

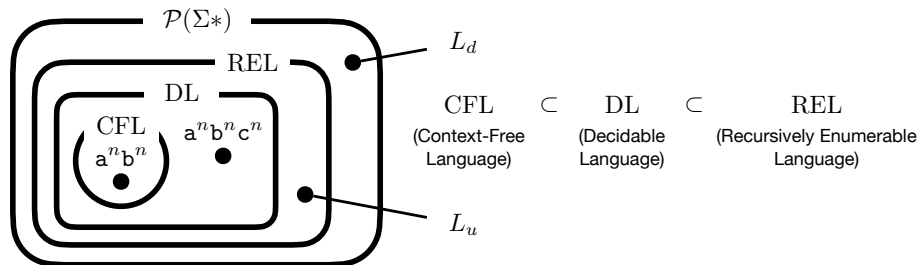
Lecture 26 – P, NP, and NP-Complete Problems

COSE215: Theory of Computation

Jihyeok Park

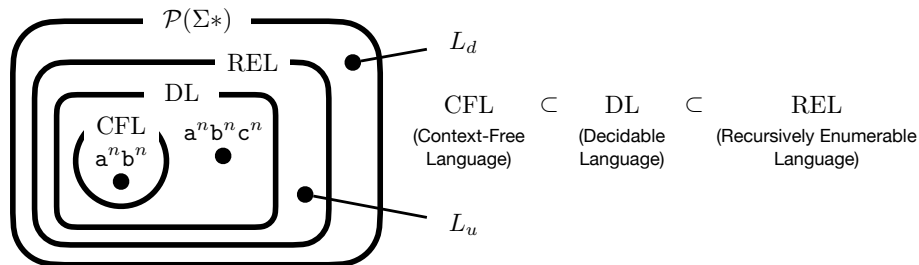


2025 Spring



Definition (Decision Problem)

A **decision problem** π is a computational problem whose answer is either **yes** or **no** for a given input.



Definition (Decision Problem)

A **decision problem** π is a computational problem whose answer is either **yes** or **no** for a given input.

In this lecture, we will **classify** decision problems based on the **time complexity** of possible TMs (or NTMs) that solve the problems.

1. **P**

Time Complexity of TMs

P – Polynomial Time Complexity (Tractable Problems)

2. **NP**

Time Complexity of NTMs

NP – Nondeterministic Polynomial Time Complexity

NP – Verifier-based Definition

3. **NP-complete**

Polynomial Time Reduction (\leq_P)

NP-complete – Hardest Problems in **NP**

<SAT> – The First **NP-complete** Problem

Other **NP-complete** Problems

4. Major Unsolved Problem: **P = NP?**

1. **P**

Time Complexity of TMs

P – Polynomial Time Complexity (Tractable Problems)

2. **NP**

Time Complexity of NTMs

NP – Nondeterministic Polynomial Time Complexity

NP – Verifier-based Definition

3. **NP-complete**

Polynomial Time Reduction (\leq_P)

NP-complete – Hardest Problems in **NP**

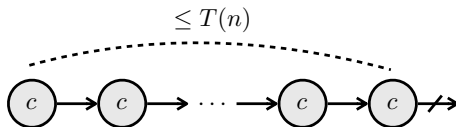
<SAT> – The First **NP-complete** Problem

Other **NP-complete** Problems

4. Major Unsolved Problem: **P = NP?**

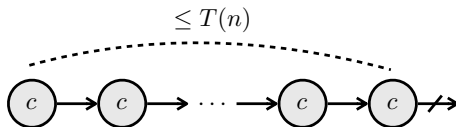
Definition (Time Complexity of TMs)

We say a **Turing machine (TM)** M has a **time complexity** $T : \mathbb{N} \rightarrow \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n .



Definition (Time Complexity of TMs)

We say a **Turing machine (TM)** M has a **time complexity** $T : \mathbb{N} \rightarrow \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n .

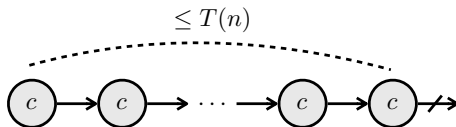


Definition (DTIME)

A decision problem π is in **DTIME**($T(n)$) if it is decidable by a TM M whose time complexity is $T(n)$.

