Oracle Turing Machines

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

Seminar Presentation

Lake Superior State University

Oracle Turing Machines

- An **oracle Turing machine** is a Turing machine that has a special read-write tape, called **oracle tape** and three special states q_{query}, q_{yes} and q_{no} (not to be confused with q_{accept} and q_{reject}).
- The operation of M, in addition to the input language, needs a specification of a language O, the **oracle language**.
- Whenever, during its execution, M enters the state q_{query}, with q the contents of the oracle tape, then the machine moves into the state
  - q_{yes} if $q \in \mathrm{O}$;
  - q_{no} if $q \notin \mathrm{O}$.
- Regardless of the choice of O, a membership query to O counts as a single computational step.
- If M is an oracle machine, $\mathrm{O} \subseteq \{0,1\}^*$ a language and $x \in \{0,1\}^*$, then the output of M on input x with oracle O is denoted $M^{\mathrm{O}}(x)$.
- **Nondeterministic oracle Turing machines** are defined similarly.

The Classes $\mathbf{P}^{\mathrm{O}}$ and $\mathbf{NP}^{\mathrm{O}}$

- For every $\mathrm{O} \subseteq \{0,1\}^*$, $\mathbf{P}^{\mathrm{O}}$ is the class of all languages that can be decided by a polynomial time deterministic Turing machine with oracle access to O.
- $\mathbf{NP}^{\mathrm{O}}$ is the class of all languages that can be decided by a polynomial time nondeterministic Turing machine with oracle access to O.

Examples

- Suppose $\overline{\text{SAT}}$ is the language of all unsatisfiable Boolean formulæ.

  Then $\overline{\text{SAT}} \in \mathbf{P}^{\text{SAT}}$.
- Let $\mathrm{O} \in \mathbf{P}$. Then $\mathbf{P}^{\mathrm{O}} = \mathbf{P}$.

  Allowing an oracle may only help decide more languages.

  Hence, $\mathbf{P} \subseteq \mathbf{P}^{\mathrm{O}}$.

  Suppose that $\mathrm{L} \in \mathbf{P}^{\mathrm{O}}$.

  Consider the polynomial time Turing machine M with oracle O that computes L.

  Transform it into a machine that operates like M except that, instead of querying the oracle, decides membership in O from scratch (in polynomial time).

  This is a polynomial time deterministic Turing machine deciding L.

  Thus, $\mathrm{L} \in \mathbf{P}$ and $\mathbf{P}^{\mathrm{O}} \subseteq \mathbf{P}$.

The Language $\textsc{ExpCom}$ of Exponential Computation

- Consider the language

$$\textsc{ExpCom} = \{\langle M, x, 1^n \rangle : M \text{ accepts } x \text{ within } 2^n \text{ steps}\}.$$

Then $\mathbf{P}^{\textsc{ExpCom}} = \mathbf{NP}^{\textsc{ExpCom}} = \mathbf{EXP}$ $(:= \bigcup_c \mathbf{DTIME}(2^{n^c}))$.

Using $\textsc{ExpCom}$ as an oracle allows performing exponential computations in a single step. So $\mathbf{EXP} \subseteq \mathbf{P}^{\textsc{ExpCom}}$.

Suppose M is a nondeterministic polynomial-time oracle Turing machine.

Exponential time is sufficient to:

  - enumerate all M's nondeterministic choices;
  - answer all of $\textsc{ExpCom}$'s oracle queries.

Therefore, $\mathbf{NP}^{\textsc{ExpCom}} \subseteq \mathbf{EXP}$.

Diagonalization and Relativization

- Several results in complexity separating classes rely on the method of **"pure" diagonalization**, a technique that relies solely on the following properties of Turing machines:
  - I The existence of an effective representation of Turing machines by strings;
  - II The ability of one Turing machine to simulate another without much overhead in running time or space.
- For any choice of oracle O, the set of all Turing machines with access to O satisfies properties I and II.
  - Turing machines with oracle O can be represented as strings;
  - The representation can be used to simulate such Turing machines by a universal Turing machine (having itself access to oracle O).
- It follows that any result about Turing machines or complexity classes that uses only I and II **relativizes**, i.e., holds also for the set of all Turing machines with oracle O.

The Baker, Gill, Solovay Theorem

The Baker, Gill, Solovay Theorem

There exist oracle languages A and B, such that $\mathbf{P}^{\mathrm{A}} = \mathbf{NP}^{\mathrm{A}}$ and $\mathbf{P}^{\mathrm{B}} \neq \mathbf{NP}^{\mathrm{B}}$.

