Lecture 13: Oracle Turing Machines

Arijit Bishnu

13.04.2010

Outline

1. Oracle Turing Machines
2. Relativization
3. Defining the Polynomial Hierarchy using Oracle Machines

Outline

1 Oracle Turing Machines

2 Relativization

3 Defining the Polynomial Hierarchy using Oracle Machines

Oracle Turing Machines

- The meaning of the word Oracle according to Webster: a person (as a priestess of ancient Greece) through whom a deity is believed to speak; a shrine in which a deity reveals hidden knowledge or the divine purpose through such a person.

Oracle Turing Machines

- The meaning of the word Oracle according to Webster: a person (as a priestess of ancient Greece) through whom a deity is believed to speak; a shrine in which a deity reveals hidden knowledge or the divine purpose through such a person.
- Oracle machines are TMs that are given access to a black box or oracle that can solve the decision problem for some language $O \subseteq \{0,1\}^*$.

Oracle Turing Machines

- The meaning of the word Oracle according to Webster: a person (as a priestess of ancient Greece) through whom a deity is believed to speak; a shrine in which a deity reveals hidden knowledge or the divine purpose through such a person.
- Oracle machines are TMs that are given access to a black box or oracle that can solve the decision problem for some language $O \subseteq \{0,1\}^*$.
- An oracle is a language $O \subseteq \{0,1\}^*$.

Oracle Turing Machines

- The meaning of the word Oracle according to Webster: a person (as a priestess of ancient Greece) through whom a deity is believed to speak; a shrine in which a deity reveals hidden knowledge or the divine purpose through such a person.
- Oracle machines are TMs that are given access to a black box or oracle that can solve the decision problem for some language $O \subseteq \{0,1\}^*$.
- An oracle is a language $O \subseteq \{0,1\}^*$.
- An oracle Turing machine(OTM) M^O is an ordinary TM with an extra tape called the oracle tape.

Oracle Turing Machines

- The meaning of the word Oracle according to Webster: a person (as a priestess of ancient Greece) through whom a deity is believed to speak; a shrine in which a deity reveals hidden knowledge or the divine purpose through such a person.
- Oracle machines are TMs that are given access to a black box or oracle that can solve the decision problem for some language $O \subseteq \{0,1\}^*$.
- An oracle is a language $O \subseteq \{0,1\}^*$.
- An oracle Turing machine(OTM) M^O is an ordinary TM with an extra tape called the oracle tape.
- Whenever M writes a string $q \in \{0,1\}^*$ on the oracle tape, M is informed whether $q \in O$ in a single computational step.

Formalizing the Concept

Definition: Oracle TM

- An OTM is a TM M that has a special read-write tape called M's oracle tape and three special states q_{query}, q_{yes} and q_{no} apart from other states.

Formalizing the Concept

Definition: Oracle TM

- An OTM is a TM M that has a special read-write tape called M's oracle tape and three special states q_{query}, q_{yes} and q_{no} apart from other states.
- To execute M, we specify the input as usual; and a language $O \subseteq \{0,1\}^*$ that is used as an oracle for M.

Formalizing the Concept

Definition: Oracle TM

- An OTM is a TM M that has a special read-write tape called M's oracle tape and three special states q_{query}, q_{yes} and q_{no} apart from other states.
- To execute M, we specify the input as usual; and a language $O \subseteq \{0,1\}^*$ that is used as an oracle for M.
- While performing its computation, if M enters the state q_{query}, then M checks whether the contents of the oracle tape $w \in O$? If $w \in O$, M moves to the state q_{yes}, it moves to q_{no} if $q \notin O$.

Formalizing the Concept

Definition: Oracle TM

- An OTM is a TM M that has a special read-write tape called M's oracle tape and three special states q_{query}, q_{yes} and q_{no} apart from other states.
- To execute M, we specify the input as usual; and a language $O \subseteq \{0,1\}^*$ that is used as an oracle for M.
- While performing its computation, if M enters the state q_{query}, then M checks whether the contents of the oracle tape $w \in O$? If $w \in O$, M moves to the state q_{yes}, it moves to q_{no} if $q \notin O$.
- Regardless of the choice of O, a query like $w \in O$ counts for a single computational step of M.

