

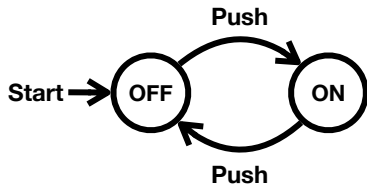
# Lecture 1 – Mathematical Preliminaries

## COSE215: Theory of Computation

Jihyeok Park



2024 Spring



### Theorem

*The current state is OFF if and only if the button is pushed **even** times.*

- Is it possible to prove it?

Let's learn **mathematical background and notation**.

## 1. Mathematical Notations

Notations in Logics

Notations in Set Theory

## 2. Inductive Proofs

Inductions on Integers

Structural Inductions

Mutual Inductions

## 3. Notations in Languages

Symbols & Words

Languages

## 1. Mathematical Notations

Notations in Logics

Notations in Set Theory

## 2. Inductive Proofs

Inductions on Integers

Structural Inductions

Mutual Inductions

## 3. Notations in Languages

Symbols & Words

Languages

Notation	Description
$A, B$	arbitrary <b>statements</b> .
$P(x)$	a <b>predicate</b> that involves a <b>variable</b> $x$ .
$A \wedge B$	the <b>conjunction</b> of $A$ and $B$ . (i.e., $A$ and $B$ ).
$A \vee B$	the <b>disjunction</b> of $A$ and $B$ . (i.e., $A$ or $B$ ).
$\neg A$	the <b>negation</b> of $A$ .

Notation	Description
$A \Rightarrow B$	the <b>implication</b> of $A$ and $B$ (i.e., if $A$ then $B$ ).
$A \Leftrightarrow B$	<b><math>A</math> if and only if (iff) <math>B</math></b> (i.e., $A \Rightarrow B \wedge B \Rightarrow A$ ).
$\forall x \in X. P(x)$	the <b>universal quantifier</b> (i.e., for all $x$ in $X$ , $P(x)$ holds).
$\exists x \in X. P(x)$	the <b>existential quantifier</b> (i.e., there exists $x$ in $X$ such that $P(x)$ holds).

- A **set** is a collection of elements, e.g.,
  - $\mathbb{N} = \{0, 1, 2, \dots\}$
  - $\{x \in \mathbb{N} \mid x \text{ is even}\} = \{0, 2, 4, 6, 8, 10, 12, \dots\}$
  - $\{x \in \mathbb{N} \mid x^2\} = \{0, 1, 4, 9, 16, 25, 36, \dots\}$
- The **empty set** is denoted by  $\emptyset$ .
- The **cardinality** of a set  $X$  is denoted by  $|X|$ .
- A **subset**  $X$  of a set  $Y$  is denoted by  $X \subseteq Y$ .

$$X \subseteq Y \iff \forall x \in X. x \in Y$$

- A **proper subset**  $X$  of a set  $Y$  is denoted by  $X \subset Y$ .

$$X \subset Y \iff X \subseteq Y \wedge X \neq Y$$

- The **union** of sets

$$\begin{aligned} X \cup Y &= \{x \mid x \in X \vee x \in Y\} \\ \bigcup \mathcal{C} &= X_1 \cup X_2 \cup \cdots \cup X_n = \{x \mid \exists X \in \mathcal{C}. x \in X\} \end{aligned}$$

where  $\mathcal{C} = \{X_1, X_2, \dots, X_n\}$ .

- The **intersection** of sets

$$\begin{aligned} X \cap Y &= \{x \mid x \in X \wedge x \in Y\} \\ \bigcap \mathcal{C} &= X_1 \cap X_2 \cap \cdots \cap X_n = \{x \mid \forall X \in \mathcal{C}. x \in X\} \end{aligned}$$

where  $\mathcal{C} = \{X_1, X_2, \dots, X_n\}$ .

- The **difference** of sets

$$X \setminus Y = \{x \mid x \in X \wedge x \notin Y\}$$



- The **complement** of a set  $X$  is denoted by  $\overline{X}$ .

$$\overline{X} = \{x \mid x \in U \wedge x \notin X\}$$

where  $U$  is the **universal set**.

