Computational Complexity Theory

Lecture 9: Relativization

Department of Computer Science,
Indian Institute of Science

Recap: NP-intermediate problems

- Definition. A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.

- Theorem. *(Ladner 1975)* If P $\neq$ NP then there is a NP-intermediate language.

  Proof. A delicate argument using diagonalization.

Limits of diagonalization

- Like in the proof of P $\neq$ EXP, can we use diagonalization to show P $\neq$ NP ?

Limits of diagonalization

- Like in the proof of P $\neq$ EXP, can we use diagonalization to show P $\neq$ NP ?
- The answer is No, if one insists on using only the two features of diagonalization.
- The proof of this fact uses diagonalization and the notion of *oracle Turing machines*!

Oracle Turing Machines

- Definition: Let $L \subseteq \{0,1\}^*$ be a language. An *oracle TM* M^L is a TM with a special query tape and three special states q_{query}, q_{yes} and q_{no} such that whenever the machine enters the q_{query} state, it immediately transits to q_{yes} or q_{no} depending on whether the string in the query tape belongs to L. (M^L has *oracle access* to L)

Oracle Turing Machines

- Definition: Let $L \subseteq \{0,1\}^*$ be a language. An *oracle TM* M^L is a TM with a special query tape and three special states q_{query}, q_{yes} and q_{no} such that whenever the machine enters the q_{query} state, it immediately transits to q_{yes} or q_{no} depending on whether the string in the query tape belongs to L. (M^L has *oracle access* to L)

- Think of physical realization of M^L as a device with access to a subroutine that decides L. We don't count the time taken by the subroutine.

Oracle Turing Machines

- We can define a nondeterministic Oracle TM similarly.

- "Important note": Oracle TMs (deterministic/nondeterministic) have the same two features used in diagonalization: For any **fixed** $L \subseteq \{0,1\}^*$,

    1. There's an *efficient universal TM* with oracle access to L,
    2. Every M^L has *infinitely many representations*.

Complexity classes using oracles

- Definition: Let $L \subseteq \{0,1\}^*$ be a language. Complexity classes P^L, NP^L and EXP^L are defined just as P, NP and EXP respectively, but with TMs replaced by oracle TMs with oracle access to L in the definitions of P, NP and EXP respectively. For e.g., $\overline{SAT} \in P^{SAT}$.

Complexity classes using oracles

- Definition: Let $L \subseteq \{0,1\}^*$ be a language. Complexity classes P^L, NP^L and EXP^L are defined just as P, NP and EXP respectively, but with TMs replaced by oracle TMs with oracle access to L in the definitions of P, NP and EXP respectively. For e.g., $\overline{SAT} \in P^{SAT}$.

- Such complexity classes help us identify a class of complexity theoretic proofs called *relativizing proofs*.

Relativization

Relativizing results

- Observation: Let $L \subseteq \{0,1\}^*$ be an arbitrarily fixed language. Owing to the "Important note", the proof of $P \neq EXP$ can be easily adapted to prove $P^L \neq EXP^L$ by working with TMs with oracle access to L.
- We say that the $P \neq EXP$ result/proof ***relativizes***.

Relativizing results

- Observation: Let $L \subseteq \{0,1\}^*$ be an arbitrarily fixed language. Owing to the "Important note", the proof of $P \neq EXP$ can be easily adapted to prove $P^L \neq EXP^L$ by working with TMs with oracle access to L.
- We say that the $P \neq EXP$ result/proof ***relativizes***.

- **Observation:** Let $L \subseteq \{0,1\}^*$ be an arbitrarily fixed language. Owing to the 'Important note', any proof/result that uses only the two features of diagonalization ***relativizes***.

Relativizing results

- If there is a resolution of the P vs. NP problem <u>using **only** the two features</u> of diagonalization, then such a proof must relativize.
- Is it true that

  - either $P^L = NP^L$ for every $L \subseteq \{0,1\}^*$,
  - or $P^L \neq NP^L$ for every $L \subseteq \{0,1\}^*$?

Relativizing results

- If there is a resolution of the P vs. NP problem using **only** the two features of diagonalization, then such a proof must relativize.
- Is it true that

  - either $P^L = NP^L$ for every $L \subseteq \{0,1\}^*$,

  - or $P^L \neq NP^L$ for every $L \subseteq \{0,1\}^*$?

Theorem *(Baker, Gill & Solovay 1975)*: The answer is No. Any proof of P = NP or P ≠ NP must **not** relativize.

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: Using diagonalization!

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: Let $A = \{(M, x, 1^m)$: M accepts x in 2^m steps$\}$.
- A is an EXP-complete language under poly-time Karp reduction. *(simple exercise)*

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: Let $A = \{(M, x, 1^m)$: M accepts x in 2^m steps$\}$.
- A is an EXP-complete language under poly-time Karp reduction.

- Then, $P^A = EXP$.
- Also, $NP^A = EXP$. Hence $P^A = NP^A$.

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: Let $A = \{(M, x, 1^m)$: M accepts x in 2^m steps$\}$.
- A is an EXP-complete language under poly-time Karp reduction.

- Then, $P^A = EXP$.
- Also, $NP^A = EXP$. Hence $P^A = NP^A$.

Why isn't $EXP^A = EXP$?

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: For any language B let

$$L_B = \{1^n : \text{there's a string of length } n \text{ in } B\}.$$

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: For any language B let

$$L_B = \{1^n : \text{there's a string of length } n \text{ in } B\}.$$

- Observe, $L_B \in NP^B$ for any B. (Guess the string, check if it has length n, and ask oracle B to verify membership.)

Baker-Gill-Solovay theorem

- Theorem: There exist languages A and B such that $P^A = NP^A$ but $P^B \neq NP^B$.
- Proof: For any language B let

$$L_B = \{1^n : \text{there's a string of length } n \text{ in } B\}.$$

- Observe, $L_B \in NP^B$ for any B.
- We'll construct B (using diagonalization) in such a way that $L_B \notin P^B$, implying $P^B \neq NP^B$.

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the status of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps. Moreover, n will grow monotonically with stages.

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the <u>status</u> of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps. Moreover, n will grow monotonically with stages.

whether or not a string belongs to B

The machine with oracle access to B that is represented by i

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the status of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps. Moreover, n will grow monotonically with stages.

- Clearly, a B satisfying the above implies $L_B \notin P^B$. Why?

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the status of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps. Moreover, n will <u>grow monotonically</u> with stages.

- Stage i: Choose n larger than the length of any string whose status has already been decided. Simulate M_i^B on input 1^n for $2^n/10$ steps.

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the status of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps.

- Stage i: If M_i^B queries oracle B with a string whose status has already been decided, answer consistently.

If M_i^B queries oracle B with a string whose status has not been decided yet, answer 'No'.

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the status of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps.

- Stage i: If M_i^B outputs 1 within $2^n/10$ steps then don't put any string of length n in B.

  If M_i^B outputs 0 or doesn't halt, put a string of length n in B.

(This is possible as the status of at most $2^n/10$ many length n strings have been decided during the simulation)

Constructing B

- We'll construct B in stages, starting from Stage 1.
- Each stage determines the status of finitely many strings.
- In Stage i, we'll ensure that the oracle TM M_i^B doesn't decide 1^n correctly (for some n) within $2^n/10$ steps.

- Homework: In fact, we can assume that $B \in EXP$.