DD2445 LECTURE 5

Last time

(1) Diagonalization

Table of all TMs running within specified resource bounds and all inputs

	1	2	3	4	5	6
M_1	$\neq$					
M_2		$\neq$				
M_3			$\neq$			
M_4				$\neq$		
M_5					$\neq$	

cell (i,j) = output of M_i on j

Prove that certain language L is not decidable within resource bound by showing that in each row one cell contains wrong answer

Namely on the diagonal – DIAGONALIZATION

(2) Time hierarchy theorem
More computation time $\Rightarrow$ more problems solved

(3) Ladner's theorem
If $P \neq NP$, then there are (infinitely many) complexity classes in between

(Only sketched proof in class – details on problem set)

What exactly is "proof by diagonalization"?

How far can diagonalizing techniques take us? Maybe be prove $P \neq NP$ if work hard enough?!

Provably NO, unfortunately...

Our diagonalization proofs relied on:

I Representation of TMs by strings (every integer is a TM)

II Efficient simulation of TM by other (universal) TM without much overhead (in time or space)

Such an approach works even if we give TM access to certain subroutines for free and don't charge for time spent in such calls during running time analysis

Such TMs are known as "oracle Turing machines"

Might sound very strange - for us it will just be a program with a subroutine that can be called free of charge

DEF Oracle TM (informal - see textbook)

An oracle Turing machine is a usual TM except

- has special read-write oracle tape
- has special oracle state

To execute M, specify oracle language $O \subseteq \{0,1\}^*$

At any time, M can

- write string y on oracle tape (takes several steps)
- jump to oracle state (one time step)
- get answer whether $y \in O$ or not written as bit 1 if $y \in O$, 0 if $y \notin O$ on oracle tape (one time step)

Output of M on x when run with oracle O denoted $M^O(x)$

Nondeterministic oracle machines defined analogously

$$P^O = \left\{ \begin{array}{l} \text{all languages decidable by poly-time} \\ \text{deterministic TM with oracle } O \end{array} \right\}$$

$$NP^O = \left\{ \begin{array}{l} \text{all languages decidable by} \\ \text{poly-time nondeterministic TM} \\ \text{with oracle } O \end{array} \right\}$$

Also say that TM M has "oracle access" to language O.

OII

Examples

(1) UNSAT $\in P^{SAT}$

Write down formula on oracle tape

Make query to SAT

Give the opposite answer as output

(2) If $O \in P$ then $P^O = P$

Oracle calls are not needed

Can compute answer by simulating machine deciding O in poly time

(3) $EXPCOM = \{\langle M, x, 1^n \rangle :$ M outputs 1 on x within 2^n steps $\}$

Then $\boxed{P^{EXPCOM} = NP^{EXPCOM} = EXP}$

Recall $EXP = \bigcup_{c \in \mathbb{N}} DTIME(2^{n^c})$

Proof: Suppose $L \in DTIME(2^{n^c})$ decided by M

Write down M, x, and then 1^{n^c} (i.e. n^c 1's) on oracle tape

Can be done in poly time

Query EXPCOM and answer the same

$\Longrightarrow$ $EXP \subseteq P^{EXPCOM}$

$P^O \subseteq NP^O$ for any oracle language O (why?)

Suppose $L' \in NP^{EXPCOM}$ decided by M' running in time $O(n^k)$

At most $2^{O(n^k)}$ nondeterministic choices which is exponential

At most that many oracle calls – can also be computed in exponential time

Exponential $\times$ exponential = exponential, so $L' \in EXP$.

OIII

Regardless of what the oracle O is

I and II holds for oracle TMs (if simulating machine is also given the oracle O)

Hence, any theorem about TMs that uses only I + II holds for oracle TMs (the theorem RELATIVIZES).

The answer to $P \stackrel{?}{=} NP$ can't be a relativizing theorem, since there are oracles to flip the answer both ways!

THEOREM (Baker, Gill, Solovay '75)
There exist oracles A and B s.t.
$P^A = NP^A$ and $P^B \neq NP^B$

Proof Set A to be EXPCOM

For any language B, let
$$U_B = \{ 1^n : \exists x \text{ s.t. } |x| = n \text{ and } x \in B \}$$

For any B, $U_B \in NP^B$
On input 1^n, guess x of length n, write on oracle tape, query B.

Want to build B s.t. $U_B \notin P^B$.
If so, proof finished

OIV

High-level intuition:

Any TM for U_B has to run in subexponential time. Can only query vanishing small part of strings of length $\{0,1\}^n$ - exponentially many. Make sure any TM "queries the wrong strings."

Construction of B

M_i: Turing machine encoded by (binary expansion of) integer i

Construct B in stages. Stage i will make sure M_i doesn't solve U_B in time $\leq 2^n / 10$.

Initially $B := \emptyset$, $i := 1$.

stage i:

B contains finite # strings so far

Fix n s.t. no string of length $\geq n$ ~~[illegible]~~ has had its status wrt B decided.

Run M_i on 1^n for $2^n/10$ steps.

Oracle queries y

case 1 status of y decided in previous stages — answer accordingly

case 2 status of y undecided — answer $y \notin B$

~~[illegible]~~

Suppose M_i finishes on 1^n and answers b

Note – Have only decided status of $\leq 2^n/10$ strings in $\{0,1\}^n$

– For all of them, answered no.

If $b = 1$, decide that no string in $\{0,1\}^n$ is in B

$\Rightarrow 1^n \notin U_B$, and M_i is wrong on 1^n

If $b = 0$, pick some string $y \in \{0,1\}^n$ not queried and decide that $y \in B$

$\Rightarrow 1^n \in U_B$, so M_i is wrong on 1^n

Only remaining worry. What if we didn't allow M_i to finish? What if it runs in polynomial time p s.t.

$p(n) \geq 2^n/10$ for this n?

The TM M_i will be repeated infinitely often for larger and larger i. Finally, will get some n' s.t. $p(n') \ll 2^n/10$, and for this n' the proof will work.

Recall our TM encoding has "stop marker" after which junk allowed, so each TM encoded by infinitely many integers

SUMMING UP

- Diagonalization can be used to separate cplx classes.
- In particular, $P \neq EXP$
- If $P \neq NP$, then there is infinite hierarchy of cplx classes between P & NP
- But diagonalization not enough to settle $P \stackrel{?}{=} NP$, since any such proof works for oracle TMs and different ~~TMs~~ oracles give different answers to $P \stackrel{?}{=} NP$...

NEXT ON THE AGENDA

- Memory consumption as the limiting factor
- Space-bounded cplx classes