Definition (Time Complexity of TMs)

We say a **Turing machine (TM)** M has a **time complexity** $T : \mathbb{N} \rightarrow \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n .



Definition (DTIME)

A decision problem π is in **DTIME**($T(n)$) if it is decidable by a TM M whose time complexity is $T(n)$.

We often use a **big O notation** to describe the time complexity of a TM:

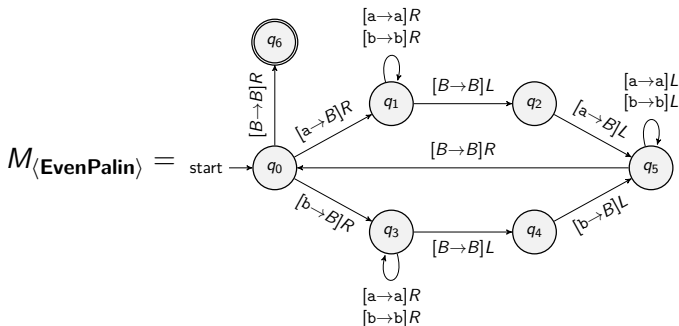
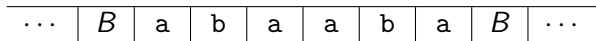
$$f(n) = O(g(n)) \iff \exists k \in \mathbb{N}, n_0 \in \mathbb{N}. \forall n \geq n_0. f(n) \leq k \cdot g(n)$$

⟨**EvenPalin**⟩ – Is a word $w \in \{a, b\}^*$ an even-length palindrome?

⟨**EvenPalin**⟩ – Is a word $w \in \{a, b\}^*$ an even-length palindrome?

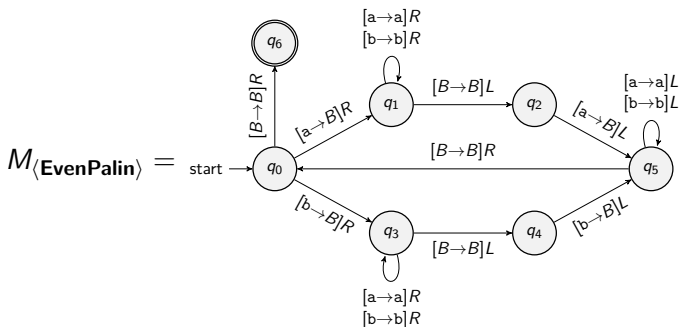
...	<i>B</i>	a	b	a	a	b	a	<i>B</i>	...
-----	----------	---	---	---	---	---	---	----------	-----

$\langle \mathbf{EvenPalin} \rangle$ – Is a word $w \in \{a, b\}^*$ an even-length palindrome?



$\langle \mathbf{EvenPalin} \rangle$ – Is a word $w \in \{a, b\}^*$ an even-length palindrome?

...	B	a	b	a	a	b	a	B	...
-----	---	---	---	---	---	---	---	---	-----



The decision problem $\langle \mathbf{EvenPalin} \rangle$ is decidable by the above TM whose time complexity is $T(n) = (n + 1)(n + 2)/2 = O(n^2)$.

$$\langle \mathbf{EvenPalin} \rangle \in \mathbf{DTIME}(O(n^2))$$

Definition (P – Polynomial Time Complexity)

A decision problem π is in **P** if it is decidable by a TM M whose time complexity is a **polynomial function** (i.e., $T(n) = O(n^k)$ for some $k \geq 0$).

$$\mathbf{P} = \bigcup_{k \geq 0} \mathbf{DTIME}(O(n^k))$$

Definition (**P** – Polynomial Time Complexity)

A decision problem π is in **P** if it is decidable by a TM M whose time complexity is a **polynomial function** (i.e., $T(n) = O(n^k)$ for some $k \geq 0$).

$$\mathbf{P} = \bigcup_{k \geq 0} \mathbf{DTIME}(O(n^k))$$

For example, the decision problem $\langle \mathbf{EvenPalin} \rangle$ is in **P**.

$$\langle \mathbf{EvenPalin} \rangle \in \mathbf{DTIME}(O(n^2)) \subseteq \mathbf{P}$$

Definition (**P** – Polynomial Time Complexity)

A decision problem π is in **P** if it is decidable by a TM M whose time complexity is a **polynomial function** (i.e., $T(n) = O(n^k)$ for some $k \geq 0$).

$$\mathbf{P} = \bigcup_{k \geq 0} \mathbf{DTIME}(O(n^k))$$

For example, the decision problem $\langle \mathbf{EvenPalin} \rangle$ is in **P**.

$$\langle \mathbf{EvenPalin} \rangle \in \mathbf{DTIME}(O(n^2)) \subseteq \mathbf{P}$$

Definition (Tractable Problems)

A problem π is called a **tractable problem** if it is a **P** problem.