Formalizing the Concept

Definition: Oracle TM

- An OTM is a TM M that has a special read-write tape called M's oracle tape and three special states q_{query}, q_{yes} and q_{no} apart from other states.
- To execute M, we specify the input as usual; and a language $O \subseteq \{0,1\}^*$ that is used as an oracle for M.
- While performing its computation, if M enters the state q_{query}, then M checks whether the contents of the oracle tape $w \in O$? If $w \in O$, M moves to the state q_{yes}, it moves to q_{no} if $q \notin O$.
- Regardless of the choice of O, a query like $w \in O$ counts for a single computational step of M.
- $M^O(x)$ denotes the output of the oracle TM M on input $x \in \{0,1\}^*$ with $O \subseteq \{0,1\}^*$ as the language.

Formalizing the Concept

Definition: Oracle TM

- An OTM is a TM M that has a special read-write tape called M's oracle tape and three special states q_{query}, q_{yes} and q_{no} apart from other states.
- To execute M, we specify the input as usual; and a language $O \subseteq \{0,1\}^*$ that is used as an oracle for M.
- While performing its computation, if M enters the state q_{query}, then M checks whether the contents of the oracle tape $w \in O$? If $w \in O$, M moves to the state q_{yes}, it moves to q_{no} if $q \notin O$.
- Regardless of the choice of O, a query like $w \in O$ counts for a single computational step of M.
- $M^O(x)$ denotes the output of the oracle TM M on input $x \in \{0,1\}^*$ with $O \subseteq \{0,1\}^*$ as the language.
- Non-deterministic OTMs can also be similarly defined.

New Classes

Definition: P^O and NP^O

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

Some Initial Results

Claim

$\overline{\text{SAT}} \in \text{P}^{\text{SAT}}$ where $\overline{\text{SAT}}$ denotes the language of unsatisfiable formulae.

Some Initial Results

Claim

$\overline{\text{SAT}} \in \text{P}^{\text{SAT}}$ where $\overline{\text{SAT}}$ denotes the language of unsatisfiable formulae.

Proof

Given oracle access to SAT, a deterministic poly-time OTM can query its oracle if $\varphi \in \text{SAT}$ and then give the opposite answer.

Some Initial Results

Claim

Let $O \in \mathrm{P}$. Then, $\mathrm{P}^O = \mathrm{P}$.

Some Initial Results

Claim

Let $O \in \mathrm{P}$. Then, $\mathrm{P}^O = \mathrm{P}$.

Proof

- It trivially follows $\mathrm{P} \subseteq \mathrm{P}^O$.
- If $O \in \mathrm{P}$, then replace each oracle call with a deterministic poly-time computation of O.

Some Initial Results

Claim

Let $O \in \mathrm{P}$. Then, $\mathrm{P}^O = \mathrm{P}$.

Proof

- It trivially follows $\mathrm{P} \subseteq \mathrm{P}^O$.
- If $O \in \mathrm{P}$, then replace each oracle call with a deterministic poly-time computation of O.
- Number of oracle calls can be polynomial and product of two polynomials is another polynomial. Hence, $\mathrm{P}^O \subseteq \mathrm{P}$.

Some Initial Results

Claim

Let EXPCOM be the following language:

$$\{< M, x, 1^n > \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps.}\}$$

Then, $\mathrm{P}^{\mathrm{EXPCOM}} = \mathrm{NP}^{\mathrm{EXPCOM}} = \mathrm{EXP}$, where $\mathrm{EXP} = \bigcup_c \mathrm{DTIME}(2^{n^c})$.

Some Initial Results

Claim

Let EXPCOM be the following language:

$$\{< M, x, 1^n > \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps.}\}$$

Then, $\mathrm{P}^{\mathrm{EXPCOM}} = \mathrm{NP}^{\mathrm{EXPCOM}} = \mathrm{EXP}$, where $\mathrm{EXP} = \bigcup_c \mathrm{DTIME}(2^{n^c})$.

Proof

Left as an exercise. Show $\mathrm{EXP} \subseteq \mathrm{P}^{\mathrm{EXPCOM}} \subseteq \mathrm{P}^{\mathrm{EXPCOM}} \subseteq \mathrm{EXP}$.

Some Initial Results

Claim

Let EXPCOM be the following language:

$$\{< M, x, 1^n > \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps.}\}$$

Then, $\mathrm{P}^{\mathrm{EXPCOM}} = \mathrm{NP}^{\mathrm{EXPCOM}} = \mathrm{EXP}$, where $\mathrm{EXP} = \bigcup_c \mathrm{DTIME}(2^{n^c})$.