- The **power set** of a set  $X$  is denoted by  $2^X$  or  $\mathcal{P}(X)$ .

$$2^X = \mathcal{P}(X) = \{Y \mid Y \subseteq X\}$$

- The **Cartesian product** of sets  $X$  and  $Y$  is denoted by  $X \times Y$ .

$$X \times Y = \{(x, y) \mid x \in X \wedge y \in Y\}$$

## 1. Mathematical Notations

Notations in Logics

Notations in Set Theory

## 2. Inductive Proofs

Inductions on Integers

Structural Inductions

Mutual Inductions

## 3. Notations in Languages

Symbols & Words

Languages

## Definition (Inductions on Integers)

Let  $P(n)$  be a predicate on integers, and if

- **(Basis Case)**  $P(k)$  holds where  $k$  is an integer, and
- **(Induction Case)** for all  $n \geq k$ ,  $P(n) \Rightarrow P(n + 1)$ ,

then  $P(i)$  holds for all  $i \geq k$ .

$P(n)$  is called **induction hypothesis**.

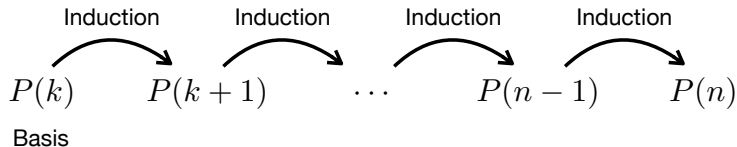
## Definition (Inductions on Integers)

Let  $P(n)$  be a predicate on integers, and if

- **(Basis Case)**  $P(k)$  holds where  $k$  is an integer, and
- **(Induction Case)** for all  $n \geq k$ ,  $P(n) \Rightarrow P(n + 1)$ ,

then  $P(i)$  holds for all  $i \geq k$ .

$P(n)$  is called **induction hypothesis**.



## Example

Prove that  $\forall n \geq 0. \sum_{i=0}^n i = \frac{n(n+1)}{2}$ .

**Proof)**

## Example

Prove that  $\forall n \geq 0. \sum_{i=0}^n i = \frac{n(n+1)}{2}$ .

## Proof)

- (Basis Case):  $0 = 0(0 + 1)/2$   $\square$

## Example

Prove that  $\forall n \geq 0. \sum_{i=0}^n i = \frac{n(n+1)}{2}$ .

### Proof)

- (Basis Case):  $0 = 0(0 + 1)/2$   $\square$
- (Induction Case): Assume that it holds for  $n$  (I.H.). Then,

$$\begin{aligned}\sum_{i=0}^{n+1} i &= (n+1) + \sum_{i=0}^n i \\ &= (n+1) + \frac{n(n+1)}{2} \quad (\because \text{I.H.}) \\ &= \frac{(n+1)(n+2)}{2} \quad \square\end{aligned}$$

## Example

Prove that  $\forall n \geq 0. \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6}$ .

**Proof)**



## Example

Prove that  $\forall n \geq 0. \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6}$ .

## Proof)

- (Basis Case):  $0^2 = 0(0+1)(2*0+1)/6$   $\square$

## Example

Prove that  $\forall n \geq 0. \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6}$ .

### Proof)

- (Basis Case):  $0^2 = 0(0+1)(2*0+1)/6$   $\square$
- (Induction Case): Assume that it holds for  $n$  (I.H.). Then,

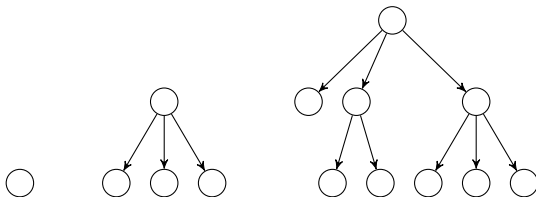
$$\begin{aligned}\sum_{i=0}^{n+1} i^2 &= (n+1)^2 + \sum_{i=0}^n i^2 \\ &= (n+1)^2 + \frac{n(n+1)(2n+1)}{6} \quad (\because \text{I.H.}) \\ &= \frac{(n+1)(n+2)(2(n+1)+1)}{6} \quad \square\end{aligned}$$

