CSE491/596 Lecture Fri. 11/20/20: Relativization and Randomness

First, picking up from the intended finish on Wednesday: If f is the 0-1 valued characteristic function of a language A, then we just write M^A. After any query y is submitted, M is left reading either 0 for "no, $y \notin A$" or 1 for "yes" on its tape. Many sources have M go to one of two special answer states q_y or q_n instead---this difference is immaterial.

Example: Via binary search, we can compute a function $f: \mathbb{N} \to \mathbb{N}$ in quadratic time using the oracle language $L_f = \{\langle x, w \rangle : w \geq f(x)\}$. (Well, in quadratic time in $|x| + |y|$ rather than $|x|$, where $y = f(x)$.) Even though there are exponentially many possibilities for $f(x)$ among numbers of length $< |x|^k$, say, that is 2^{n^k} possibilities, the binary search needs at most n^k iterations, each writing a query of length up to n^k. This is summarized by saying that s*earch reduces to decision* and is a major reason why languages not functions are used as the main objects in complexity theory.

Technical difference: $L'_f = \{\langle x, w \rangle : w \sqsubseteq f(x)\}$, where $\sqsubseteq$ means "is a prefix of."

If C is a class of *machines*, then we write C^B to be the class of languages $L(M^B)$ over all machines M. Thus for any language B,

- $\mathsf{P}^B = \left\{L(M^B) : \textit{The DTM M runs in polynomial time (with oracle B)}\right\}$
- $\mathsf{NP}^B = \left\{L(N^B) : \textit{The NTM N runs in polynomial time (with oracle B)}\right\}$
- $\mathsf{PSPACE}^B = \left\{L(M^B) : \textit{The DTM M runs in polynomial space (with oracle B)}\right\}$

If $A \in \mathsf{P}^B$ then we also write $A \leq_T^p B$ and say that A **polynomial-time Turing reduces** to B. (Older usage: A "Cook-reduces" to B, in constrast to saying "Karp-reduces" for $\leq_m^p$.)

Theorem: $\mathsf{P}^{\mathsf{TQBF}} = \mathsf{NP}^{\mathsf{TQBF}} = \mathsf{PSPACE}$.

Proof: PSPACE is contained in $\mathsf{P}^{\mathsf{TQBF}}$ because **TQBF** is PSPACE-complete under $\leq_m^p$ and an OTM can carry out a many-one reduction f by making just one query $y = f(x)$ and accepting x iff the oracle says "yes" to y. Now let $A \in \mathsf{NP}^{\mathsf{TQBF}}$ via a *nondeterministic* OTM (NOTM) N that runs in time $O(n^k)$. Then on any input x of length n, $N(x)$ can make up to n^k nondeterministic steps and can write queries y (which are quantified Boolean sentences that may be true or false) of length up to n^k. Without loss of generality, we may suppose the nondeterministic steps are binary-branching. A **deterministic**, **non-oracle** Turing machine M can accept A the following way:

1. Use n^k tape cells to cycle through all possible nondeterministic sequences $r \in \{0,1\}^{n^k}$.
2. For each r, simulate $N(x)$ with r deterministically until it writes a query y (which gets stored on a worktape of n^k cells that counts against the space bound.)
3. Answer y deterministically by solving TQBF in linear space, which here means space $O(n^k)$.
4. Then go back to step 2 until the next query y', answer it per step 3, and repeat until the computation path of $N^{\text{TQBF}}(x)$ with r finishes. If it accepts, then accept x; else try the next r, and finally reject if no r works.

This all adds up to $O(n^k)$ space used by M, so A belongs to PSPACE. ☒

Because $n \times n$ chess (with generalized 50-move rule) is PSPACE-complete and hence $\equiv_m^{\log}$ to TQBF, the upshot is that M^{TQBF} and N^{TQBF} are effectively cheating with a perfect chess program. In actual chess cheating, doing so wipes out the natural difference in strength between players. The same happens here insofar as an NTM is often "stronger than" a DTM, certainly if we believe $\mathsf{P} \neq \mathsf{NP}$ in the absence of an oracle. The subversive impact of $\mathsf{P}^{\text{TQBF}} = \mathsf{NP}^{\text{TQBF}}$ comes from the following "meta-theorem":

Meta-Theorem 1: Every theorem about (un-)decidability and reductions and relationships among deterministic and nondeterministic time and space that is taught in this course **relativizes**, meaning that for any oracle language B (or oracle function f), the statements hold when all machines and classes are defined with access to B (or f). For example, the relativized diagonal language

$$D^B = \left\{\langle M\rangle : M \textit{ is a deterministic OTM and } M^B \textit{ does not accept } \langle M\rangle\right\}$$

is not "decidable in B" nor even "computably enumerable in B"---the latter meaning there is no OTM Q such that $L(Q^B) = D^B$. The proof is essentially the same. This is because the proof only pays attention to the input/output behavior of the programs, taking no care about how they process information internally---and might cheat! Thus, for any B, we have:

1. $\mathsf{RE}^B \neq \mathsf{co\text{-}RE}^B$
2. $\mathsf{RE}^B \cap \mathsf{co\text{-}RE}^B = \mathsf{REC}^B$.
3. $A_{TM}^B = \left\{\langle M, w\rangle : w \in L(M^B)\right\}$ is complete for RE^B under $\leq_m^{\log}$. (The reductions do not need to use the oracle---they are only translating syntax with no regard to the oracle feature.)

