Umans
Complexity Theory Lectures

Lecture 16:
Relativization

Approaches to open problems

- Almost all major open problems we have seen entail proving **lower bounds**
  - **P ≠ NP**
  - **L ≠ P**
  - **P ≠ PSPACE**
  - **NC** proper
  - **BPP ≠ EXP**
  - **PH** proper
  - **EXP * P/poly**
  - **P = BPP** *
  - **NP = AM** *

• we know circuit lower bounds imply derandomization

• more difficult (and recent): derandomization implies circuit lower bounds!

Approaches to open problems

- two natural approaches
  - simulation + diagonalization (uniform)
  - circuit lower bounds (non-uniform)
- no success for either approach as applied to date

Why?

Approaches to open problems

in a precise, formal sense
these approaches are
too powerful !

- if they could be used to resolve major open problems, a side effect would be:
  - proving something that is false, or
  - proving something that is believed to be false

Oracle Turing Machines

- Oracle Turing Machine (OTM):
  - Deterministic multitape TM M with special "query" tape
  - special states $q_?$, q_{yes}, q_{no}
  - on input x, with oracle language A
  - M^A runs as usual, except…
  - when M^A enters state $q_?$:
    - y = contents of query tape
    - $y \in A \Rightarrow$ transition to q_{yes}
    - $y \notin A \Rightarrow$ transition to q_{no}

Relativization

- Many proofs and techniques we have seen **relativize**:
  - they hold after replacing all TMs with oracle TMs that have access to an oracle A
  - e.g. $\mathbf{L^A \subset P^A}$ for all oracles A
  - e.g. $\mathbf{P^A \neq EXP^A}$ for all oracles A

Relativization

- Idea: design an oracle A relative to which some statement is *false*
  - implies there can be no **relativizing proof** of that statement
  - e.g. design A for which $\mathbf{P^A = NP^A}$
- Better: also design an oracle B relative to which statement is *true*
  - e.g. also design B for which $\mathbf{P^B \neq NP^B}$
  - implies no **relativizing proof** can resolve truth of the statement either way !

Relativization

- Oracles are known that falsify almost every major conjecture concerning complexity classes
  - for these conjectures, non-relativizing proofs are required
  - almost all known proofs in Complexity relativize (sometimes after some reformulation)
  - notable exceptions:
    - The PCP Theorem
    - IP = PSPACE
    - most circuit lower bounds (more on these later)

Relativization: Oracles for **P** vs. **NP**

- Goal:
  - oracle A for which $\mathbf{P^A = NP^A}$
  - oracle B for which $\mathbf{P^B \neq NP^B}$

- conclusion: resolving

  P vs. NP

  requires a non-relativizing proof

Relativization: Oracles for **P** vs. **NP**

Theorem: there exists an oracle A for which $\mathbf{P^A = NP^A}$.

Proof: For $\mathbf{P^A = NP^A}$ need A to be powerful

- warning: intend to make **P** more powerful, but also make **NP** more powerful.
- e.g. A = SAT doesn't work
- however A = QSAT works:

(QSAT = satisfiability of quantified Boolean formulas)

$$\mathbf{PSPACE \subset P^{QSAT} \subset NP^{QSAT} \subset NPSPACE}$$

and we know $\mathbf{NPSPACE \subset PSPACE}$

QED

Relativization: Oracles for **P** vs. **NP**

Theorem: there exists an oracle B for which

$$\mathbf{P^B \neq NP^B}.$$

Proof:

– define

$$L = \{1^i : \exists\, x \in B \text{ s.t. } |x| = i\}$$

– we will show $L \in \mathbf{NP^B - P^B}$.

– easy: $L \in \mathbf{NP^B}$ (no matter what B is)

Relativization: Oracles for **P** vs. **NP**

– Idea: Design B by diagonalizing against all

"$\mathbf{P}^{B}$ machines"

– $M_1, M_2, M_3, \ldots$ is an enumeration of deterministic polynomial time Oracle TMs (OTMs)

– B_i will be those strings of length $\leq i$ in B

– We build B_i after simulating machine M_i

Relativization: Oracles for **P** vs. **NP**

Recall we defined: $L = \{1^i : \exists\, x \in B \text{ s.t. } |x| = i\}$

Proof (continued):

- maintain "exceptions" X that must not go in B
- initially $X = \{\}$, $B_0 = \{\}$

Stage i:

- simulate $M_i(1^i)$ for $i^{\log i}$ steps
- when M_i makes an oracle query y:
  - if $|y| < i$, answer using B_{i-1}
  - if $|y| \geq i$, answer "no"; add y to X
- if simulated $\mathbf{M_i}$ **accepts** $\mathbf{1^i}$ then set $B_i = B_{i-1}$
- if simulated $\mathbf{M_i}$ **rejects** $\mathbf{1^i}$, then

  set $B_i = B_{i-1} \cup \{x \in \{0,1\}^i : x \notin X\}$

Relativization: Oracles for **P** vs. **NP**

Recall we defined: $L = \{1^i : \exists\ x \in B \text{ s.t. } |x| = i\}$

Proof (continued):

Case 1: if M_i accepts 1^i :

- then $B_i = B_{i-1}$ which insures no strings of length i in B.
- therefore $1^i \notin L$, and so **M_i does not decide L**

Relativization: Oracles for **P** vs. **NP**

Recall we defined: $L = \{1^i : \exists\, x \in B \text{ s.t. } |x| = i\}$

Proof (continued):

Case 2: if M_i rejects 1^i : we ensure *some* string of length i in B

In this case we set $B_i = B_{i-1} \cup \{x \in \{0,1\}^i : x \notin X\}$

and $|X|$ is at most $\Sigma_{j \le i}\, j^{\log j} << 2^i$

- therefore $1^i \in L$, and so **M_i does not decide L**

Relativization: Oracles for **P** vs. **NP**

Recall we defined: $L = \{1^i : \exists\, x \in B \text{ s.t. } |x| = i\}$

Proof (conclusion):

B defined by diagonalizing against all "$\mathbf{P^B}$ machines"

$M_1, M_2, M_3, \ldots$

an enumeration of deterministic Oracle TMs (OTMs).

B is defined so M_i does not decide L for both Cases:

- Case 1 (M_i accepts 1^i)
- Case 2 (M_i rejects 1^i)

We do this for each OTM M_i and so Conclude: $L \notin P^B$

Hence $L \in NP^B - P^B$,

and so $P^B \neq NP^B$. QED