Lecture 2 - Ladner's Theorem, Oracle TMs, Relativization

Rafael Oliveira
rafael.oliveira.teaching@gmail.com
University of Waterloo

CS 860 - Graduate Complexity Theory
Fall 2022

Overview

- Ladner's Theorem: NP-intermediate problems
- Oracle TMs & Relativization: limits of diagonalization

NP-intermediate languages

Theorem (**[Ladner 1975]**)

If P $\neq$ NP then there exists a language $L \in$ NP $\setminus$ P that is not NP-complete.[1]

[1] Ladner actually proved more - a hierarchy of intermediate problems.

NP-intermediate languages

Theorem ([Ladner 1975])

If P $\neq$ NP then there exists a language $L \in$ NP $\setminus$ P that is not NP-complete.[1]

1. For every function $h : \mathbb{N} \to \mathbb{N}$, let

$$\mathsf{SAT}_h := \{\psi 0 1^{n^{h(n)}} \mid \psi \in \mathsf{SAT} \text{ and } n = |\psi|\}$$

[1] Ladner actually proved more - a hierarchy of intermediate problems.

NP-intermediate languages

Theorem ([Ladner 1975])

If P $\neq$ NP then there exists a language $L \in$ NP $\setminus$ P that is not NP-complete.[1]

1. For every function $h : \mathbb{N} \to \mathbb{N}$, let

$$\text{SAT}_h := \{\psi 0 1^{n^{h(n)}} \mid \psi \in \text{SAT and } n = |\psi|\}$$

2. Take $H(n)$ to be:
   - the smallest integer $i < \log \log n$ such that for every $x \in \{0,1\}^*$ with $|x| \leq \log n$,

$$M_i(x) = \text{SAT}_H(x) \text{ within } i|x|^i \text{ steps.}$$

   - else, $H(n) = \lceil \log \log n \rceil$.

[1] Ladner actually proved more - a hierarchy of intermediate problems.

NP-intermediate languages

Theorem ([Ladner 1975])

If P $\neq$ NP then there exists a language $L \in$ NP $\setminus$ P that is not NP-complete.[1]

1. For every function $h : \mathbb{N} \to \mathbb{N}$, let

$$\text{SAT}_h := \{\psi 0 1^{n^{h(n)}} \mid \psi \in \text{SAT and } n = |\psi|\}$$

2. Take $H(n)$ to be:
   - the smallest integer $i < \log\log n$ such that for every $x \in \{0,1\}^*$ with $|x| \leq \log n$,

$$M_i(x) = \text{SAT}_H(x) \text{ within } i|x|^i \text{ steps.}$$

   - else, $H(n) = \lceil \log\log n \rceil$.
3. H can be computed in $O(n^3)$ time

[1] Ladner actually proved more - a hierarchy of intermediate problems.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $\mathsf{SAT}_H \in \mathsf{P} \Rightarrow$ there is $c \in \mathbb{N}$ constant and TM M such that M computes SAT_H in time cn^c.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $\mathsf{SAT}_H \in \mathsf{P} \Rightarrow$ there is $c \in \mathbb{N}$ constant and TM M such that M computes SAT_H in time cn^c.
2. TM M represented by infinitely many strings, let $t > c$ be a constant such that $M = M_t$.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $\mathsf{SAT}_H \in \mathsf{P} \Rightarrow$ there is $c \in \mathbb{N}$ constant and TM M such that M computes SAT_H in time cn^c.
2. TM M represented by infinitely many strings, let $t > c$ be a constant such that $M = M_t$.
3. By definition of H and $M = M_t$, we have $H(n) \leq t$ for all $n > 2^{2^t}$. Thus, $H(n) = O(1)$.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $H(n) = O(1) \Rightarrow$ there is $C \in \mathbb{N}$ s.t. $H(n) \leq C$ for all $n \in \mathbb{N}$.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $H(n) = O(1) \Rightarrow$ there is $C \in \mathbb{N}$ s.t. $H(n) \leq C$ for all $n \in \mathbb{N}$.
2. Take $1 \leq c \leq C$ such that $H(n) = c$ for ∞'ly many $n \in \mathbb{N}$.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $H(n) = O(1) \Rightarrow$ there is $C \in \mathbb{N}$ s.t. $H(n) \leq C$ for all $n \in \mathbb{N}$.
2. Take $1 \leq c \leq C$ such that $H(n) = c$ for ∞'ly many $n \in \mathbb{N}$.
3. By definition of H, taking TM $M := M_c$, above implies M solves SAT_H in cn^c time.
   - Suppose not. Then there is $x \in \{0,1\}^*$ s.t. $M(x) \neq \mathsf{SAT}_H(x)$.