In CS, we often define somethings as **inductively-defined sets**.  
For example, we can define **trees** as follows:

## Example (Inductive Definition of Trees)

A **tree** is defined as follows:

- **(Basis Case)** A single **node**  $N$  is a tree.
- **(Induction Case)** If  $T_1, \dots, T_n$  are trees, then a graph defined with a new node  $N$  and edges from  $N$  to  $T_1, \dots, T_n$  is a tree as well.



Another example is a set of **arithmetic expressions**:

## Example (Inductive Definition of Arithmetic Expressions)

An **arithmetic expression** is defined as follows:

- **(Basis Case)** A **number** or a **variable** is an arithmetic expression.
- **(Induction Case)** If  $E$  and  $F$  are arithmetic expressions, then so are  $E+F$ ,  $E * F$ , and  $(E)$ .

42

$x$

$x + y$

$42 * x$

$(x)$

$(x * y) * z$

$(2 + x) * y$

$x * (x * y)$

$((((x))))$

## Definition (Structural Inductions)

Let  $P(x)$  be a predicate on a **inductively-defined set**  $X$ , and if

- **(Basis Case)**  $P(b_1), \dots, P(b_k)$  hold for all basis cases  $b_1, \dots, b_k$ .
- **(Induction Case)** for all  $x \in X$ ,

$$P(x_1) \wedge \dots \wedge P(x_n) \Rightarrow P(x)$$

where  $x_1, \dots, x_n$  are the **sub-structures** of  $x$ .

then  $P(x)$  holds for all  $x \in X$ .

$P(x_1), \dots, P(x_n)$  are called **induction hypotheses**.

## Definition (Structural Inductions)

Let  $P(x)$  be a predicate on a **inductively-defined set**  $X$ , and if

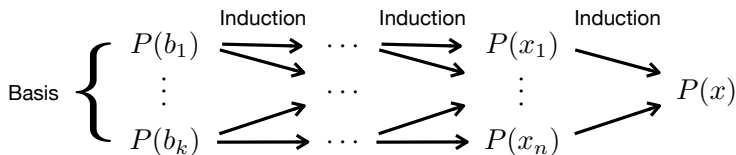
- **(Basis Case)**  $P(b_1), \dots, P(b_k)$  hold for all basis cases  $b_1, \dots, b_k$ .
- **(Induction Case)** for all  $x \in X$ ,

$$P(x_1) \wedge \dots \wedge P(x_n) \Rightarrow P(x)$$

where  $x_1, \dots, x_n$  are the **sub-structures** of  $x$ .

then  $P(x)$  holds for all  $x \in X$ .

$P(x_1), \dots, P(x_n)$  are called **induction hypotheses**.



## Example

Prove that for all tree  $T$ , the number of nodes in  $T$  is equal to the number of edges in  $T$  plus one.

**Proof)**

## Example

Prove that for all tree  $T$ , the number of nodes in  $T$  is equal to the number of edges in  $T$  plus one.

**Proof)** Let  $N(T)$  be the number of node and  $E(T)$  be the number of edges in  $T$ . Let's prove  $\forall T. N(T) = E(T) + 1$ .



## Example

Prove that for all tree  $T$ , the number of nodes in  $T$  is equal to the number of edges in  $T$  plus one.

**Proof)** Let  $N(T)$  be the number of node and  $E(T)$  be the number of edges in  $T$ . Let's prove  $\forall T. N(T) = E(T) + 1$ .

- (Basis Case):  $N(T) = 1$  and  $E(T) = 0$ .  $\square$

## Example

Prove that for all tree  $T$ , the number of nodes in  $T$  is equal to the number of edges in  $T$  plus one.