1. **P**

Time Complexity of TMs

P – Polynomial Time Complexity (Tractable Problems)

2. **NP**

Time Complexity of NTMs

NP – Nondeterministic Polynomial Time Complexity

NP – Verifier-based Definition

3. **NP-complete**

Polynomial Time Reduction (\leq_P)

NP-complete – Hardest Problems in **NP**

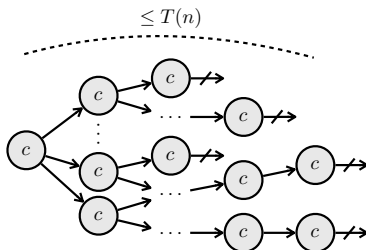
<SAT> – The First **NP-complete** Problem

Other **NP-complete** Problems

4. Major Unsolved Problem: **P = NP?**

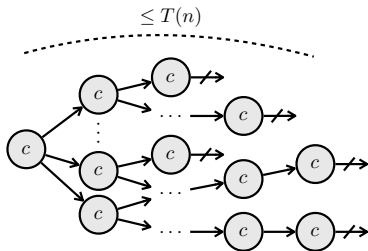
Definition (Time Complexity of NTMs)

We say a **nondeterministic Turing machine (NTM)** M has a **time complexity** $T : \mathbb{N} \rightarrow \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n .



Definition (Time Complexity of NTMs)

We say a **nondeterministic Turing machine (NTM)** M has a **time complexity** $T : \mathbb{N} \rightarrow \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n .



Definition (NTIME)

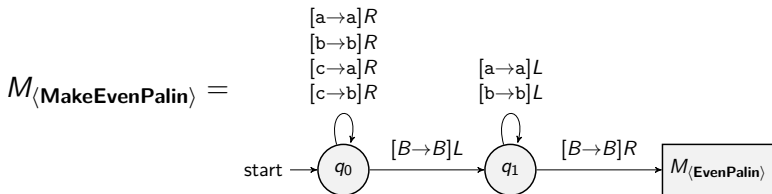
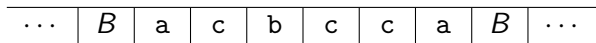
A decision problem π is in **NTIME**($T(n)$) if it is decidable by a NTM M whose time complexity is $T(n)$.

⟨MakeEvenPalin⟩ – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c's with a's or b's?

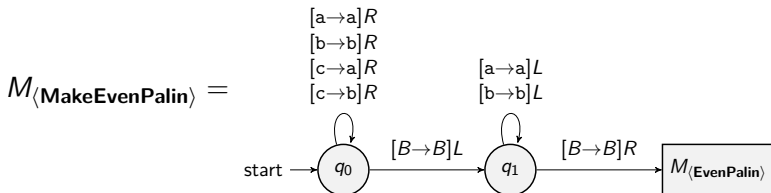
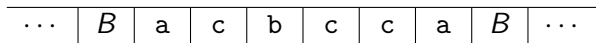
⟨MakeEvenPalin⟩ – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c's with a's or b's?

...	<i>B</i>	a	c	b	c	c	a	<i>B</i>	...
-----	----------	---	---	---	---	---	---	----------	-----

$\langle \text{MakeEvenPalin} \rangle$ – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c's with a's or b's?



$\langle \mathbf{MakeEvenPalin} \rangle$ – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c's with a's or b's?



