

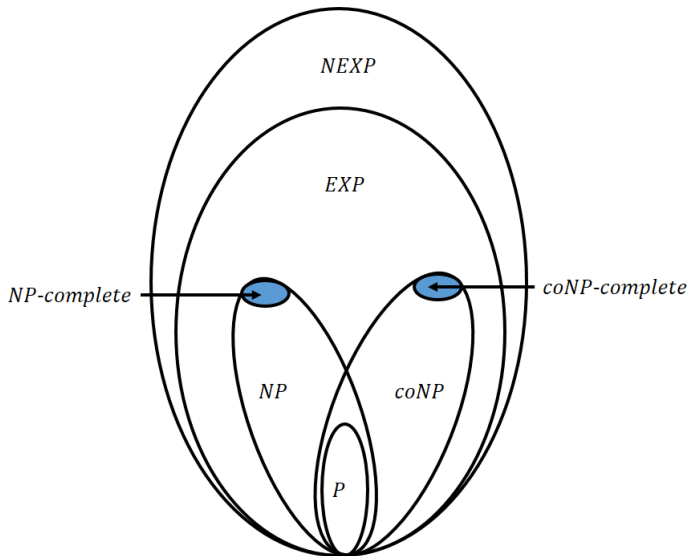
CSL853: Complexity Theory

Ragesh Jaiswal, CSE, IIT Delhi

Complexity Diagram

Complexity Diagram

Our current view of complexity classes



Diagonalization: NP-Intermediate Problems

Theorem (NP-Intermediate Languages, Ladner, 1975)

Suppose that $P \neq NP$. Then there exists a language $L \in NP \setminus P$ that is not NP-complete.

Proof (based on unpublished proof by Impagliazzo).

- For any $H : \mathbb{N} \rightarrow \mathbb{N}$, define
$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} \mid \psi \in \text{SAT} \text{ and } n = |\psi|\}.$$
- Define $H : \mathbb{N} \rightarrow \mathbb{N}$ in the following manner:
 - $H(n)$ is the smallest number $i < \log \log n$ such that for every $x \in \{0,1\}^*$ with $|x| \leq \log n$, M_i outputs $\text{SAT}_H(x)$ within $i \cdot |x|^i$ steps. If there is no such number i , then $H(n) = \log \log n$.

Diagonalization

NP-Intermediate Problems

Theorem (NP-Intermediate Languages, Ladner, 1975)

Suppose that $P \neq NP$. Then there exists a language $L \in NP \setminus P$ that is not NP-complete.

Proof (based on unpublished proof by Impagliazzo).

- For any $H : \mathbb{N} \rightarrow \mathbb{N}$, define
$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} \mid \psi \in \text{SAT} \text{ and } n = |\psi|\}.$$
- Define $H : \mathbb{N} \rightarrow \mathbb{N}$ in the following manner:
 - $H(n)$ is the smallest number $i < \log \log n$ such that for every $x \in \{0, 1\}^*$ with $|x| \leq \log n$, M_i outputs $\text{SAT}_H(x)$ within $i \cdot |x|^i$ steps. If there is no such number i , then $H(n) = \log \log n$.
- Claim 1: $\text{SAT}_H \in P$ if and only if $H(n) = O(1)$. Moreover, if $\text{SAT}_H \notin P$, then $H(n)$ tends to infinity with n .

Diagonalization

NP-Intermediate Problems

Theorem (NP-Intermediate Languages, Ladner, 1975)

Suppose that $P \neq NP$. Then there exists a language $L \in NP \setminus P$ that is not NP-complete.

Proof (based on unpublished proof by Impagliazzo).

- For any $H : \mathbb{N} \rightarrow \mathbb{N}$, define $SAT_H = \{\psi 01^{n^{H(n)}} \mid \psi \in SAT \text{ and } n = |\psi|\}$.
- Define $H : \mathbb{N} \rightarrow \mathbb{N}$ in the following manner:
 - $H(n)$ is the smallest number $i < \log \log n$ such that for every $x \in \{0, 1\}^*$ with $|x| \leq \log n$, M_i outputs $SAT_H(x)$ within $i \cdot |x|^i$ steps. If there is no such number i , then $H(n) = \log \log n$.
- Claim 1: $SAT_H \in P$ if and only if $H(n) = O(1)$. Moreover, if $SAT_H \notin P$, then $H(n)$ tends to infinity with n .
- Claim 2: If $P \neq NP$, then SAT_H is neither in P nor NP-complete.

Diagonalization

NP-Intermediate Problems

Theorem (NP-Intermediate Languages, Ladner, 1975)

Suppose that $P \neq NP$. Then there exists a language $L \in NP \setminus P$ that is not NP-complete.