Proof

Left as an exercise. Show $\mathrm{EXP} \subseteq \mathrm{P}^{\mathrm{EXPCOM}} \subseteq \mathrm{P}^{\mathrm{EXPCOM}} \subseteq \mathrm{EXP}$.

Claim

$\mathrm{NP} \subseteq \mathrm{P}^{\mathrm{SAT}}$ and $\mathrm{coNP} \subseteq \mathrm{P}^{\mathrm{SAT}}$

Some Initial Results

Claim

Let EXPCOM be the following language:

$$\{< M, x, 1^n > \mid M \text{ outputs 1 on } x \text{ within } 2^n \text{ steps.}\}$$

Then, $\mathrm{P}^{\mathrm{EXPCOM}} = \mathrm{NP}^{\mathrm{EXPCOM}} = \mathrm{EXP}$, where $\mathrm{EXP} = \bigcup_c \mathrm{DTIME}(2^{n^c})$.

Proof

Left as an exercise. Show $\mathrm{EXP} \subseteq \mathrm{P}^{\mathrm{EXPCOM}} \subseteq \mathrm{P}^{\mathrm{EXPCOM}} \subseteq \mathrm{EXP}$.

Claim

$\mathrm{NP} \subseteq \mathrm{P}^{\mathrm{SAT}}$ and $\mathrm{coNP} \subseteq \mathrm{P}^{\mathrm{SAT}}$

Proof

Left as an exercise.

Exercise

Exercise

Two boolean formulas ϕ and ψ are equivalent if the formulas have the same value on any assignment to the boolean variables. A formula is minimal if no smaller formula is equivalent to it. Let

$$\text{NONMIN} = \{< \phi > \mid \phi \text{ is not a minimal boolean formula.}\}$$

Show that $\text{NONMIN} \in \text{NP}^{\text{SAT}}$.

Outline

1 Oracle Turing Machines

2 Relativization

3 Defining the Polynomial Hierarchy using Oracle Machines

Relativization

We go back to diagonalization and its limits. For that, we use the concept of relativization.

Relativization

Relativization

We go back to diagonalization and its limits. For that, we use the concept of relativization.

Relativization

- The OTM introduced earlier allows us to "magically" solve the SAT problem in one step if our O was SAT.

Relativization

We go back to diagonalization and its limits. For that, we use the concept of relativization.

Relativization

- The OTM introduced earlier allows us to "magically" solve the SAT problem in one step if our O was SAT.
- Irrespective of the relation between P and NP, this OTM can solve any NP problem in deterministic polynomial time because every problem in NP is poly-time reducible to SAT.

Relativization

We go back to diagonalization and its limits. For that, we use the concept of relativization.

Relativization

- The OTM introduced earlier allows us to "magically" solve the SAT problem in one step if our O was SAT.
- Irrespective of the relation between P and NP, this OTM can solve any NP problem in deterministic polynomial time because every problem in NP is poly-time reducible to SAT.
- Such a TM is said to be computing relative to SAT. This is the significance of the term relativization.

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Theorem

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

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Theorem

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

- But, how does the above theorem help?

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Theorem

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

- But, how does the above theorem help?
- The diagonalization method is a simulation of one TM (Y) by another TM (X). X can determine the behaviour of Y and then behave differently.

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Theorem

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

- But, how does the above theorem help?
- The diagonalization method is a simulation of one TM (Y) by another TM (X). X can determine the behaviour of Y and then behave differently.
- Suppose, X and Y were given identical oracles.

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Theorem

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

- But, how does the above theorem help?
- The diagonalization method is a simulation of one TM (Y) by another TM (X). X can determine the behaviour of Y and then behave differently.
- Suppose, X and Y were given identical oracles.
- Now, whenever, Y queries the oracle, so can X and the simulation can proceed as before.

Limits of the Diagonalization Method

Our goal is to show the limits of diagonalization in proving the P vs NP question. To that end, we will prove the following theorem:

Theorem

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

- But, how does the above theorem help?
- The diagonalization method is a simulation of one TM (Y) by another TM (X). X can determine the behaviour of Y and then behave differently.
- Suppose, X and Y were given identical oracles.
- Now, whenever, Y queries the oracle, so can X and the simulation can proceed as before.
- So, any theorem proved about TMs using only diagonalization would still hold if both machines were given the same oracle.

Limits of the Diagonalization Method

P, NP and diagonalization

Limits of the Diagonalization Method

P, NP and diagonalization

- If we could show P $\neq$ NP by diagonalization, we conclude that they are different relative to any oracle as well.

