are equal by Savitch's Theorem, for later) $\subseteq \mathsf{EXP} =_{def} \mathsf{DTIME}[2^{n^{O(1)}}] \subseteq \mathsf{NEXP} \subseteq \mathsf{EXPSPACE}$. The only multi-link chain of differences we know from these classes is $\mathsf{NL} \subset \mathsf{PSPACE} \subset \mathsf{EXPSPACE}$ (or with L in place of NL). The links in that chain are not only different, they are *vastly* different, because deterministic space has a tight hierarchy:

Theorem 2. *If s_1, s_2 are "reasonable" space functions and $s_1(n) = o(s_2(n))$, then $\mathsf{DSPACE}[s_1(n)]$ is properly contained in $\mathsf{DSPACE}[s_2(n)]$.*

Thus for example, in the case starting with $s_1(n) = \log_2 n$ and $s_2(n) = \log^2(n) =_{def} (\log n)^2$, we get:

$$\mathsf{L} \subset \mathsf{DSPACE}[(\log n)^2] \subset \mathsf{DSPACE}[(\log n)^3] \subset \cdots \subset \mathsf{DSPACE}[n] \subset \mathsf{DSPACE}[n \log n] \subset \mathsf{DSPACE}[n^2] \subset \cdots$$

The separations in the above chain tell us that at least one of the containments in Theorem 1 must be proper (for $s = \log_2 n$ and generally). In fact, "we" *believe* all four of them are proper, but we haven't *proved* any of them.[1] The hierarchy for deterministic time is almost as tight:

Theorem 3. *If t_1, t_2 are "reasonable" time functions and $t_1(n)\log(t_1(n)) = o(t_2(n))$, then $\mathsf{DTIME}[t_1(n)]$ is properly contained in $\mathsf{DTIME}[t_2(n)]$.*

In particular, this means that even *within* P, deterministic time is quite stratified:

$$\mathsf{DLIN} =_{def} \mathsf{DTIME}[O(n)] \subset \mathsf{DTIME}[N^{1.000001}] \subset \mathsf{DTIME}[n\sqrt{n}] \subset \mathsf{DTIME}[n^2] \subset \mathsf{DTIME}[n^3] \subset \cdots \subset \mathsf{P}.$$

So why can't we tell that SAT does not belong to $\mathsf{DTIME}[N^{1.000001}]$, let alone that it does not belong to P? A good question! The best road for understanding the issue is to see how the common proof of both theorems 2 and 3 works. The notes by Debray prove only a weaker version with $\sqrt{t_1(n)}$ in place of the $\log(t_1(n))$ factor, and the reason the professor at Stanford did this is that existing presentations of the stronger result are so 'yucky' that Allender and Loui and I didn't prove them in our notes either. However, I have found a way to roll several technical propositions into a single statement that gives the springboard for the final diagonalization step of the proof:

Theorem 4. *We can build a single 3-tape DTM M_3 with tape alphabet $\Gamma_3 = \{0, 1, \_\}$ such that for any DTM M with input alphabet $\Sigma = \{0,1\}$ but any number k of tapes and work alphabet Γ_M of any size, there is a constant $C > 0$ such that for any $x \in \Sigma^*$ and $t > 0$, the first t steps of the computation of M on input x are simulated by the first $C + Ct\log(t)$ steps of M_3 on input $\langle M, x\rangle$, using at most C times as much space.*

The constant C depends on the given M. It does not depend on w. It mostly comes from the string length of the code $\langle M\rangle$ of M and reflects not only the number of states and instructions but also the overhead for encoding Γ_M by the binary-plus-blank alphabet Γ_3. It also gets a contribution from the constant factor in the $O(\log t)$ time overhead for reducing k tapes to 2 tapes. Note that going from time t to time $O(t \log t)$ is markedly better than the $O(t^2)$ time shown in class for getting down to a single tape. The machine M_3 on input $\langle M, x\rangle$ first copies the M part to its third tape so that it doesn't get in the way of access to the x part on the first tape, which it divides into not just k but $2k$ tracks. The second tape is needed only to help unspool data from ℓ cells on 2 tracks to 2ℓ cells on 1 track and vice-versa. The purpose is that whereas the actual tape heads of $M(x)$ might get spaced far apart, their virtualizations on the tracks of M_3 can be kept close together so that M_3 usually doesn't have to incur the full t-step overhead of the k-tapes-to-1 proof on every iteration. There are three further statements (but one can jump straight to the proof):

[1] Except for $\mathsf{DTIME}[O(n)] \subset \mathsf{NTIME}[n+1]$ for a technical reason that doesn't port to other machine models.