Proof (based on unpublished proof by Impagliazzo).

- For any $H : \mathbb{N} \rightarrow \mathbb{N}$, define $SAT_H = \{\psi 01^{n^{H(n)}} \mid \psi \in SAT \text{ and } n = |\psi|\}$.
- Define $H : \mathbb{N} \rightarrow \mathbb{N}$ in the following manner:
 - $H(n)$ is the smallest number $i < \log \log n$ such that for every $x \in \{0,1\}^*$ with $|x| \leq \log n$, M_i outputs $SAT_H(x)$ within $i \cdot |x|^i$ steps. If there is no such number i , then $H(n) = \log \log n$.
- Claim 1: $SAT_H \in P$ if and only if $H(n) = O(1)$. Moreover, if $SAT_H \notin P$, then $H(n)$ tends to infinity with n .
- Claim 2: If $P \neq NP$, then SAT_H is neither in P nor NP-complete.
 - Case 1: $SAT_H \in P \Rightarrow H(n) \leq C \Rightarrow SAT \in P \perp$.
 - Case 2: SAT_H in NP-complete $\Rightarrow SAT$ formulas can be “shrunk” $\Rightarrow SAT \in P \perp$.



Diagonalization: Limitations

Diagonalization

Limitations

- All the diagonalization based results that we saw, relied only on the following two properties of Turing Machines:
 - ① Turing machines can be represented using binary strings.
 - ② One TM can simulate another TM without much overhead in running time or space.
- Any argument that uses just the above two properties treat TMs as *black boxes* and do not utilise the internal workings of TMs.
- We will show that such arguments cannot resolve $P \neq NP$ question.

Diagonalization

Limitations

- All the diagonalization based results that we saw, relied only on the following two properties of Turing Machines:
 - ① Turing machines can be represented using binary strings.
 - ② One TM can simulate another TM without much overhead in running time or space.
- Any argument that uses just the above two properties treat TMs as *black boxes* and do not utilise the internal workings of TMs.
- We will show that such arguments cannot resolve $P \neq NP$ question.
 - Vague Plan: Any such argument will also hold for *Oracle Turing Machines*. But then, there is an *oracle* with respect to which $P = NP$ and another oracle with respect to which $P \neq NP$.

Definition (Oracle Turing Machines)

An *oracle* Turing machine is a TM M that has a special read/write tape we call M 's oracle tape and three special states

q_{query} , q_{yes} , q_{no} . To execute M , we specify in addition to the input a language $O \subseteq \{0,1\}^*$ that is used as the oracle for M . Whenever during the execution M enters the state q_{query} , the machine moves into the state q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes the contents of the special oracle tape.

- Membership query to O counts as a single computation step.
- If M is an oracle machine, $O \subseteq \{0,1\}^*$ a language, and $x \in \{0,1\}^*$, then we denote the output of M on input x and with oracle O by $M^O(x)$.

Diagonalization

Limitations

Definition (Oracle Turing Machines)

An *oracle* Turing machine is a TM M that has a special read/write tape we call M 's oracle tape and three special states

q_{query} , q_{yes} , q_{no} . To execute M , we specify in addition to the input a language $O \subseteq \{0, 1\}^*$ that is used as the oracle for M . Whenever during the execution M enters the state q_{query} , the machine moves into the state q_{yes} if $q \in O$ and q_{no} if $q \notin O$, where q denotes the contents of the special oracle tape.

Definition

For every $O \subseteq \{0, 1\}^*$, P^O is the set of languages decided by a polynomial-time deterministic TM with oracle access to O and NP^O is the set of languages decided by a polynomial-time non-deterministic TM with oracle access to O .

Diagonalization

Limitations

Definition

For every $O \subseteq \{0,1\}^*$, P^O is the set of languages decided by a polynomial-time deterministic TM with oracle access to O . and NP^O is the set of languages decided by a polynomial-time non-deterministic TM with oracle access to O .

- Show that $\overline{\text{SAT}} \in P^{\text{SAT}}$.
- Show that for any $O \in P$, $P^O = P$.

Diagonalization

Limitations

- For any oracle O , the set of all TMs with access to O satisfy the two properties:
 - ① Turing machines can be represented using binary strings.
 - ② One TM can simulate another TM without much overhead in running time or space.
- Any result about TMs that uses only the above two properties also holds for TMs with access to O . Such results are called *relativizing*.

Theorem (Baker, Gill, Solovay, 1975)

There exists oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.

Diagonalization

Limitations: [BGS75]

Theorem (Baker, Gill, Solovay, 1975)

There exists oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.

Proof.

- Let $A = \{\langle M, x, 1^n \rangle \mid M \text{ outputs } 1 \text{ on } x \text{ within } 2^n \text{ steps}\}$.
- Claim 1: $P^A = NP^A$.