Limits of the Diagonalization Method

P, NP and diagonalization

- If we could show $P \neq NP$ by diagonalization, we conclude that they are different relative to any oracle as well.
- But, $P^B = NP^B$; so the conclusion is false.

Limits of the Diagonalization Method

P, NP and diagonalization

- If we could show $P \neq NP$ by diagonalization, we conclude that they are different relative to any oracle as well.
- But, $P^B = NP^B$; so the conclusion is false.
- On the other side, no proof that relies on simulation (as in diagonalization) can show that two classes are same because that shows they are same relative to any oracle.

Limits of the Diagonalization Method

P, NP and diagonalization

- If we could show $P \neq NP$ by diagonalization, we conclude that they are different relative to any oracle as well.
- But, $P^B = NP^B$; so the conclusion is false.
- On the other side, no proof that relies on simulation (as in diagonalization) can show that two classes are same because that shows they are same relative to any oracle.
- But, $P^A \neq NP^A$.

Limits of the Diagonalization Method

Theorem

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

Limits of the Diagonalization Method

Theorem

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

Proof

Limits of the Diagonalization Method

Theorem

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

Proof

- How to fix B?

Limits of the Diagonalization Method

Theorem

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

Proof

- How to fix B?
- What about the hardest problem in the class PSPACE, i.e. TQBF?

Limits of the Diagonalization Method

Theorem

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

Proof

- How to fix B?
- What about the hardest problem in the class PSPACE, i.e. TQBF?
- One can easily verify $\mathrm{P}^{\mathrm{TQBF}} \subseteq \mathrm{NP}^{\mathrm{TQBF}}$.

Limits of the Diagonalization Method

Theorem

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

Proof

- How to fix B?
- What about the hardest problem in the class PSPACE, i.e. TQBF?
- One can easily verify $\mathrm{P}^{\mathrm{TQBF}} \subseteq \mathrm{NP}^{\mathrm{TQBF}}$.
- We now try $\mathrm{NP}^{\mathrm{TQBF}} \subseteq \mathrm{P}^{\mathrm{TQBF}}$.

Limits of the Diagonalization Method

Theorem

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

Proof

- How to fix B?
- What about the hardest problem in the class PSPACE, i.e. TQBF?
- One can easily verify $\mathrm{P}^{\mathrm{TQBF}} \subseteq \mathrm{NP}^{\mathrm{TQBF}}$.
- We now try $\mathrm{NP}^{\mathrm{TQBF}} \subseteq \mathrm{P}^{\mathrm{TQBF}}$.
- For this, we will look at a series of containment relations like:

$$\mathrm{NP}^{\mathrm{TQBF}} \subseteq \mathrm{NSPACE} \subseteq \mathrm{PSPACE} \subseteq \mathrm{P}^{\mathrm{TQBF}}.$$

Proving $NP^{TQBF} \subseteq P^{TQBF}$

Proving $NP^{TQBF} \subseteq NSPACE$

We can convert the NDOTM computing in poly-time to a NDTM using polynomial space that computes the answers to the queries regarding TQBF instead of using the oracle.

Proving $NP^{TQBF} \subseteq P^{TQBF}$

Proving $NP^{TQBF} \subseteq NSPACE$

We can convert the NDOTM computing in poly-time to a NDTM using polynomial space that computes the answers to the queries regarding TQBF instead of using the oracle.

Proving $NSPACE \subseteq PSPACE$

Savitch's theorem relating NSPACE and PSPACE.

Proving $NP^{TQBF} \subseteq P^{TQBF}$

Proving $NP^{TQBF} \subseteq NSPACE$

We can convert the NDOTM computing in poly-time to a NDTM using polynomial space that computes the answers to the queries regarding TQBF instead of using the oracle.

Proving $NSPACE \subseteq PSPACE$

Savitch's theorem relating NSPACE and PSPACE.

Proving $PSPACE \subseteq P^{TQBF}$

This relation holds as TQBF is PSPACE-complete.

Proving $NP^{TQBF} \subseteq P^{TQBF}$

Proving $NP^{TQBF} \subseteq NSPACE$

We can convert the NDOTM computing in poly-time to a NDTM using polynomial space that computes the answers to the queries regarding TQBF instead of using the oracle.

Proving $NSPACE \subseteq PSPACE$

Savitch's theorem relating NSPACE and PSPACE.

Proving $PSPACE \subseteq P^{TQBF}$

This relation holds as TQBF is PSPACE-complete.