It is even possible to define a "relativized version" 3SAT^B in which special formula variables reference the oracle's yes/no answers and prove by extending the Cook-Levin idea that it is complete for NP^B

(again under $\leq_m^{\log}$ reductions that do not need the oracle for themselves). This is broadly known as the "principle of relativization for elementary methods." **But** this means:

Meta-Theorem 2: Such elementary methods cannot prove $\mathsf{P} \neq \mathsf{NP}$.

Proof: If they could, then by the principle of relativization, they would prove $\mathsf{P}^B \neq \mathsf{NP}^B$ for all languages B. But we just proved that $\mathsf{P}^{\mathsf{TQBF}} = \mathsf{NP}^{\mathsf{TQBF}}$. ☒

More briefly put: the formal system behind CSE491/596 can prove its own inability to prove the conjectured side of the greatest problem in the field (unless CSE491/596 is inconsistent). It cannot prove the other side $\mathsf{P} = \mathsf{NP}$ either, owing to the counterpart result:

Theorem: We can build a language B such that $\mathsf{P}^B \neq \mathsf{NP}^B$.

[Proof Sketch: For any language B, NP^B includes the language

$$L^B = \left\{0^n : (\exists y)[|y| = n \wedge y \in B]\right\}.$$

Now suppose we have any finite set F and deterministic OTM M that runs in polynomial time $p(n)$. We can choose a number n such that $2^n > p(n)$ and all strings F have length $< n$. Now simulate M^F on input 0^n while keeping track of (i) all strings y of length n that $M^F\left(0^n\right)$ queries, (ii) if $M^F\left(0^n\right)$ queries a string of length $> n$, let m be the length of the longest such string, else $m = n+1$, and (iii) whether M^F accepts 0^n. Note that all queries of length n and higher get answered "no" since F has no strings of those lengths.

- If M accepts 0^n, then do nothing at length n. We have $0^n \notin L^F$ but $0^n \in L(M^F)$.
- If M rejects 0^n, then by $2^n > p(n)$, there must be some string $z \in \{0,1\}^n$ that was not queried. Add z to F.
- Either way, there must be some string $z' \in \{0,1\}^m$ that M did not query either. Add z' to F to make F'.
- Then the second step gave us $0^n \in L^{F'} \setminus L\left(M^{F'}\right)$ and the third step did nothing to change that.

Moreover, because we added 1^m to F', if we repeat this process with a new polynomial $p'(n)$-time machine M' using oracle F', all further alterations will occur at lengths $> m$ and hence not disturb the way we guaranteed that M cannot agree with L^F on the string 0^n. Thus we can repeat the process on M' without disturbing how we "digonalizes against" M. As we continue, the sequence of finite sets builds a language B. Since every polynomial-time OTM eventually gets tried and defeated, we get that the final L^B does not belong to P^B. ☒]

Note, by the way, that there is no contradiction or paradox in the *fact* of having $\mathsf{P}^B \neq \mathsf{NP}^B$ yet $\mathsf{P}^A = \mathsf{NP}^A$ (with $A =$ TQBF). The issue is what range of techniques can be used to *prove* it. The great lack in the field as it stands is that no one has been able to formulate an incisive measure of how much information has been incrementally internally "processed." We have been blocked from doing any better than "Intelligent Design" theorists who have tried to prove that Nature must have "cheated with divine input" in order to have created complex information structures in (only?) 13.8 billion years. But we have come to understand the blockages---called **barriers**---better and better. Most of the ones after the "relativzation barrier" have to do with *randomness* and its relation to computational *hardness*.

Preview of Monday:

Randomized Complexity

Matrix $AB = C$ example by checking $A \cdot Bu = Cu$ for multiple vectors u. Idea is that multiplying two $n \times n$ matrices by the usual method takes time order-of n^3 but multiplying a matrix by a vector is only order-of n^2. We have to do the latter three times so it's $3n^2$, then doing it for k different random choices of u makes $3kn^2$, but for large enough n we can arrange $3k \ll n$ and yet the chance of getting all k choices of u giving $A \cdot Bu = Cu$ even when $AB \neq C$ is tiny enough to downplay. The lecture will do this rigorously when the matrices are over the field mod-2, which is actually the worst case.

This will be one lecture on classical randomized complexity and the class BPP, as a setup for quantum randomized complexity and the class BQP, where the matrices get exponentially bigger but Nature doesn't seem to mind...