

CSci 5403

COMPLEXITY THEORY

LECTURE V: DIAGONALIZATION

Theorem (Cantor). Let $f : S \rightarrow \mathcal{P}(S)$ map elements of the set S to subsets of S . Then f is not onto.

In particular, let $D_f = \{ y \mid y \notin f(y) \}$. There is no $s \in S$ such that $f(s) = D_f$:

Suppose $f(s) = D_f$. Then $s \in D_f \Leftrightarrow s \notin f(s) \Leftrightarrow s \notin D_f$.

Example: Let $S = \{0,1,2\}$ and let $f(0) = \emptyset$, $f(1) = \{1,2\}$, $f(2) = \{0,1\}$.

y	$0 \in f(y)?$	$1 \in f(y)?$	$2 \in f(y)?$
0	N	N	N
1	N	Y	Y
2	Y	Y	N
D_f	Y	N	Y

Theorem (Turing). The diagonal language $D_{TM} = \{ \langle M \rangle : M \text{ does not accept on input } \langle M \rangle \}$ is undecidable.

Proof. Let $f_{TM} : \{0,1\}^* \rightarrow \mathcal{P}(\{0,1\}^*)$ map a TM's code to the language it decides.

If D_{TM} is decidable, there must be a TM H such that $f_{TM}(\langle H \rangle) = D_{TM}$. But D_{TM} is the diagonal set for f_{TM} .

Cantor, Gödel, and Turing used it... now let's try!
(Insert cheesy 1970s theme music)

TIME HEIRARCHY THEOREM

Theorem. Let f and g be time constructible, and let $g(n) = \omega(f(n)^2)$. Then $DTIME(f(n)) \subsetneq DTIME(g(n))$.

Corollary. $P \neq EXP$.

Proof. Let $D_f = \{ \langle M \rangle : M \text{ does not accept } \langle M \rangle \text{ in } f(|\langle M \rangle|) \text{ steps.} \}$. $D_f \notin DTIME(f(n))$, since it is the diagonal set for L_f mapping a TM to the set it decides in time $f(n)$.

But recall that we can simulate a TM that runs in time t in time $O(t^2)$. Thus $D_f \in DTIME(f(n)^2)$.

SPACE HEIRARCHY THEOREM

Theorem. Let f and g be space constructible, and let $g(n) = \omega(f(n))$. Then $SPACE(f(n)) \subsetneq SPACE(g(n))$.

Corollary. $L \neq PSPACE$.

Proof. Analogous. We can simulate a machine using space $s(n)$ in space $O(s(n))$.

NTIME HEIRARCHY THEOREM

Theorem. Let f and g be time constructible, and let $g(n) = \omega(f(n)^2)$. Then $NTIME(f(n)) \subsetneq NTIME(g(n))$.

Proof. Let $D_f = \{ \langle M \rangle : M \text{ does not accept } \langle M \rangle \text{ in } f(|\langle M \rangle| \text{ steps}) \}$. $D_f \notin NTIME(f(n))$, since this is the diagonal set for L_f mapping a TM to the set it accepts in time $f(n)$.

But recall that we can simulate a NTM that runs in time t in time $O(t^2)$. Thus $D_f \in NTIME(g(n))$.

NTIME HEIRARCHY THEOREM

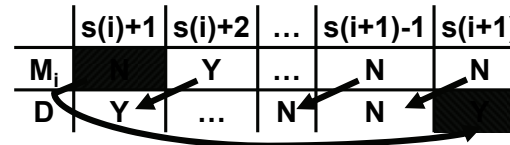
We don't know how to simulate a "coNTIME($f(n)$)" machine in $NTIME(g(n))$, where $g(n) = o(2^{f(n)})$!

Instead of differing from M_i at string i , we'll build D that is different from M_i in $(s(i), s(i+1)]$.

	$s(1)$	$s(2)$	$s(3)$...	$s(i)$	$s(i+1)$...	$s(i+1)$
M_1	Y	N	Y	...	N	Y	...	Y
M_2	N	Y	Y	...	Y	N	...	N
\vdots								
M_i	Y	Y	Y	...	N	N	...	N
D	N	N	Y	...	N	Y	...	Y

NTIME HEIRARCHY THEOREM

Key idea: make $s(i+1)$ big enough that we can simulate a coNTIME machine on input $s(i)$.



$$\text{Let } s(i) = \begin{cases} 1, & \text{if } i = 0 \\ 2^{f(s(i-1))} f(s(i-1))^2, & \text{otherwise} \end{cases}$$

Let NTM $D(1^n) =$
 find i such that $s(i) < n \leq s(i+1)$
 if $n = s(i+1)$ return $\text{not}(M_i(1^{s(i)}))$ ← simulate all branches $f(s(i))$ steps
 else return $\text{NSIM}_f(M_i, 1^{n+1})$

NTIME HEIRARCHY THEOREM

Theorem. Let f and g be time constructible, and $g(n) = \omega(f(n+1)^2)$. Then $\text{NTIME}(f(n)) \subsetneq \text{NTIME}(g(n))$.

Proof. The language accepted by machine D is not in $\text{NTIME}(f(n))$ by construction.

But the running time of D on input size n is:

Case 1: $n \notin s(\mathbb{N})$: Time to simulate an NTM for $f(n+1)$ steps = $O(f(n+1)^2)$.

Case 2: $n = s(i+1)$: Time to simulate all branches of length $f(s(i)) = 2^{f(s(i))}f(s(i))^2 = s(i+1) = O(n)$.