Finally

$NP^{TQBF} \subseteq P^{TQBF}$. So, setting $B = \text{TQBF}$ works. Thus, $P^{TQBF} = NP^{TQBF}$

The other one

$\exists$ oracle A such that $\mathrm{P}^A \neq \mathrm{NP}^A$.

One can show with some effort that such an oracle A can be designed. We skip it. Refer text book.

Outline

1 Oracle Turing Machines

2 Relativization

3 Defining the Polynomial Hierarchy using Oracle Machines

Characterizing Polynomial Hierarchy using Oracle Machines

Recall Σ_iSAT

$$\Sigma_i\text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

So, $\Sigma_1\text{SAT} = \exists x_1 \ \varphi(x_1) = 1$. We now show a characterization of Σ_2^p using a NDTM with oracle access to SAT.

Characterizing Polynomial Hierarchy using Oracle Machines

Recall Σ_iSAT

$$\Sigma_i\text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

So, $\Sigma_1\text{SAT} = \exists x_1 \ \varphi(x_1) = 1$. We now show a characterization of Σ_2^p using a NDTM with oracle access to SAT.

Theorem

$\Sigma_2^p = \text{NP}^{\text{SAT}}$

$\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

Proof of $\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

- Fix a language $L \in \Sigma_2^p$. So, $\exists$ a poly-time TM M and a polynomial q such that

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2) = 1.$$

$\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

Proof of $\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

- Fix a language $L \in \Sigma_2^p$. So, $\exists$ a poly-time TM M and a polynomial q such that

  $$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2) = 1.$$

- For every fixed x, u_1, the above statement becomes $\forall u_2 M(x, u_1, u_2) = 1$ which is the negation of an NP-statement and hence, its truth can be determined using an oracle for SAT.

$\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

Proof of $\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

- Fix a language $L \in \Sigma_2^p$. So, $\exists$ a poly-time TM M and a polynomial q such that

  $$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2) = 1.$$

- For every fixed x, u_1, the above statement becomes $\forall u_2 M(x, u_1, u_2) = 1$ which is the negation of an NP-statement and hence, its truth can be determined using an oracle for SAT.
- So, an NDTM N that is given oracle access to SAT can decide L. How?

$\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

Proof of $\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

- Fix a language $L \in \Sigma_2^p$. So, $\exists$ a poly-time TM M and a polynomial q such that

  $$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2) = 1.$$

- For every fixed x, u_1, the above statement becomes $\forall u_2 M(x, u_1, u_2) = 1$ which is the negation of an NP-statement and hence, its truth can be determined using an oracle for SAT.
- So, an NDTM N that is given oracle access to SAT can decide L. How?
- On input x, non-deterministically guess u_1 and use the oracle to decide if $\forall u_2 M(x, u_1, u_2) = 1$.

$\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

Proof of $\Sigma_2^p \subseteq \mathrm{NP}^{\mathrm{SAT}}$

- Fix a language $L \in \Sigma_2^p$. So, $\exists$ a poly-time TM M and a polynomial q such that

  $$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2) = 1.$$

- For every fixed x, u_1, the above statement becomes $\forall u_2 M(x, u_1, u_2) = 1$ which is the negation of an NP-statement and hence, its truth can be determined using an oracle for SAT.
- So, an NDTM N that is given oracle access to SAT can decide L. How?
- On input x, non-deterministically guess u_1 and use the oracle to decide if $\forall u_2 M(x, u_1, u_2) = 1$.
- So, $x \in L$ iff $\exists$ a choice u_1 that makes N accept.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.
- We need to replace N by a machine for the class Σ_2^p.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.
- We need to replace N by a machine for the class Σ_2^p.
- Non-deterministically guess all future queries as well as SAT oracle's answers and then make a single coNP query whose answer verifies that all this guessing was correct.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.
- We need to replace N by a machine for the class Σ_2^p.
- Non-deterministically guess all future queries as well as SAT oracle's answers and then make a single coNP query whose answer verifies that all this guessing was correct.
- $x \in L$ iff $\exists$ a sequence of non-deterministic choices and correct oracle answers that makes N accept x.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.
- We need to replace N by a machine for the class Σ_2^p.
- Non-deterministically guess all future queries as well as SAT oracle's answers and then make a single coNP query whose answer verifies that all this guessing was correct.
- $x \in L$ iff $\exists$ a sequence of non-deterministic choices and correct oracle answers that makes N accept x.
- So, $\exists$ a sequence of choices $c_1, c_2, \ldots, c_m \in \{0, 1\}$ and answers to oracle queries $a_1, \ldots, a_k \in \{0, 1\}$ s.t. on input x, if N uses the above choices in its execution and receives a_i as the answer to its i-th query, then