Helpful Claim

$$\mathsf{SAT}_H \in \mathsf{P} \Leftrightarrow H(n) = O(1).$$

1. $H(n) = O(1) \Rightarrow$ there is $C \in \mathbb{N}$ s.t. $H(n) \leq C$ for all $n \in \mathbb{N}$.
2. Take $1 \leq c \leq C$ such that $H(n) = c$ for ∞'ly many $n \in \mathbb{N}$.
3. By definition of H, taking TM $M := M_c$, above implies M solves SAT_H in cn^c time.
   - Suppose not. Then there is $x \in \{0,1\}^*$ s.t. $M(x) \neq \mathsf{SAT}_H(x)$.
   - If $n > 2^{|x|}$, then $H(n) \neq c$, since we know that x as above is s.t. $|x| \leq \log n$ and $M(x) \neq \mathsf{SAT}_H(x)$.
   - contradicts $H(n) = c$ for ∞'ly many n.

Proof of Ladner's Theorem

Assume $\mathsf{P} \neq \mathsf{NP}$. We'll show SAT_H is not in P nor NP-complete.

Proof of Ladner's Theorem

Assume P $\neq$ NP. We'll show SAT$_H$ is not in P nor NP-complete.

1. SAT$_H$ $\in$ P $\Rightarrow H = O(1) \Rightarrow$ SAT$_H$ $=$ SAT $\Rightarrow$ P $=$ NP

Proof of Ladner's Theorem

Assume P $\neq$ NP. We'll show SAT_H is not in P nor NP-complete.

1. $\mathsf{SAT}_H \in \mathsf{P} \Rightarrow H = O(1) \Rightarrow \mathsf{SAT}_H = \mathsf{SAT} \Rightarrow \mathsf{P} = \mathsf{NP}$
   - $\mathsf{SAT}_H = \mathsf{SAT}$ since the formulas are padded with a polynomial number of 1's
   - Any algorithm to solve SAT_H can be p-converted into an algorithm solving SAT (just pad first, then solve SAT_H) Here we use that $H(n)$ can be computed in time $O(n^3)$!

Proof of Ladner's Theorem

Assume $\mathsf{P} \neq \mathsf{NP}$. We'll show SAT_H is not in P nor NP-complete.

1. $\mathsf{SAT}_H \in \mathsf{P} \Rightarrow H = O(1) \Rightarrow \mathsf{SAT}_H = \mathsf{SAT} \Rightarrow \mathsf{P} = \mathsf{NP}$
2. If SAT_H is NP-complete, then there is a poly-time reduction from SAT to SAT_H.
   - Let $C \in \mathbb{N}$ be s.t. the reduction takes n^C time.

Proof of Ladner's Theorem

Assume P $\neq$ NP. We'll show SAT$_H$ is not in P nor NP-complete.

1. SAT$_H$ $\in$ P $\Rightarrow H = O(1) \Rightarrow$ SAT$_H$ = SAT $\Rightarrow$ P = NP
2. If SAT$_H$ is NP-complete, then there is a poly-time reduction from SAT to SAT$_H$.
   - Let $C \in \mathbb{N}$ be s.t. the reduction takes n^C time.
   - $\phi \in$ SAT can only be mapped to $\psi 0 1^{|\psi|^{H(|\psi|)}}$ such that

$$|\psi| + 1 + |\psi|^{H(|\psi|)} \leq |\phi|^C$$

Proof of Ladner's Theorem

Assume P $\neq$ NP. We'll show SAT$_H$ is not in P nor NP-complete.