The decision problem $\langle \mathbf{MakeEvenPalin} \rangle$ is decidable by the above NTM whose time complexity is $T(n) = 2(n + 1) + O(n^2) = O(n^2)$.

$$\langle \mathbf{MakeEvenPalin} \rangle \in \mathbf{NTIME}(O(n^2))$$

Definition (**NP** – Nondeterministic Polynomial Time Complexity)

A decision problem π is in **NP** if it is decidable by an NTM M whose time complexity is a **polynomial function** (i.e., $T(n) = O(n^k)$ for some $k \geq 0$).

$$\mathbf{NP} = \bigcup_{k \geq 0} \mathbf{NTIME}(O(n^k))$$

For example, the decision problem $\langle \mathbf{MakeEvenPalin} \rangle$ is in **NP**.

$$\langle \mathbf{MakeEvenPalin} \rangle \in \mathbf{NTIME}(O(n^2)) \subseteq \mathbf{NP}$$

Definition (Search Problem)

A **search problem** π is a decision problem that asks for the existence of a **witness** x (i.e., a solution) in the search space $S(w)$ for a given input w , satisfying the another decision problem π' as a **verification problem**.

$$\forall w \in \Sigma^*. \pi(w) = \text{yes} \iff \exists x \in S(w). \pi'(w, x) = \text{yes}$$

Definition (Search Problem)

A **search problem** π is a decision problem that asks for the existence of a **witness** x (i.e., a solution) in the search space $S(w)$ for a given input w , satisfying the another decision problem π' as a **verification problem**.

$$\forall w \in \Sigma^*. \pi(w) = \text{yes} \iff \exists x \in S(w). \pi'(w, x) = \text{yes}$$

For example, $\langle \text{MakeEvenPalin} \rangle$ is a **search problem** with $\langle \text{EvenPalin} \rangle$ as a **verification problem**:

Definition (Search Problem)

A **search problem** π is a decision problem that asks for the existence of a **witness** x (i.e., a solution) in the search space $S(w)$ for a given input w , satisfying the another decision problem π' as a **verification problem**.

$$\forall w \in \Sigma^*. \pi(w) = \text{yes} \iff \exists x \in S(w). \pi'(w, x) = \text{yes}$$

For example, $\langle \mathbf{MakeEvenPalin} \rangle$ is a **search problem** with $\langle \mathbf{EvenPalin} \rangle$ as a **verification problem**:

$$\langle \mathbf{MakeEvenPalin} \rangle(w) = \text{yes} \iff \exists x \in S(w). \langle \mathbf{EvenPalin} \rangle(x) = \text{yes}$$

where the search space $S(w)$ of an input w is defined as follows:

$$S(w) = \{x \mid x = (\text{a possible replacement of all } c\text{'s in } w \text{ with } a\text{'s or } b\text{'s})\}$$

Definition (Search Problem)

A **search problem** π is a decision problem that asks for the existence of a **witness** x (i.e., a solution) in the search space $S(w)$ for a given input w , satisfying the another decision problem π' as a **verification problem**.

$$\forall w \in \Sigma^*. \pi(w) = \text{yes} \iff \exists x \in S(w). \pi'(w, x) = \text{yes}$$

For example, $\langle \mathbf{MakeEvenPalin} \rangle$ is a **search problem** with $\langle \mathbf{EvenPalin} \rangle$ as a **verification problem**:

$$\langle \mathbf{MakeEvenPalin} \rangle(w) = \text{yes} \iff \exists x \in S(w). \langle \mathbf{EvenPalin} \rangle(x) = \text{yes}$$

where the search space $S(w)$ of an input w is defined as follows:

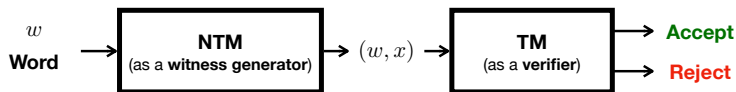
$$S(w) = \{x \mid x = (\text{a possible replacement of all } c\text{'s in } w \text{ with } a\text{'s or } b\text{'s})\}$$

e.g., $w = acbccca$ $S(w) = \left\{ \begin{array}{cccc} aabaaa, & aababa, & aabbba, & aabbba, \\ abbaaa, & abbaba, & abbbba, & abbbba \end{array} \right\}$

Definition (NP – Verifier-based Definition)

A search problem π defined with a verification problem π' is in **NP** if there is a polynomial time TM M as a **verifier** for π :