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.
- We need to replace N by a machine for the class Σ_2^p.
- Non-deterministically guess all future queries as well as SAT oracle's answers and then make a single coNP query whose answer verifies that all this guessing was correct.
- $x \in L$ iff $\exists$ a sequence of non-deterministic choices and correct oracle answers that makes N accept x.
- So, $\exists$ a sequence of choices $c_1, c_2, \ldots, c_m \in \{0, 1\}$ and answers to oracle queries $a_1, \ldots, a_k \in \{0, 1\}$ s.t. on input x, if N uses the above choices in its execution and receives a_i as the answer to its i-th query, then
  - M reaches the accepting state q_{accept}.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Fix a language $L \in NP^{SAT}$. L is decidable by a poly-time NDTM N with oracle access to SAT.
- We need to replace N by a machine for the class Σ_2^p.
- Non-deterministically guess all future queries as well as SAT oracle's answers and then make a single coNP query whose answer verifies that all this guessing was correct.
- $x \in L$ iff $\exists$ a sequence of non-deterministic choices and correct oracle answers that makes N accept x.
- So, $\exists$ a sequence of choices $c_1, c_2, \ldots, c_m \in \{0, 1\}$ and answers to oracle queries $a_1, \ldots, a_k \in \{0, 1\}$ s.t. on input x, if N uses the above choices in its execution and receives a_i as the answer to its i-th query, then
  - M reaches the accepting state q_{accept}.
  - All the answers are correct.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Let φ_i denote the i-th query that N makes to its oracle when executing on x using the above choices and answers.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Let φ_i denote the i-th query that N makes to its oracle when executing on x using the above choices and answers.
- Then all the answers are correct can be encoded as:

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Let φ_i denote the i-th query that N makes to its oracle when executing on x using the above choices and answers.
- Then all the answers are correct can be encoded as:
  - If $a_i = 1$, $\exists$ an assignment u_i s.t. $\varphi_i(u_i) = 1$.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Let φ_i denote the i-th query that N makes to its oracle when executing on x using the above choices and answers.
- Then all the answers are correct can be encoded as:
  - If $a_i = 1$, $\exists$ an assignment u_i s.t. $\varphi_i(u_i) = 1$.
  - If $a_i = 0$, $\forall$ assignment v_i s.t. $\varphi_i(v_i) = 0$.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Let φ_i denote the i-th query that N makes to its oracle when executing on x using the above choices and answers.
- Then all the answers are correct can be encoded as:
  - If $a_i = 1$, $\exists$ an assignment u_i s.t. $\varphi_i(u_i) = 1$.
  - If $a_i = 0$, $\forall$ assignment v_i s.t. $\varphi_i(v_i) = 0$.
- So, we have
  $x \in L \iff \exists c_1, \ldots c_m, a_1, \ldots a_k, u_1, \ldots, u_k \forall v_1, \ldots v_k$ such that N accepts x using choices $c_1, \ldots, c_m$ and answers $a_1, \ldots, a_k$ AND $\forall i \in [k]$, if $a_i = 1$ then $\varphi_i(u_i) = 1$ AND $\forall i \in [k]$, if $a_i = 0$ then $\varphi_i(v_i) = 0$.

$NP^{SAT} \subseteq \Sigma_2^p$

Proof of $NP^{SAT} \subseteq \Sigma_2^p$

- Let φ_i denote the i-th query that N makes to its oracle when executing on x using the above choices and answers.
- Then all the answers are correct can be encoded as:
  - If $a_i = 1$, $\exists$ an assignment u_i s.t. $\varphi_i(u_i) = 1$.
  - If $a_i = 0$, $\forall$ assignment v_i s.t. $\varphi_i(v_i) = 0$.
- So, we have
  $x \in L \iff \exists c_1, \ldots c_m, a_1, \ldots a_k, u_1, \ldots, u_k \forall v_1, \ldots v_k$ such that N accepts x using choices $c_1, \ldots, c_m$ and answers $a_1, \ldots, a_k$ AND $\forall i \in [k]$, if $a_i = 1$ then $\varphi_i(u_i) = 1$ AND $\forall i \in [k]$, if $a_i = 0$ then $\varphi_i(v_i) = 0$.
- This implies $L \in \Sigma_2^p$.