Diagonalization

Limitations: [BGS75]

Theorem (Baker, Gill, Solovay, 1975)

There exists oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.

Proof.

- Let $A = \{\langle M, x, 1^n \rangle \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps}\}$.
- Claim 1: $P^A = NP^A$.
- For any language $L \subseteq \{0, 1\}^*$, let $U_L = \{1^n \mid \text{some string of length } n \text{ is in } L\}$.
- Claim 2: There is a language B such that $U_B \in NP^B$ and $U_B \notin P^B$.

Diagonalization

Limitations: [BGS75]

Theorem (Baker, Gill, Solovay, 1975)

There exists oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.

Proof.

- Let $A = \{\langle M, x, 1^n \rangle \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps}\}$.
- Claim 1: $P^A = NP^A$.
- For any language $L \subseteq \{0, 1\}^*$, let $U_L = \{1^n \mid \text{some string of length } n \text{ is in } L\}$.
- Claim 2: There is a language B such that $U_B \in NP^B$ and $U_B \notin P^B$.
 - For any language B , $U_B \in NP^B$ since the NDTM can guess the string and make a single oracle query to B to determine the membership of U_B .
 - We will construct a B such that $U_B \notin P^B$.

Diagonalization

Limitations: [BGS75]

Theorem (Baker, Gill, Solovay, 1975)

There exists oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.

Proof.

- Let $A = \{\langle M, x, 1^n \rangle \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps}\}$.
- Claim 1: $P^A = NP^A$.
- For any language $L \subseteq \{0, 1\}^*$, let $U_L = \{1^n \mid \text{some string of length } n \text{ is in } L\}$.
- Claim 2: There is a language B such that $U_B \in NP^B$ and $U_B \notin P^B$.
 - For any language B , $U_B \in NP^B$ since the NDTM can guess the string and make a single oracle query to B to determine the membership of U_B .
 - We will construct a B such that $U_B \notin P^B$.
 - We construct B in stages. In stage i , we will carefully put/remove strings in B that rules out machine M_i^B .

Diagonalization

Limitations: [BGS75]

Theorem (Baker, Gill, Solovay, 1975)

There exists oracles A, B such that $P^A = NP^A$ and $P^B \neq NP^B$.

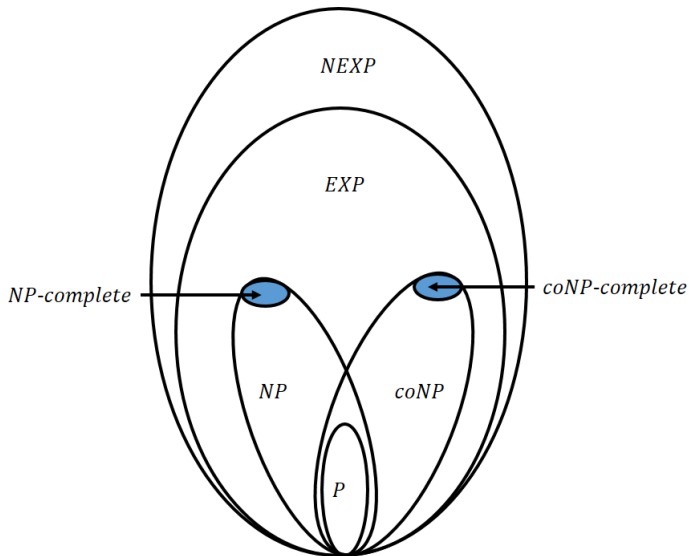
Proof.

- Let $A = \{\langle M, x, 1^n \rangle \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps}\}$.
- Claim 1: $P^A = NP^A$.
- For any language $L \subseteq \{0, 1\}^*$, let $U_L = \{1^n \mid \text{some string of length } n \text{ is in } L\}$.
- Claim 2: There is a language B such that $U_B \in NP^B$ and $U_B \notin P^B$.
 - For any language B , $U_B \in NP^B$ since the NDTM can guess the string and make a single oracle query to B to determine the membership of U_B .
 - We will construct a B such that $U_B \notin P^B$.
 - Stage i : Let n be such that no string of length n has been queried by $M_1^B, M_2^B, \dots, M_{i-1}^B$. Run M_i on input 1^n for $2^{n/10}$ steps. Whenever M_i queries any string for which the inclusion is known it is returned the appropriate answer. Whenever M_i queries any string whose status is undetermined, we declare that the string is not in B .
 - Claim 2.1 There is a way to make sure that M_i outputs the incorrect answer on finishing its computation.

Complexity Diagram

Complexity Diagram

Our current view of complexity classes



End