1. SAT$_H$ $\in$ P $\Rightarrow H = O(1) \Rightarrow$ SAT$_H$ = SAT $\Rightarrow$ P = NP
2. If SAT$_H$ is NP-complete, then there is a poly-time reduction from SAT to SAT$_H$.
   - Let $C \in \mathbb{N}$ be s.t. the reduction takes n^C time.
   - $\phi \in$ SAT can only be mapped to $\psi 0 1^{|\psi|^{H(|\psi|)}}$ such that

     $$|\psi| + 1 + |\psi|^{H(|\psi|)} \leq |\phi|^C$$

   - Since P $\neq$ NP, our claim implies $H(n) = \omega(1)$ hence

     $$|\psi| \leq |\phi|^{1/2}$$

     for large enough $|\phi|$.

Proof of Ladner's Theorem

Assume P $\neq$ NP. We'll show SAT$_H$ is not in P nor NP-complete.

1. SAT$_H$ $\in$ P $\Rightarrow H = O(1) \Rightarrow$ SAT$_H$ = SAT $\Rightarrow$ P = NP
2. If SAT$_H$ is NP-complete, then there is a poly-time reduction from SAT to SAT$_H$.
   - Let $C \in \mathbb{N}$ be s.t. the reduction takes n^C time.
   - $\phi \in$ SAT can only be mapped to $\psi 0 1^{|\psi|^{H(|\psi|)}}$ such that

     $$|\psi| + 1 + |\psi|^{H(|\psi|)} \leq |\phi|^C$$

   - Since P $\neq$ NP, our claim implies $H(n) = \omega(1)$ hence

     $$|\psi| \leq |\phi|^{1/2}$$

     for large enough $|\phi|$.
   - Such reduction implies that SAT $\in$ P!

contradiction

Remarks

- Open question: are there “natural” problems in NP which are neither in P nor NP-complete?
  Candidates:
  - factoring
  - graph isomorphism

Remarks

- Open question: are there "natural" problems in NP which are neither in P nor NP-complete?
  Candidates:
  - factoring
  - graph isomorphism
- Our separations so far have all used diagonalization
- Defining "diagonalization:" any proof technique which relies only on
  1. existence of efficient representation of TMs by strings
  2. efficient simulation of TMs (universal TMs)

Remarks

- Open question: are there "natural" problems in NP which are neither in P nor NP-complete?
  Candidates:
  - factoring
  - graph isomorphism
- Our separations so far have all used diagonalization
- Defining "diagonalization:" any proof technique which relies only on
  1. existence of efficient representation of TMs by strings
  2. efficient simulation of TMs (universal TMs)
- any argument using the above treats TMs as black boxes

Could diagonalization alone prove P vs NP?

- Ladner's Theorem: NP-intermediate problems
- Oracle TMs & Relativization: limits of diagonalization

Oracle TMs & Relativization

- Given a language $O \subset \{0,1\}^*$, an oracle TM is a TM $M^O := (\Sigma, \Gamma, \mathcal{Q}, \delta)$ s.t.:
  1. M^O has a special oracle tape (in addition to other tapes)

Oracle TMs & Relativization

- Given a language $O \subset \{0,1\}^*$, an oracle TM is a TM $M^O := (\Sigma, \Gamma, \mathcal{Q}, \delta)$ s.t.:
  1. M^O has a special oracle tape (in addition to other tapes)
  2. $q_{yes}, q_{no}, q_{query} \in \mathcal{Q}$ (special states)

  When M^O enters state q_{query}, then M^O moves to q_{yes} if content of oracle tape is in O and q_{no} otherwise

  Query to O counts as 1 computational step!