(a) In fact, M_3 doesn't need the third tape. It can adopt the "cartouche" idea from Assignment 3 to keep the code of M on a $(2k+1)$-st track and caterpillar it along as needed. So the code $\langle M \rangle$ part doesn't get in the way of the x part after all.

(b) For any fixed M but variable x, both M_3 and the resulting 2-tape simulator (call it M_2) can also be given the property that the locations of their tape heads at any time t' depend only on $n = |x|$ and on t', not on the particular bits of x. This property is called *obliviousness.* It is achievable even though the heads of M may be far from oblivious.

(c) Since the second tape is used only for data movements that are knowable in advance, the conversion of M_2 to equivalent Boolean circuits can be laid out for the first tape much as shown during the proof of the Cook-Levin Theorem. By the obliviousness, however, the location of the "six-cell lozenge" in row t' can be known in advance for any t'. Therefore we only need to give it once for each row, while the remaining (binary encodings of) characters in other cells are merely preserved. Likewise, the movements using Tape 2 are just directly coded by single wires in the circuit for each cell that traverse many rows at once. The upshot is that the number of gates needed for each row is constant, so the total number of gates and wires is order-of the running time of M_2, which is $O(t \log t)$. Therefore:

  - Every language accepted by an M running in time $t(n)$ has Boolean circuits of size $O(t(n) \log t(n))$.
  - The reduction in the Cook-Levin theorem can be computed in time $O(p(n) \log p(n))$, which is notably better than the $\tilde{O}(p(n)^2)$ time that was stated. It is asymptotically very efficient. It means that 3SAT is also complete for $\mathsf{NTIME}[\tilde{O}(n)]$ under reductions that are computable in deterministic $\tilde{O}(n)$ time.

Proof of Theorems 2 and 3. We describe diagonal languages $D_s \in \mathsf{DSPACE}[s_2(n)] \setminus \mathsf{DSPACE}[s_1(n)]$ and $D_t \in \mathsf{DTIME}[t_2(n)] \setminus \mathsf{DTIME}[t_1(n)]$ in terms of machines M_s and M_t that expressly run within the space bound $s_2(n)$ and time bound $t_2(n)$, respectively. Since their descriptions differ only in the initial detail, we describe both machines in the same breath. They each have the same three tapes as M_3 above, plus M_t has a fourth tape to count up to $t_2(n)$—which is possible by the definition of $t_2(n)$ being "reasonable" (in Debray's notes, or formally, "fully time constructible" in other sources).

> On any input x, taking $n = |x|$, M_s lays out $s_2(n)$ tape cells that its run of M_3 will be allowed to use, while M_t starts counting up to $t_2(n)$.
>
> Both machines try to decode $x = \langle M \rangle y$ for some Turing machine M. If this is not possible, they reject x.
>
> On success, they begin simulating $M_3(\langle M, x \rangle)$. Note that the "own code" $\langle M \rangle$ remains part of x, as does the "padding" y. Since the $\langle M \rangle$ part still gets copied to the third tape, this is a real-not-virtual run of M_3 with no overhead. If the simulation doesn't stay within the $s_2(n)$ marked-off cells in M_s, or takes longer than $t_2(n)$ steps in M_t, the overstep is immediately detected and the machine rejects x.
>
> Otherwise, the run of $M_3(\langle M, x \rangle)$ successfully completes. If M_3 accepts x, then M_s and M_t each *reject* x. If M_3 rejects x, that's when M_s and M_t *accept* x.

Considering first the case of space, M_s enforces the $s_2(n)$ space bound on itself, so $D_s =_{def} L(M_s) \in \mathsf{DSPACE}[s_2(n)]$. Now suppose we had $D_s \in \mathsf{DSPACE}[s_1(n)]$. Then there would be a DTM Q running in $s_1(n)$ space such that $L(Q) = D_s$. Now consider what happens when M_s runs on inputs of the form $x = \langle Q \rangle y$:

1. After taking $n = |x| = |\langle Q \rangle| + |y|$ and laying out $s_2(n)$ tape cells, M_s successfully decodes x into $\langle Q \rangle$ and y.