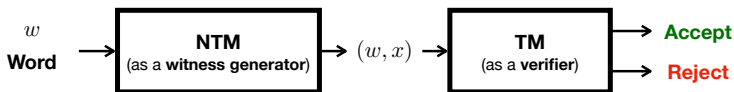
$$\forall w \in \Sigma^*. \forall x \in S(w). \pi'(w, x) = \text{yes} \iff (w, x) \in L(M)$$



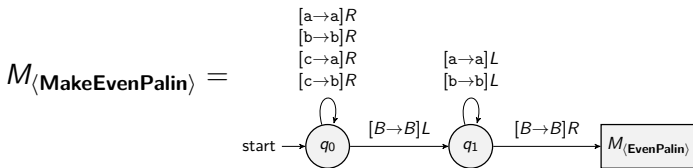
Definition (NP – Verifier-based Definition)

A search problem π defined with a verification problem π' is in **NP** if there is a polynomial time TM M as a **verifier** for π :

$$\forall w \in \Sigma^*. \forall x \in S(w). \pi'(w, x) = \text{yes} \iff (w, x) \in L(M)$$



For example, $\langle \text{MakeEvenPalin} \rangle$ is a search problem in **NP**:



NP – Example: $\langle \text{SAT} \rangle$

$\langle \text{SAT} \rangle$ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, \wedge , \vee , and \neg) satisfiable?

$\langle \text{SAT} \rangle$ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, \wedge , \vee , and \neg) satisfiable?

For example, is the following Boolean formula satisfiable?

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

$\langle \text{SAT} \rangle$ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, \wedge , \vee , and \neg) satisfiable?

For example, is the following Boolean formula satisfiable?

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Yes! For example, $x_1 = \#f$, $x_2 = \#f$, and $x_3 = \#t$ is a satisfying assignment.

$\langle \text{SAT} \rangle$ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, \wedge , \vee , and \neg) satisfiable?

For example, is the following Boolean formula satisfiable?

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Yes! For example, $x_1 = \#f$, $x_2 = \#f$, and $x_3 = \#t$ is a satisfying assignment.

Is it $\langle \text{SAT} \rangle$ in **NP**?

$\langle \text{SAT} \rangle$ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, \wedge , \vee , and \neg) satisfiable?

For example, is the following Boolean formula satisfiable?

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Yes! For example, $x_1 = \#f$, $x_2 = \#f$, and $x_3 = \#t$ is a satisfying assignment.

Is it $\langle \text{SAT} \rangle$ in **NP**? Yes!

We can construct a polynomial time TM as a **verifier** for $\langle \text{SAT} \rangle$, which takes 1) a **Boolean formula** and 1) an **assignment** of Boolean variables, and checks whether the assignment satisfies the formula.

$\langle \text{SAT} \rangle$ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, \wedge , \vee , and \neg) satisfiable?

For example, is the following Boolean formula satisfiable?

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Yes! For example, $x_1 = \#f$, $x_2 = \#f$, and $x_3 = \#t$ is a satisfying assignment.

Is it $\langle \text{SAT} \rangle$ in **NP**? Yes!

We can construct a polynomial time TM as a **verifier** for $\langle \text{SAT} \rangle$, which takes 1) a **Boolean formula** and 1) an **assignment** of Boolean variables, and checks whether the assignment satisfies the formula.

In other words, we can construct a polynomial time NTM for $\langle \text{SAT} \rangle$ by 1) **generating all assignments** of Boolean variables and 2) **verifying** whether the assignment satisfies the formula using the verifier.

1. **P**

Time Complexity of TMs

P – Polynomial Time Complexity (Tractable Problems)

2. **NP**

Time Complexity of NTMs

NP – Nondeterministic Polynomial Time Complexity

NP – Verifier-based Definition

3. **NP-complete**

Polynomial Time Reduction (\leq_P)

NP-complete – Hardest Problems in **NP**

<SAT> – The First **NP-complete** Problem

Other **NP-complete** Problems

4. Major Unsolved Problem: **P = NP?**

Definition (Polynomial Time Reduction (\leq_P))

A decision problem π_1 is **polynomial time reducible** to another decision problem π_2 (denoted by $\pi_1 \leq_P \pi_2$) if there exists a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that:

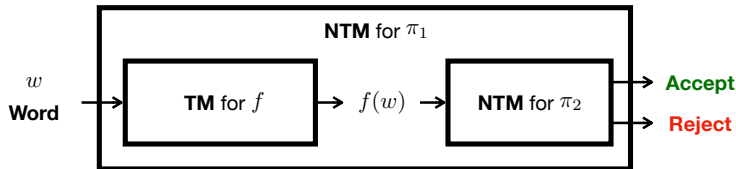
$$\forall w \in \Sigma^*. \pi_1(w) = \text{yes} \iff \pi_2(f(w)) = \text{yes}$$

Definition (Polynomial Time Reduction (\leq_P))

A decision problem π_1 is **polynomial time reducible** to another decision problem π_2 (denoted by $\pi_1 \leq_P \pi_2$) if there exists a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that:

$$\forall w \in \Sigma^*. \pi_1(w) = \text{yes} \iff \pi_2(f(w)) = \text{yes}$$

We say that π_2 is **harder** than π_1 if $\pi_1 \leq_P \pi_2$ because we can solve π_1 in polynomial time if we can solve π_2 in polynomial time.

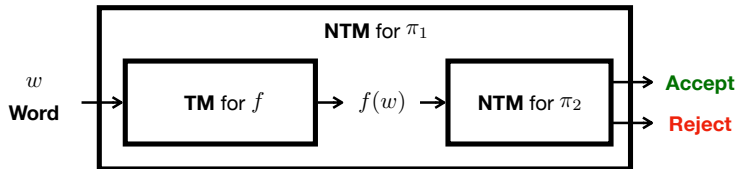


Definition (Polynomial Time Reduction (\leq_P))

A decision problem π_1 is **polynomial time reducible** to another decision problem π_2 (denoted by $\pi_1 \leq_P \pi_2$) if there exists a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that:

$$\forall w \in \Sigma^*. \pi_1(w) = \text{yes} \iff \pi_2(f(w)) = \text{yes}$$

We say that π_2 is **harder** than π_1 if $\pi_1 \leq_P \pi_2$ because we can solve π_1 in polynomial time if we can solve π_2 in polynomial time.



If a decision problem π_2 is in **NP** and $\pi_1 \leq_P \pi_2$, then π_1 is in **NP**.

Consider the following two decision problems:

- **⟨MakeEvenPalin⟩** – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c 's with a 's or b 's?
- **⟨SAT⟩** – Is a given Boolean formula satisfiable?

Consider the following two decision problems:

- **⟨MakeEvenPalin⟩** – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c 's with a 's or b 's?
- **⟨SAT⟩** – Is a given Boolean formula satisfiable?

We can show that **⟨MakeEvenPalin⟩** \leq_P **⟨SAT⟩** by the following polynomial time computable function f :

$$f(a_1 a_2 \cdots a_n) = \bigwedge_{i=1}^n ((x_i \wedge x_{n+1-i}) \vee (\neg x_i \wedge \neg x_{n+1-i})) \\ \wedge \bigwedge \{x_i \mid a_i = a\} \wedge \bigwedge \{\neg x_i \mid a_i = b\}$$

($x_i = \#t$ means that $a_i = a$ and $x_i = \#f$ means that $a_i = b$)

Consider the following two decision problems:

- **⟨MakeEvenPalin⟩** – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c 's with a 's or b 's?
- **⟨SAT⟩** – Is a given Boolean formula satisfiable?

We can show that **⟨MakeEvenPalin⟩** \leq_P **⟨SAT⟩** by the following polynomial time computable function f :

$$f(a_1 a_2 \cdots a_n) = \bigwedge_{i=1}^n ((x_i \wedge x_{n+1-i}) \vee (\neg x_i \wedge \neg x_{n+1-i})) \\ \wedge \bigwedge \{x_i \mid a_i = a\} \wedge \bigwedge \{\neg x_i \mid a_i = b\}$$

($x_i = \#t$ means that $a_i = a$ and $x_i = \#f$ means that $a_i = b$)

For example,

$$f(acba) = ((x_1 \wedge x_4) \vee (\neg x_1 \wedge \neg x_4)) \wedge ((x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3)) \wedge x_1 \wedge \neg x_3 \wedge x_4$$

Consider the following two decision problems:

- **⟨MakeEvenPalin⟩** – Is a word $w \in \{a, b, c\}^*$ convertible to an even-length palindrome by replacing all c 's with a 's or b 's?
- **⟨SAT⟩** – Is a given Boolean formula satisfiable?

