Polynomial Time With Oracles
How to Think about Oracles?

Think in terms of Turing Machine pseudocode or a subroutine

An oracle Turing machine $M$ with oracle $B \subseteq \Gamma^*$ lets you include the following kind of branching instructions:

“if ($z$ in $B$) then <do something> else <do something else>”

where $z$ is some string defined earlier in pseudocode.

By definition, the oracle TM can always check the condition ($z$ in $B$) in one step.
Some Complexity Classes With Oracles

\[ P^B = \{ L \mid L \text{ can be decided by some polynomial-time TM with an oracle for } B \} \]

\[ P^{\text{SAT}} = \text{the class of languages decidable in polynomial time with an oracle for SAT} \]

\[ P^{\text{NP}} = \text{the class of languages decidable by some polynomial-time oracle TM with an oracle for some } B \text{ in NP} \]
Is $P^{SAT} \subseteq P^{NP}$?

Yes. By definition...

Is $P^{NP} \subseteq P^{SAT}$?

Yes:

Every NP language can be reduced to SAT!

For every poly-time TM $M$ with oracle $B \in NP$, we can simulate every query $z$ to oracle $B$ by reducing $z$ to a formula $\phi$ in poly-time, then asking an oracle for SAT instead.
For every poly-time TM $M$ with oracle $B \in P$, we can simulate every query $z$ to oracle $B$ by simply running a polynomial-time decider for $B$.

$P_B = \{ L \mid L$ can be decided by a polynomial-time TM with an oracle for $B \}$

Suppose $B$ is in $P$.

Is $P_B \subseteq P$?

Yes

For every poly-time TM $M$ with oracle $B \in P$, we can simulate every query $z$ to oracle $B$ by simply running a polynomial-time decider for $B$.

The resulting machine runs in polynomial time
Is $\text{NP} \subseteq \text{P}^{\text{NP}}$?

Yes

*Just ask the oracle for the answer!*

For every $L \in \text{NP}$ define an oracle TM $M^L$ which asks the oracle if the input is in $L$. 
Is $\text{coNP} \subseteq \text{P}^{\text{NP}}$?

Yes!

Again, just ask the oracle for the answer!

For every $L \in \text{coNP}$ we know $\neg L \in \text{NP}$

Define an oracle TM $M^{\neg L}$ which asks the oracle if the input is in $\neg L$

*accept* if the answer is no,

*reject* if the answer is yes

More generally, we have $\text{P}^{\text{NP}} = \text{P}^{\text{coNP}}$
NP^B = \{ L \mid L \text{ can be decided by a polynomial-time nondeterministic TM with an oracle for } B \} \\
coNP^B = \{ L \mid L \text{ can be decided by a poly-time co-nondeterministic TM with an oracle for } B \} \\
Is \, NP = NP^{NP}?

Is coNP^{NP} = NP^{NP}?

It is believed that the answers are NO
Logic Minimization is in $\text{coNP}^{\text{NP}}$

Two Boolean formulas $\phi$ and $\psi$ over the variables $x_1, \ldots, x_n$ are equivalent if they have the same value on every assignment to the variables

Are $x$ and $x \lor x$ equivalent? Yes

Are $x$ and $x \lor \neg x$ equivalent? No

Are $(x \lor \neg y) \land \neg(\neg x \land y)$ and $x \lor \neg y$ equivalent? Yes

A Boolean formula $\phi$ is minimal if no smaller formula is equivalent to $\phi$

$\text{MIN\text{-}FORMULA} = \{\phi \mid \phi \text{ is minimal}\}$
Theorem: \( \text{MIN-FORMULA} \in \text{coNP}^{\text{NP}} \)

Proof:
Define \( \text{NEQUIV} = \{(\phi, \psi)|\phi \text{ and } \psi \text{ are not equivalent}\} \)

Observation: \( \text{NEQUIV} \in \text{NP} \) (Why?)

Here is a \( \text{coNP}^{\text{NEQUIV}} \) machine for \( \text{MIN-FORMULA} \):

Given a formula \( \phi \),

*Try all formulas \( \psi \) smaller than \( \phi \):
  If \((\phi, \psi) \in \text{NEQUIV}\) then accept else reject*

\( \text{MIN-FORMULA} \) is not known to be in \( \text{coNP} \)
Decidable

coNP

MIN-FORMULA

NP

TAUT

FACTORIZATION

P

SAT

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable

Undecidable
Parting thoughts:
We discussed the first levels of the polynomial hierarchy. Contain many interesting computational problems. Much we don’t know.