- Let $\mathrm{A} = \textsc{ExpCom}$. We saw that $\mathbf{P}^{\mathrm{A}} = \mathbf{NP}^{\mathrm{A}}$.
- Let B be any language. Define

$$\mathrm{U_B} = \{1^n : (\exists y \in \mathrm{B})(|y| = n)\}.$$

$\mathrm{U_B} \in \mathbf{NP}^{\mathrm{B}}$. The following polynomial time nondeterministic Turing machine with oracle B decides $\mathrm{U_B}$.

On input x:
Check (in linear time) whether $x = 1^{|x|}$; If not, reject;
Guess in linear time $y \in \{0,1\}^{|x|}$;
Query oracle whether $y \in \mathrm{B}$;
If yes, accept; else reject.

The heart of the argument is to construct B, such that $\mathrm{U_B} \notin \mathbf{P}^{\mathrm{B}}$.

Stage-Wise Construction of B

- For all i, let M_i be the oracle TM represented by i in binary.
  B is constructed in stages, where Stage i ensures that M_i^{B} does not decide $\mathrm{U_B}$ within $\frac{2^n}{10}$ steps (n depends on i).
  Initialize $\mathrm{B} = \emptyset$;
  Stage i: Assume "$\in \mathrm{B}$?" has been decided for finitely many strings.
  Choose n exceeding the length of all such strings.
  Run M_i on 1^n for $\frac{2^n}{10}$ steps.
    - If M_i queries the oracle on a decided string, answer consistently;
    - Otherwise, declare that the string $\notin \mathrm{B}$.

  We have decided the fate of $\leq \frac{2^n}{10}$ strings of length n, all declared $\notin \mathrm{B}$.
    - If M_i accepts 1^n, all remaining $\frac{9 \cdot 2^n}{10}$ strings of length n are declared $\notin \mathrm{B}$. So $1^n \notin \mathrm{U_B}$.
    - If M_i rejects 1^n, pick a string x of length n not queried upon and declare $x \in \mathrm{B}$. So $1^n \in \mathrm{U_B}$.

  We made sure that M_i does not decide $\mathrm{U_B}$.
  Since every polynomial is smaller than $\frac{2^n}{10}$ for large n and every Turing machine is represented by infinitely many strings, $\mathrm{U_B} \notin \mathbf{P}^{\mathrm{B}}$.

Significance for $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

- We saw that "pure" diagonalization relativizes.
- Since there are oracles A and B, relative to which $\mathbf{P}^{\mathrm{A}} = \mathbf{NP}^{\mathrm{A}}$ and $\mathbf{P}^{\mathrm{B}} \neq \mathbf{NP}^{\mathrm{B}}$, "pure" diagonalization alone cannot resolve $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.
- It is still possible that diagonalization, or a technique involving simulation, may be used to tackle $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$, but it has to use some fact about Turing machines that does not hold in the presence or oracles, i.e., that does not relativize.

  That is, some property different from I and II must be added in the mix.

Oracle Classes: Warm-Up

- For a class $\mathcal{C}$ of languages, we set

$$\mathbf{P}^{\mathcal{C}} = \bigcup_{\mathrm{O} \in \mathcal{C}} \mathbf{P}^{\mathrm{O}} \quad \text{and} \quad \mathbf{NP}^{\mathcal{C}} = \bigcup_{\mathrm{O} \in \mathcal{C}} \mathbf{NP}^{\mathrm{O}}.$$

- We obviously have

$$\mathbf{NP} \subseteq \mathbf{P}^{\mathbf{NP}} \quad \text{and} \quad \mathbf{co\text{-}NP} \subseteq \mathbf{P}^{\mathbf{NP}}.$$

- It is likely that

$$\mathbf{NP} \cup \mathbf{co\text{-}NP} \subsetneq \mathbf{P}^{\mathbf{NP}}.$$

  However, if $\mathbf{NP} = \mathbf{P}$, then $\mathbf{P}^{\mathbf{NP}} = \mathbf{P}$ and all three classes above would be identical.

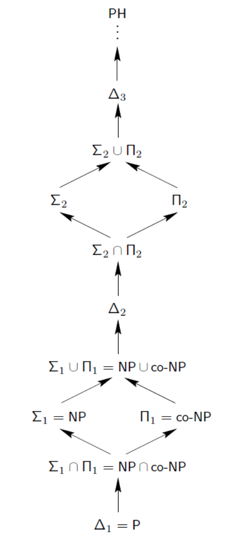
The Polynomial Hierarchy via Oracles

- Let $\Sigma_1 := \mathbf{NP}$, $\Pi_1 := \mathbf{co}\text{-}\mathbf{NP}$, and $\Delta_1 := \mathbf{P}$.
- For $k \geq 1$, let
  - $\Sigma_{k+1} := \mathbf{NP}^{\Sigma_k}$;
  - $\Pi_{k+1} := \mathbf{co}\text{-}\Sigma_{k+1}$;
  - $\Delta_{k+1} := \mathbf{P}^{\Sigma_k}$.
- The polynomial hierarchy **PH** is the union