LADNER'S THEOREM

Theorem. If $P \neq \text{NP}$, then there is a language $L \in \text{NP} \setminus P$ that is not NP-complete.

Proof.

Suppose $P \neq \text{NP}$. Let $H: \mathbb{N} \rightarrow \mathbb{N}$ and consider the set $\text{SAT}_H = \{ (\psi, 1^k) : \psi \in \text{SAT} \text{ and } k = n^{H(n)} \}$.

If $H(n) = c (= O(1))$, then $\text{SAT}_H \notin P$. (why?)

If $H(n) = \omega(1)$ then SAT_H cannot be NP-complete: The reduction f from SAT to SAT_H must reduce the size of the formula.

We will build an $H(n) = \omega(1)$ so that $\text{SAT}_H \notin P$.

For a set S , define $S[n] = \{ s \in S : |s| < n \}$.

$H(n) = \min \{ j < n : M_j \text{ decides } \text{SAT}_H[n] \text{ in time } j^n \}$

if M_i decides SAT_H in poly(n) time, then $H(n) \leq i$.

Suppose $H(n) = O(1)$, i.e. for $n > N$, $H(n) \leq k$. Then there is a $k^2 n^k$ -time algorithm for SAT_H . (why?)

Combined we have $\text{SAT}_H \notin P$ and SAT_H is not NP Complete.

What's wrong with this proof?

Fix: make it easy to compute $H(n)$:

To test M_i on only poly(n) inputs:
replace $\text{SAT}_H[n]$ with $\text{SAT}_H[\lg n]$

To test each input for only poly(n) steps:
replace $j < n$ with $j < \lg n / \lg \lg n$

$H(n) = \min \{ j < \lg n / \lg \lg n : M_j \text{ decides } \text{SAT}_H[\lg n] \text{ in time } j^n \}$

LIMITS OF DIAGONALIZATION

All of the previous theorems are true because TMs can be enumerated and simulated.

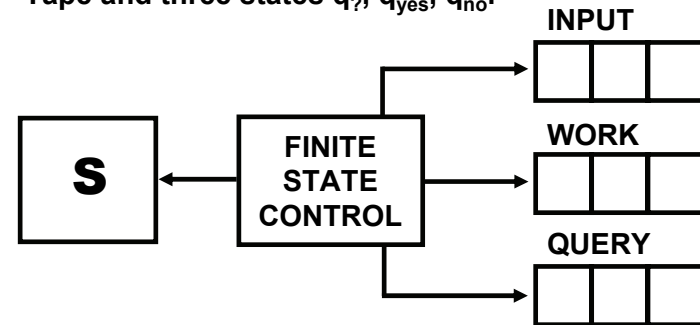
Thus, any proof that uses only these facts must also hold for any computational model that also has these properties.

We will show an enumerable, simulatable model where $P=NP$, and one where $P \neq NP$.

So a proof using only diagonalization cannot resolve P vs NP .

ORACLE TM

An oracle Turing Machine is a TM with a query Tape and three states $q_?$, q_{yes} , q_{no} .



With an oracle for set S , it goes in one step from state $q_?$ with x on the query tape to $q_{x \in S}$.

We denote M running with oracle S by M^S .
 M^S is pronounced “ M relative to S ”

For fixed O , we can define complexity classes NP^O , P^O , $PSPACE^O$, etc.

e.g. $S \in P^O$ iff \exists poly(n)-time P . $L(P^O) = S$

$S \in NP^O$ iff \exists poly(n)-time V , poly q .
 $S = \{ x \mid \exists y \in \{0,1\}^{q(|x|)}. V^O(x,y) \text{ accepts} \}$

Claim. Relative to a fixed oracle O , the set of OTMs is enumerable and simulatable.

Theorem (Baker, Gill, Solovay). There exist oracles A and B such that $P^A = NP^A$ and $P^B \neq NP^B$.

$PA = NPA$

Let A be any PSPACE-complete problem.

We know that $P^A \subseteq NP^A$. (why?)

Also $NP^A \subseteq NPSPACE$, since it only takes poly(n) space to answer queries to A .

But notice that $NPSPACE = PSPACE \subseteq P^A$.

$P^B \neq NP^B$

We construct a set B so that the language $U_B = \{1^n : \exists x \in B. |x| = n\}$ is not in P^B .

Note that for any set B , we have $U_B \in NP^B$.

We build B as a union $\bigcup_{i \in \mathbb{N}} B_i$.

$B_0 = \emptyset$.

To build B_i :

let $n = 1 + \max \{ |x| : x \in B_j \cup Q_j, j \leq i \}$.

Run $M_i^{B_i}(1^n)$ for $2^n/10$ steps,

adding every oracle query to Q_i .

If M_i accepts, $B_i = \emptyset$. Else $B_i = \{0,1\}^n \setminus Q_i$.

$P^B \neq NP^B$

	1	...	n_i	...	$f(n_i)$	n_{i+1}	...
M_1	Y	...					
\vdots							
M_i			N				
M_{i+1}						Y	
U_B	N	...	Y	...	N	N	
B	\emptyset	...	$\{0,1\}^n$	\emptyset	

RELATIVIZATION

If an argument about two complexity classes is true relative to any oracle, we say it relativizes.

The oracles A and B show that any argument that relativizes cannot resolve P vs NP .

Diagonalization relativizes because it doesn't care about how a machine computes.

Results that "care": $NP \leq_p 3SAT$, $IP = PSPACE$, $NP = PCP(1, \lg n)$...

CS5403.info