2. M_s seques into simulating $M_3(\langle Q, x \rangle)$ step-for-step. There is a constant C depending only on Q such that this takes at most $C + Cs_1(n)$ tape cells. What's important from Theorem 4 is that C doesn't change if the padding-y part of x changes.

3. The space usage by $M_3(\langle Q, x \rangle)$ still could overstep the boundaries laid out by M_s. But by $s_1(n) = o(s_2(n))$, for all C there is an n_0 such that whenever $n \geq n_0$, $C + Cs_1(n) \leq s_2(n)$. We may also wlog. suppose that $n_0 \geq |\langle Q \rangle|$.

4. So consider what happens on the particular input $x = \langle Q \rangle y$ with $y = 0^{n_0 - |\langle Q \rangle|}$. Then x has length $n = n_0$, so $C + Cs_1(n) \leq s_2(n)$.

5. Thus the simulation of $M_3(\langle Q, x \rangle)$ stays within the bound and runs to completion. So $M_s(x)$ gives the opposite answer to $M_3(\langle Q, x \rangle)$.

6. But $M_3(\langle Q, x \rangle)$ gives the same answer as $Q(x)$, so we get $M_s(x) \neq Q(x)$. This contradicts $L(Q_s) = D_s$.

As with the original "diagonal contradiction," this implies that the "quixotic" machine Q running in space $s_1(n)$ cannot exist. So D_s does not belong to $\mathsf{DSPACE}[s_1(n)]$.

The argument for time is entirely similar. Suppose Q accepts $D_t =_{def} L(M_t)$ in time $t_1(n)$. Then for any y, M_3 on input $\langle Q, x \rangle$ where $x = \langle Q \rangle y$ stops within $Ct_1(n) \log t_1(n) + C$ steps, where the constant C depends only on Q. Since $t_1(n) \geq n + 1$ by assumption about time functions, we can add in the initial $2n$ steps for decoding x into $\langle Q \rangle y$ and get n_0 such that for all $n \geq n_0$, $Ct_1(n) + C + 2n \leq t_2(n)$. Thus on the input $x = \langle Q \rangle 0^{n_0 - |\langle Q \rangle|}$ defined as before, the whole run by $M_t(x)$ finishes $M_3(\langle Q, x \rangle)$ and gives the opposite answer before the $t_2(n)$ "clock" counts all the way down and "rings." So $L(M_t) \neq L(Q)$, which contradicts $L(Q) = D_t$. $\square$

For some technical footnotes, there are analogous theorems for nondeterministic space and time whose proofs are trickier but give results that are even tighter—without the $\log(t)$ factor in the case of NTIME. The ALR ch. 27 notes give the proof of the latter but it is a "skim" at most. Various researchers including myself have devised realistic alternative models to the multitape TM that give a fully-tight deterministic time hierarchy without the $\log t(n)$ factor but none of them has "caught on." The multitape TM is actually IMHO pretty realistic already. Regarding the above proof, other sources make the "padding" y to be part of the machine. It could be extra "dummy states" that aren't reachable or even `#comments` in the code file of Q, just to make the code longer without changing its function.

So if this "padded diagonalization" technique works so tightly *within* any given complexity measure, why can't it work *between* them? Can we use it to get a language $D \in \mathsf{NP}$ that is not in $\mathsf{DTIME}[p(n)]$ for any polynomial p? We can restrict attention to $p(n) = n^r$ for various r, or if we want to be really strict on the time bound, $p(n) = n^r + r$. The problem is that r can vary—as illustrated on problem (3) on HW5. All attempts to make a machine N_3 analogous to M_3 that can take any machine M of any polynomial time n^r and run in one fixed polynomial time n^{r_0} have failed. We would need such an N_3 to imitate the above proof with a machine N_t that would give $D_t =_{def} L(N_t) \in \mathsf{NP}$. Getting $D_t \in \mathsf{NP}$ is the sticking point because having r vary does not help one achieve a polynomial-time NTM for D_t. There is a theorem by Dexter Kozen to the effect that "if $\mathsf{P} \neq \mathsf{NP}$ is provable at all then it is provable by diagonalization" but in https://rjlipton.wordpress.com/2014/11/26/cornell-cs-at-50/ I discuss whether it can be a "horse" or just the "cart" being pulled by some other proof.