**Proof)** Let  $N(T)$  be the number of node and  $E(T)$  be the number of edges in  $T$ . Let's prove  $\forall T. N(T) = E(T) + 1$ .

- (Basis Case):  $N(T) = 1$  and  $E(T) = 0$ .  $\square$
- (Induction Case): Assume that it holds for  $T_1, \dots, T_n$  (I.H.). Then,

$$\begin{aligned} N(T) &= 1 + \sum_{i=1}^n N(T_i) \\ &= 1 + \sum_{i=1}^n (E(T_i) + 1) \quad (\because \text{I.H.}) \\ &= 1 + n + \sum_{i=1}^n E(T_i) \\ &= 1 + E(T) \quad \square \end{aligned}$$

## Example

Prove that for all arithmetic expression  $E$ , the number of left parentheses in  $E$  is equal to the number of right parentheses in  $E$ .

**Proof)**

## Example

Prove that for all arithmetic expression  $E$ , the number of left parentheses in  $E$  is equal to the number of right parentheses in  $E$ .

**Proof)** Let  $L(E)$  be the number of left parentheses and  $R(E)$  be the number of right parentheses in  $E$ . Let's prove  $\forall E. L(E) = R(E)$ .

## Example

Prove that for all arithmetic expression  $E$ , the number of left parentheses in  $E$  is equal to the number of right parentheses in  $E$ .

**Proof)** Let  $L(E)$  be the number of left parentheses and  $R(E)$  be the number of right parentheses in  $E$ . Let's prove  $\forall E. L(E) = R(E)$ .

- (Basis Case):  $L(E) = R(E) = 0$  for numbers and variables.  $\square$

## Example

Prove that for all arithmetic expression  $E$ , the number of left parentheses in  $E$  is equal to the number of right parentheses in  $E$ .

**Proof)** Let  $L(E)$  be the number of left parentheses and  $R(E)$  be the number of right parentheses in  $E$ . Let's prove  $\forall E. L(E) = R(E)$ .

- (Basis Case):  $L(E) = R(E) = 0$  for numbers and variables.  $\square$
- (Induction Case): Assume that it holds for  $E$  and  $F$  (I.H.). Then,

$$\begin{aligned} L(E+F) &= L(E) + L(F) = R(E) + R(F) & (\because \text{I.H.}) \\ &= R(E+F) & \square \end{aligned}$$

$$\begin{aligned} L(E * F) &= L(E) + L(F) = R(E) + R(F) & (\because \text{I.H.}) \\ &= R(E * F) & \square \end{aligned}$$

$$\begin{aligned} L((E)) &= L(E) + 1 = R(E) + 1 & (\because \text{I.H.}) \\ &= R((E)) & \square \end{aligned}$$

## Definition (Mutual Inductions)

Let  $P(x)$  and  $Q(x)$  are predicates on integers, and if

- **(Basis Case)**  $P(k)$  and  $Q(k)$  hold where  $k$  is an integer, and
- **(Induction Case)** for all  $n \geq k$ ,

$$P(n) \wedge Q(n) \Rightarrow P(n+1) \wedge Q(n+1)$$

then  $P(i)$  and  $Q(i)$  hold for all  $i \geq k$ .

$P(n)$  and  $Q(n)$  are called **induction hypotheses**.

## Definition (Mutual Inductions)

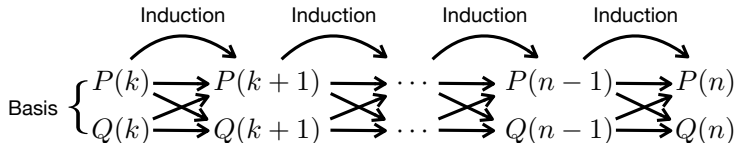
Let  $P(x)$  and  $Q(x)$  are predicates on integers, and if

- **(Basis Case)**  $P(k)$  and  $Q(k)$  hold where  $k$  is an integer, and
- **(Induction Case)** for all  $n \geq k$ ,

$$P(n) \wedge Q(n) \Rightarrow P(n+1) \wedge Q(n+1)$$

then  $P(i)$  and  $Q(i)$  hold for all  $i \geq k$ .