We can show that **⟨MakeEvenPalin⟩** \leq_P **⟨SAT⟩** by the following polynomial time computable function f :

$$f(a_1 a_2 \cdots a_n) = \bigwedge_{i=1}^n ((x_i \wedge x_{n+1-i}) \vee (\neg x_i \wedge \neg x_{n+1-i})) \\ \wedge \bigwedge \{x_i \mid a_i = a\} \wedge \bigwedge \{\neg x_i \mid a_i = b\}$$

($x_i = \#t$ means that $a_i = a$ and $x_i = \#f$ means that $a_i = b$)

For example,

$$f(acba) = ((x_1 \wedge x_4) \vee (\neg x_1 \wedge \neg x_4)) \wedge ((x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3)) \wedge x_1 \wedge \neg x_3 \wedge x_4$$

Thus, we can solve **⟨MakeEvenPalin⟩** using a machine for **⟨SAT⟩**, and **⟨SAT⟩** is harder problem than **⟨MakeEvenPalin⟩**.

Definition (**NP-hard** – Harder Problems Than All **NP**)

A decision problem π is in **NP-hard** if $\forall \pi' \in \mathbf{NP}, \pi' \leq_P \pi$.

Definition (**NP-hard** – Harder Problems Than All NP)

A decision problem π is in **NP-hard** if $\forall \pi' \in \mathbf{NP}, \pi' \leq_P \pi$.

In other word, π is in **NP-hard** if π is **harder than all problems in NP**.

Definition (NP-hard – Harder Problems Than All NP)

A decision problem π is in **NP-hard** if $\forall \pi' \in \mathbf{NP}, \pi' \leq_P \pi$.

In other word, π is in **NP-hard** if π is **harder than all problems in NP**.

Definition (NP-complete – Hardest Problems in NP)

A decision problem π is in **NP-complete** if

- 1 π is in **NP**, and
- 2 π is in **NP-hard** (i.e., $\forall \pi' \in \mathbf{NP}, \pi' \leq_P \pi$).

Definition (NP-hard – Harder Problems Than All NP)

A decision problem π is in **NP-hard** if $\forall \pi' \in \text{NP}, \pi' \leq_P \pi$.

In other word, π is in **NP-hard** if π is **harder than all problems in NP**.

Definition (NP-complete – Hardest Problems in NP)

A decision problem π is in **NP-complete** if

- 1 π is in **NP**, and
- 2 π is in **NP-hard** (i.e., $\forall \pi' \in \text{NP}, \pi' \leq_P \pi$).

In other word, π is in **NP-complete** if π is the **hardest problem in NP**.

Theorem (Cook–Levin theorem)

⟨SAT⟩ *is in* NP-complete.

¹https://en.wikipedia.org/wiki/Cook-Levin_theorem

Theorem (Cook–Levin theorem)

$\langle \text{SAT} \rangle$ *is in* **NP-complete**.

We need to show that

- 1 $\langle \text{SAT} \rangle$ is in **NP**, and
- 2 $\langle \text{SAT} \rangle$ is in **NP-hard**.

¹https://en.wikipedia.org/wiki/Cook-Levin_theorem

Theorem (Cook–Levin theorem)

$\langle \text{SAT} \rangle$ *is in* **NP-complete**.

We need to show that

- ① $\langle \text{SAT} \rangle$ is in **NP**, and
- ② $\langle \text{SAT} \rangle$ is in **NP-hard**.

For ①, we already know that $\langle \text{SAT} \rangle$ is in **NP**.

¹https://en.wikipedia.org/wiki/Cook-Levin_theorem

Theorem (Cook–Levin theorem)

$\langle \text{SAT} \rangle$ is in **NP-complete**.

We need to show that

- ① $\langle \text{SAT} \rangle$ is in **NP**, and
- ② $\langle \text{SAT} \rangle$ is in **NP-hard**.

For ①, we already know that $\langle \text{SAT} \rangle$ is in **NP**.

For ②, we need to show that $\forall \pi \in \mathbf{NP}, \pi \leq_P \langle \text{SAT} \rangle$.

¹https://en.wikipedia.org/wiki/Cook-Levin_theorem

Theorem (Cook–Levin theorem)

$\langle \text{SAT} \rangle$ is in **NP-complete**.