Oracle TMs & Relativization

- Given a language $O \subset \{0,1\}^*$, an oracle TM is a TM $M^O := (\Sigma, \Gamma, \mathcal{Q}, \delta)$ s.t.:
  1. M^O has a special oracle tape (in addition to other tapes)
  2. $q_{yes}, q_{no}, q_{query} \in \mathcal{Q}$ (special states)

  When M^O enters state q_{query}, then M^O moves to q_{yes} if content of oracle tape is in O and q_{no} otherwise

  Query to O counts as 1 computational step!

- Similar definition for NTMs

Oracle TMs & Relativization

- Given a language $O \subset \{0,1\}^*$, an oracle TM is a TM $M^O := (\Sigma, \Gamma, \mathcal{Q}, \delta)$ s.t.:
  1. M^O has a special oracle tape (in addition to other tapes)
  2. $q_{yes}, q_{no}, q_{query} \in \mathcal{Q}$ (special states)

  When M^O enters state q_{query}, then M^O moves to q_{yes} if content of oracle tape is in O and q_{no} otherwise

  Query to O counts as 1 computational step!
- Similar definition for NTMs
- Complexity classes:
  - P^O := set of languages decided by poly-time deterministic O-oracle TMs
  - NP^O := set of languages decided by poly-time nondeterministic O-oracle TMs

  Can also define A^B where A, B are complexity classes.

Oracle TMs & Relativization

- Given a language $O \subset \{0,1\}^*$, an oracle TM is a TM $M^O := (\Sigma, \Gamma, \mathcal{Q}, \delta)$ s.t.:
  1. M^O has a special oracle tape (in addition to other tapes)
  2. $q_{yes}, q_{no}, q_{query} \in \mathcal{Q}$ (special states)

  When M^O enters state q_{query}, then M^O moves to q_{yes} if content of oracle tape is in O and q_{no} otherwise

  Query to O counts as 1 computational step!
- Similar definition for NTMs
- Complexity classes:
  - P^O := set of languages decided by poly-time deterministic O-oracle TMs
  - NP^O := set of languages decided by poly-time nondeterministic O-oracle TMs

  Can also define A^B where A, B are complexity classes.
- Oracle TMs satisfy diagonalization properties! (relativize)

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.
2. if $O \in \mathsf{P}$ then $\mathsf{P}^O = \mathsf{P}$

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.
2. if $O \in \mathsf{P}$ then $\mathsf{P}^O = \mathsf{P}$
3. $EXPCOM := \{\langle M, x, 1^n \rangle \mid M(x) = 1 \text{ within } 2^n \text{ steps}\}$

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.
2. if $O \in \mathsf{P}$ then $\mathsf{P}^O = \mathsf{P}$
3. $EXPCOM := \{\langle M, x, 1^n\rangle \mid M(x) = 1 \text{ within } 2^n \text{ steps}\}$

Claim

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP} := \bigcup_{c\in\mathbb{N}} \mathsf{DTIME}(2^{n^c})$$

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.
2. if $O \in \mathsf{P}$ then $\mathsf{P}^{O} = \mathsf{P}$
3. $EXPCOM := \{\langle M, x, 1^n\rangle \mid M(x) = 1 \text{ within } 2^n \text{ steps}\}$

Claim

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP} := \bigcup_{c \in \mathbb{N}} \mathsf{DTIME}(2^{n^c})$$

- $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$ since P^{EXPCOM} can perform exponential-time computation in one step

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.
2. if $O \in \mathsf{P}$ then $\mathsf{P}^{O} = \mathsf{P}$
3. $EXPCOM := \{\langle M, x, 1^n \rangle \mid M(x) = 1 \text{ within } 2^n \text{ steps}\}$

Claim

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP} := \bigcup_{c \in \mathbb{N}} \mathsf{DTIME}(2^{n^c})$$

- $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$ since P^{EXPCOM} can perform exponential-time computation in one step
- $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$ since can simulate every $M^{EXPCOM} \in \mathsf{NP}^{EXPCOM}$ in exponential-time