$P(n)$  and  $Q(n)$  are called **induction hypotheses**.





## Theorem

*The current state is OFF if and only if the button is pushed **even** times.*

**Proof)**

## Theorem

*The current state is OFF if and only if the button is pushed **even** times, and the current state is ON if and only if the button is pushed **odd** times.*

**Proof)**

## Theorem

*The current state is OFF if and only if the button is pushed **even** times, and the current state is ON if and only if the button is pushed **odd** times.*

**Proof)** Let  $S(i)$  be the current state after  $i$  times of pushing. Let's prove

$$\forall i \geq 0. S(i) = \text{OFF} \iff i \equiv 0 \pmod{2} \quad (P)$$

$$\forall i \geq 0. S(i) = \text{ON} \iff i \equiv 1 \pmod{2} \quad (Q)$$

## Theorem

*The current state is OFF if and only if the button is pushed **even** times, and the current state is ON if and only if the button is pushed **odd** times.*

**Proof)** Let  $S(i)$  be the current state after  $i$  times of pushing. Let's prove

$$\forall i \geq 0. S(i) = \text{OFF} \iff i \equiv 0 \pmod{2} \quad (P)$$

$$\forall i \geq 0. S(i) = \text{ON} \iff i \equiv 1 \pmod{2} \quad (Q)$$

- (Basis Case): Known facts:  $S(0) = \text{OFF}$  and  $0 \equiv 0 \pmod{2}$ 
  - $(P, \Rightarrow)$ :  $0 \equiv 0 \pmod{2} \implies S(0) = \text{OFF} \Rightarrow 0 \equiv 0 \pmod{2}$
  - $(P, \Leftarrow)$ :  $S(0) = \text{OFF} \implies S(0) = \text{OFF} \Leftarrow 0 \equiv 0 \pmod{2}$
  - $(Q, \Rightarrow)$ :  $\neg(S(0) = \text{ON}) \implies S(0) = \text{ON} \Rightarrow 0 \equiv 1 \pmod{2}$
  - $(Q, \Leftarrow)$ :  $\neg(0 \equiv 1 \pmod{2}) \implies S(0) = \text{ON} \Leftarrow 0 \equiv 1 \pmod{2}$

- (Induction Case): Assume that it holds for  $n$  (I.H.):

$$S(n) = \text{OFF} \iff n \equiv 0 \pmod{2} \quad (P - \text{I.H.})$$

$$S(n) = \text{ON} \iff n \equiv 1 \pmod{2} \quad (Q - \text{I.H.})$$

- $(P, \iff)$ :

$$\begin{aligned} S(n+1) = \text{OFF} &\iff S(n) = \text{ON} \\ &\iff n \equiv 1 \pmod{2} \quad (\because Q - \text{I.H.}) \\ &\iff n+1 \equiv 0 \pmod{2} \end{aligned}$$

- $(Q, \iff)$ :

$$\begin{aligned} S(n+1) = \text{ON} &\iff S(n) = \text{OFF} \\ &\iff n \equiv 0 \pmod{2} \quad (\because P - \text{I.H.}) \\ &\iff n+1 \equiv 1 \pmod{2} \end{aligned}$$

## 1. Mathematical Notations

Notations in Logics

Notations in Set Theory

## 2. Inductive Proofs

Inductions on Integers

Structural Inductions

Mutual Inductions

## 3. Notations in Languages

Symbols & Words

Languages

- We first define a finite and non-empty set of **symbols**  $\Sigma$ .

- We first define a finite and non-empty set of **symbols**  $\Sigma$ .
- A **word**  $w \in \Sigma^*$  is a sequence of symbols.