We need to show that

- ① $\langle \text{SAT} \rangle$ is in **NP**, and
- ② $\langle \text{SAT} \rangle$ is in **NP-hard**.

For ①, we already know that $\langle \text{SAT} \rangle$ is in **NP**.

For ②, we need to show that $\forall \pi \in \mathbf{NP}, \pi \leq_P \langle \text{SAT} \rangle$.

The core idea is to simulate an NTM M for π using a Boolean formula ϕ such that ϕ is satisfiable if and only if M accepts w . But, we skip the details of the proof. Please refer to the link¹ for the details.

¹https://en.wikipedia.org/wiki/Cook-Levin_theorem

Theorem (Lemma)

A decision problem π is in **NP-hard** if $\langle \mathbf{SAT} \rangle \leq_P \pi$

²https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Theorem (Lemma)

*A decision problem π is in **NP-hard** if $\langle \mathbf{SAT} \rangle \leq_P \pi$*

This lemma is very useful to show that a decision problem π is in **NP-complete** by showing that 1) π is in **NP** and 2) $\langle \mathbf{SAT} \rangle \leq_P \pi$.

²https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Theorem (Lemma)

A decision problem π is in **NP-hard** if $\langle \mathbf{SAT} \rangle \leq_P \pi$

This lemma is very useful to show that a decision problem π is in **NP-complete** by showing that 1) π is in **NP** and 2) $\langle \mathbf{SAT} \rangle \leq_P \pi$.

We can show that all of the following decision problems are in **NP-complete** by using this lemma:²

- $\langle \mathbf{SubsetSum} \rangle$ – Given a set of integers S and an integer t , is there a subset $S' \subseteq S$ such that $\sum S' = t$?
- $\langle \mathbf{Clique} \rangle$ – Given a graph G and an integer k , is there a clique of size k in G ?
- $\langle \mathbf{VertexCover} \rangle$ – Given a graph G and an integer k , is there a vertex cover of size k in G ?
- ...

²https://en.wikipedia.org/wiki/List_of_NP-complete_problems

1. **P**

Time Complexity of TMs

P – Polynomial Time Complexity (Tractable Problems)

2. **NP**

Time Complexity of NTMs

NP – Nondeterministic Polynomial Time Complexity

NP – Verifier-based Definition

3. **NP-complete**

Polynomial Time Reduction (\leq_P)

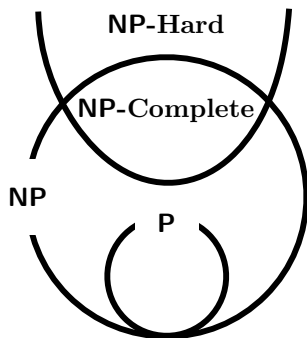
NP-complete – Hardest Problems in **NP**

<SAT> – The First **NP-complete** Problem

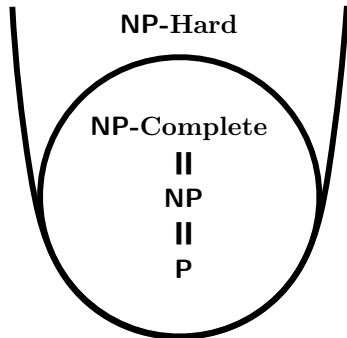
Other **NP-complete** Problems

4. Major Unsolved Problem: **P = NP?**

Major Unsolved Problem: $P = NP?$



$P \neq NP$



$P = NP$

"If $P = NP$, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in creative leaps, no fundamental gap between solving a problem and recognizing the solution once it's found."

— Scott Aaronson, UT Austin

1. **P**

Time Complexity of TMs

P – Polynomial Time Complexity (Tractable Problems)

2. **NP**

Time Complexity of NTMs

NP – Nondeterministic Polynomial Time Complexity

NP – Verifier-based Definition

3. **NP-complete**

Polynomial Time Reduction (\leq_P)

NP-complete – Hardest Problems in **NP**

<SAT> – The First **NP-complete** Problem

Other **NP-complete** Problems

4. Major Unsolved Problem: **P = NP?**

- Course Review

Jihyeok Park

jihyeok_park@korea.ac.kr

<https://plrg.korea.ac.kr>