Examples

1. $\overline{\mathsf{SAT}}$ class of unsatisfiable formulae, then $\overline{\mathsf{SAT}} \in \mathsf{P}^{\mathsf{SAT}}$.
2. if $O \in \mathsf{P}$ then $\mathsf{P}^O = \mathsf{P}$
3. $EXPCOM := \{\langle M, x, 1^n \rangle \mid M(x) = 1 \text{ within } 2^n \text{ steps}\}$

Claim

$$\mathsf{P}^{EXPCOM} = \mathsf{NP}^{EXPCOM} = \mathsf{EXP} := \bigcup_{c \in \mathbb{N}} \mathsf{DTIME}(2^{n^c})$$

- $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM}$ since P^{EXPCOM} can perform exponential-time computation in one step
- $\mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$ since can simulate every $M^{EXPCOM} \in \mathsf{NP}^{EXPCOM}$ in exponential-time
- $\mathsf{EXP} \subseteq \mathsf{P}^{EXPCOM} \subseteq \mathsf{NP}^{EXPCOM} \subseteq \mathsf{EXP}$.

Baker-Gill-Solovay

Theorem ([Baker Gill Solovay, 1975])

There exist oracles A, B such that

$$\mathsf{P}^A = \mathsf{NP}^A \quad \textit{and} \quad \mathsf{P}^B \neq \mathsf{NP}^B.$$

Baker-Gill-Solovay

Theorem ([Baker Gill Solovay, 1975])

There exist oracles A, B such that

$$\mathsf{P}^A = \mathsf{NP}^A \text{ and } \mathsf{P}^B \neq \mathsf{NP}^B.$$

Corollary

No proof for P vs NP can relativize!

Baker-Gill-Solovay

Theorem ([Baker Gill Solovay, 1975])

There exist oracles A, B such that

$$\mathsf{P}^A = \mathsf{NP}^A \quad \text{and} \quad \mathsf{P}^B \neq \mathsf{NP}^B.$$

Take $A = EXPCOM$.

Baker-Gill-Solovay

Theorem ([Baker Gill Solovay, 1975])
There exist oracles A, B such that

$$\mathsf{P}^A = \mathsf{NP}^A \ \text{ and } \ \mathsf{P}^B \neq \mathsf{NP}^B.$$

1. For any language $B \subset \{0,1\}^*$, let

$$U_B := \{1^n \mid B \text{ contains some string of length } n\}$$

Baker-Gill-Solovay

Theorem ([Baker Gill Solovay, 1975])

There exist oracles A, B such that

$$\mathsf{P}^A = \mathsf{NP}^A \text{ and } \mathsf{P}^B \neq \mathsf{NP}^B.$$

1. For any language $B \subset \{0,1\}^*$, let

$$U_B := \{1^n \mid B \text{ contains some string of length } n\}$$

2. For any B, $U_B \in \mathsf{NP}^B$
   Guess string $x \in \{0,1\}^n$ and ask oracle

Baker-Gill-Solovay

Theorem ([Baker Gill Solovay, 1975])

There exist oracles A, B such that

$$P^A = NP^A \quad \text{and} \quad P^B \neq NP^B.$$

1. For any language $B \subset \{0,1\}^*$, let

$$U_B := \{1^n \mid B \text{ contains some string of length } n\}$$

2. For any B, $U_B \in \mathsf{NP}^B$
3. We'll construct B such that $U_B \notin \mathsf{P}^B$

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
  Construct B in stages: $B = \bigcup_{i \in \mathbb{N}} B_i$.
  Each B_i is a finite set.

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$
  2. suppose we have handled TMs $M_0^B, \ldots, M_{i-1}^B$
     - Since $B = B_0 \cup \cdots \cup B_{i-1}$ is finite, choose n_i larger than length of any string in B

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$
  2. suppose we have handled TMs $M_0^B, \ldots, M_{i-1}^B$
     - Since $B = B_0 \cup \cdots \cup B_{i-1}$ is finite, choose n_i larger than length of any string in B

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$
  2. suppose we have handled TMs $M_0^B, \ldots, M_{i-1}^B$
     - Since $B = B_0 \cup \cdots \cup B_{i-1}$ is finite, choose n_i larger than length of any string in B
     - Run $M_i^B(1^{n_i})$ for $2^{n_i}/10$ steps
       M_i^B deterministic, so we know all queries to B from $M_i^B(1^{n_i})$!