- We first define a finite and non-empty set of **symbols**  $\Sigma$ .
- A **word**  $w \in \Sigma^*$  is a sequence of symbols.
  - $\Sigma = \{0, 1\}$  – binary symbols.

$$\epsilon, 0, 1, 00, 01, 10010, \dots \in \Sigma^*$$

- We first define a finite and non-empty set of **symbols**  $\Sigma$ .
- A **word**  $w \in \Sigma^*$  is a sequence of symbols.
  - $\Sigma = \{0, 1\}$  – binary symbols.

$\epsilon, 0, 1, 00, 01, 10010, \dots \in \Sigma^*$

- $\Sigma = \{a, b, \dots, z\}$  – lowercase letters.

$\epsilon, a, b, abc, \text{hello}, \text{cs}, \text{students}, \dots \in \Sigma^*$

- We first define a finite and non-empty set of **symbols**  $\Sigma$ .
- A **word**  $w \in \Sigma^*$  is a sequence of symbols.
  - $\Sigma = \{0, 1\}$  – binary symbols.

$\epsilon, 0, 1, 00, 01, 10010, \dots \in \Sigma^*$

- $\Sigma = \{a, b, \dots, z\}$  – lowercase letters.

$\epsilon, a, b, abc, \text{hello}, \text{cs}, \text{students}, \dots \in \Sigma^*$

- $\Sigma = \{a \mid a \text{ is an Unicode character}\}$  – Unicode characters.

$\epsilon, \text{안녕하세요}, \text{こんにちは}, \star \blacksquare \blacktriangle \oplus, \dots \in \Sigma^*$

Notation	Description
$\epsilon$	the <b>empty word</b> .
$w_1 w_2$	the <b>concatenation</b> of $w_1$ and $w_2$ . ( $w_1$ is a <b>prefix</b> of $w_1 w_2$ and $w_2$ is a <b>suffix</b> of $w_1 w_2$ )
$w^R$	the <b>reverse</b> of $w$ .
$ w $	the <b>length</b> of $w$ .
$\Sigma^k$	the set of all words of length $k$ .
$\Sigma^*$	the set of all words (the <b>Kleene star</b> ). (i.e., $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \dots = \bigcup_{k=0} \Sigma^k$ )
$\Sigma^+$	the set of all words except $\epsilon$ (the <b>Kleene plus</b> ). (i.e., $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{k=1} \Sigma^k$ )

A **language**  $L \subseteq \Sigma^*$  is a specific set of words.

A **language**  $L \subseteq \Sigma^*$  is a specific set of words.

When  $\Sigma = \{0, 1\}$ , we can define the following languages:

- $L = \{\epsilon, 0, 1\}$  – the empty word, zero, and one.
- $L = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$  – all binary words.
- $L = \{0^n 1^n \mid n \geq 0\}$  – equal number of consecutive zeros and ones.
- $L = \{10, 11, 101, 111, 1011, \dots\}$  – ???

- The **union**, **intersection**, and **difference** of languages:

$$L_1 \cup L_2 \quad L_1 \cap L_2 \quad L_1 \setminus L_2$$

- The **reverse** of a language:

$$L^R = \{w^R \mid w \in L\}$$

- The **complement** of a language:

$$\bar{L} = \Sigma^* \setminus L$$

- The **concatenation** of languages:

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$$

- The **power** of a language:

$$\begin{aligned} L^0 &= \{\epsilon\} \\ L^n &= L^{n-1}L \quad (n \geq 1) \end{aligned}$$

- The **Kleene star** of a language:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots = \bigcup_{n \geq 0} L^n$$

- The **Kleene plus** of a language:

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \dots = \bigcup_{n \geq 1} L^n$$



## 1. Mathematical Notations

Notations in Logics

Notations in Set Theory

## 2. Inductive Proofs

Inductions on Integers

Structural Inductions

Mutual Inductions

## 3. Notations in Languages

Symbols & Words

Languages

- Basic Introduction of Scala

Jihyeok Park

[jihyeok\\_park@korea.ac.kr](mailto:jihyeok_park@korea.ac.kr)

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