$$\mathbf{PH} = \bigcup_{k \geq 1} \Sigma_k.$$

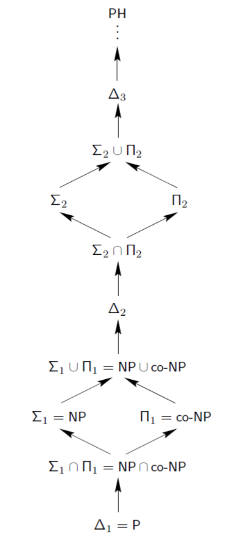
- It is also consistent to let $\Sigma_0 = \Pi_0 = \Delta_0 = \mathbf{P}$, and to extend the definition to all $k \geq 0$.

  Indeed we have, $\Sigma_1 = \mathbf{NP}$, $\Pi_1 = \mathbf{co}\text{-}\mathbf{NP}$ and $\Delta_1 = \mathbf{P}$.



Complexity Theoretic Hypotheses

- The conjecture that the classes of the polynomial hierarchy form a genuine hierarchy contains the conjecture that:
  - all the inclusions are strict inclusions;
  - the classes Σ_k and Π_k are incomparable with respect to set inclusion.
- Thus we obtain the following complexity theoretical hypotheses:
  - $\Sigma_k \neq \Sigma_{k+1}$;
  - $\Pi_k \neq \Pi_{k+1}$;
  - $\Sigma_k \neq \Pi_k$;
  - $\Delta_k \neq \Sigma_k \cap \Pi_k \neq \Sigma_k \neq \Sigma_k \cup \Pi_k \neq \Delta_{k+1}$.



Logical Characterizations

Theorem

A decision problem L belongs to the class Σ_k if and only if there is a poly p and a decision problem $\mathrm{L}' \in \mathbf{P}$, such that for $A = \{0,1\}^{p(|x|)}$,

$$\mathrm{L} = \{x : (\exists y_1 \in A)(\forall y_2 \in A)(\exists y_3 \in A) \cdots (Qy_k \in A)(x, y_1, \ldots, y_k) \in \mathrm{L}'\}.$$

The quantifier Q is chosen to be an existential or universal quantifier in such a way that the sequence of quantifiers is alternating.

- Using DeMorgan's Laws we obtain:

Corollary

A decision problem L is in Π_k if and only if there is a polynomial p and a decision problem $\mathrm{L}' \in \mathbf{P}$, such that for $A = \{0,1\}^{p(|x|)}$, then

$$\mathrm{L} = \{x : (\forall y_1 \in A)(\exists y_2 \in A) \cdots (Qy_k \in A)(x, y_1, \ldots, y_k) \in \mathrm{L}'\}.$$

Horizontal Collapsibility

Theorem

If $\Sigma_k = \Pi_k$, then $\mathbf{PH} = \Sigma_k$.

- We show that $\Sigma_k = \Pi_k$ implies $\Sigma_{k+1} = \Pi_{k+1} = \Sigma_k$.

  The argument can be completed using induction on k.

  Let's look at the case $k = 4$. From the logical characterizations, $\Sigma_4 = \Pi_4$, means that $\exists\forall\exists\forall\mathbf{P} = \forall\exists\forall\exists\mathbf{P}$, where:
  - Behind the quantifiers we may only have polynomially many variables;
  - $\mathbf{P}$ stands for decision problems from $\mathbf{P}$, which may be different on the two sides of the equation.

  Now we consider Σ_5, i.e., a problem of the form $\exists(\forall\exists\forall\exists\mathbf{P})$.

  By hypothesis, this is of form $\exists\exists\forall\exists\forall\mathbf{P}$. But two quantifiers of the same type can be brought together as a single quantifier.

  So every Σ_5-problem is of the form $\exists\forall\exists\forall\mathbf{P}$ and so belongs to Σ_4.

  It follows that $\Sigma_5 = \Sigma_4 = \Pi_4$. Similarly, we get $\Pi_5 = \Pi_4 = \Sigma_4$.

Vertical Collapsibility

Corollary

If $\Sigma_k = \Sigma_{k+1}$, then $\mathbf{PH} = \Sigma_k$.

- We know that $\Sigma_k \subseteq \Pi_{k+1}$.

  From $\Sigma_k = \Sigma_{k+1}$, we get $\Sigma_{k+1} \subseteq \Pi_{k+1}$.

  But, by definition, $\Pi_{k+1} := \mathbf{co}\text{-}\Sigma_{k+1}$, whence, $\Sigma_{k+1} = \Pi_{k+1}$.

  The Theorem implies that $\mathbf{PH} = \Sigma_{k+1}$.

  By hypothesis, $\mathbf{PH} = \Sigma_k$.

Thank you!

- In closing...

Thank you for your Attention!!