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$
  2. suppose we have handled TMs $M_0^B, \ldots, M_{i-1}^B$
     - Since $B = B_0 \cup \cdots \cup B_{i-1}$ is finite, choose n_i larger than length of any string in B
     - Run $M_i^B(1^{n_i})$ for $2^{n_i}/10$ steps
     - whenever $M_i^B(1^{n_i})$ queries previously determined strings, answer consistently. Else, answer NO.

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$
  2. suppose we have handled TMs $M_0^B, \ldots, M_{i-1}^B$
     - Since $B = B_0 \cup \cdots \cup B_{i-1}$ is finite, choose n_i larger than length of any string in B
     - Run $M_i^B(1^{n_i})$ for $2^{n_i}/10$ steps
     - whenever $M_i^B(1^{n_i})$ queries previously determined strings, answer consistently. Else, answer NO.
     - Have determined if $x \in^? B \cap \{0,1\}^{n_i}$ for at most $2^{n_i}/10$ strings! (and all of them NO answer!)

       If $M_i^B(1^{n_i}) = 1$ then declare $\{0,1\}^{n_i} \cap B = \emptyset$. Else, add a non-queried string from $\{0,1\}^{n_i}$ to B.

Constructing B

- Idea: diagonalization! :D
  Property: for every i, M_i^B does not decide U_B in time $2^n/10$.
- Induction:
  1. Start with $B = \emptyset$
  2. suppose we have handled TMs $M_0^B, \ldots, M_{i-1}^B$
     - Since $B = B_0 \cup \cdots \cup B_{i-1}$ is finite, choose n_i larger than length of any string in B
     - Run $M_i^B(1^{n_i})$ for $2^{n_i}/10$ steps
     - whenever $M_i^B(1^{n_i})$ queries previously determined strings, answer consistently. Else, answer NO.
     - Have determined if $x \in^? B \cap \{0,1\}^{n_i}$ for at most $2^{n_i}/10$ strings! (and all of them NO answer!)

       If $M_i^B(1^{n_i}) = 1$ then declare $\{0,1\}^{n_i} \cap B = \emptyset$. Else, add a non-queried string from $\{0,1\}^{n_i}$ to B.
  3. Thus, for any $f(n) = o(2^n)$, as any TM has ∞'ly many string representations, above construction implies

$$U_B \notin \mathsf{DTIME}^B(f(n)) \forall f(n) = o(2^n) \Rightarrow U_B \notin \mathsf{P}^B$$

Conclusion

- Learned powers and limitations of diagonalization

Conclusion

- Learned powers and limitations of diagonalization
- Oracles: abstraction of “subroutines as black-boxes”

Conclusion

- Learned powers and limitations of diagonalization
- Oracles: abstraction of “subroutines as black-boxes”
- Can diagonalization be used to solve P vs NP?

Could be, but proof has to differentiate between oracle-machines and non-oracle machines.

I.e., non-relativize.

Conclusion

- Learned powers and limitations of diagonalization
- Oracles: abstraction of "subroutines as black-boxes"
- Can diagonalization be used to solve P vs NP?

  Could be, but proof has to differentiate between oracle-machines and non-oracle machines.

  I.e., non-relativize.

- Connection to mathematical logic:

  Independence results: certain mathematical statements cannot be proved or disproved in a particular set of axioms.

  1. Independence of Euclid's fifth postulate (non-Euclidean geometries)
  2. Continuum Hypothesis from Zermelo-Fraenkel

References I

Arora, Sanjeev and Barak, Boaz (2009)
Computational Complexity, A Modern Approach
Cambridge University Press

Baker, Theodore and Gill, John and Solovay, Robert (1975)
Relativizations of the P $=?$NP question.
SIAM Journal on computing

Ladner, Richard E. (1975)
On the structure of polynomial time reducibility.
Journal of the ACM (JACM)