

# ***Chap. 1 Automata: The Methods and Madness***

*Automata theory*

*abstract computing device or “machines”*

***Historical reviews***

1. 1930's      *Gödel, Turing, Church, ...*

*Boundary between what a **computing machine** could do  
and what it could **not** do*

***Turing machine***(Chap. 8) ***computable vs uncomputable***

***Terminate***(Chap. 9) ***decidable vs undecidable***

2. 1940' ~ 1950's      ***finite automata***(Chap. 2)

*simpler computing machine*

***regular expressions***(Chap. 3), ***regular languages***(Chap. 4)

한글 모아쓰기 ***finite automata***(1978)    *project*

3. late 1950's N. Chomsky

formal “**grammar**”

close relationships between **grammars** and **automata**

**context-free grammars**(Chap. 5), **pushdown automata**(Chap. 6)

**context-free languages**(Chap. 7)

**Left/Right parsers of context-free grammars**(supplement)

4. 1969 Cook

A boundary between what a computer could solve **efficiently** or not

**intractable problem**, “**NP-completeness**”(Chap. 10)

other class of problems(Chap. 11)

0. proofs, languages(Chap. 1)

reviews on mathematics(not in the text)

automata, grammars, languages, problems, and programs

## 1.2 Deductive Proof

Read [한글강의노트](#) or [supplement TP](#) for 1.2, 1.3, and 1.4!!!

*proof*

*personal feeling about the truth of statement*

*step-by-step formal proof*

*deductive proof*

*a sequence of true statements from **hypothesis** to **conclusion***

*Deductive proof of “if  $H$ , then  $C$ ”*

*We say conclusion  $C$  is **deduced** from hypothesis  $H$*

*Theorem 1.3                      inductive proof*

*Theorem 1.4                      deductive proof*

***modus ponens***

*$H$ , “if  $H$ , then  $C$ ”  $\Rightarrow C$*

## Way of Saying “If-Then”

“if  $H$ , then  $C$ ”

1.  $H$  implies  $C$ .
2.  $H$  only if  $C$ .
3.  $C$  if  $H$ .
4. Whenever  $H$  holds,  $C$  follows.
5.  $H \Rightarrow C$ .

## If-and-only-if Statements

“ $A$  if and only if  $B$ ”, “ $A$  iff  $B$ ”, “ $A$  is equivalent to  $B$ ”

$A \Leftrightarrow B$ ,  $A \equiv B$

1. if part: “if  $B$ , then  $A$ ”
2. only if part: “if  $A$ , then  $B$ ”

### 1.3.1 Proving Equivalence About Sets

Two sets  $E$  and  $F$  are equivalent”,  $E = F$ .

1.  $E \subseteq F$ , if  $x \in E$ , then  $x \in F$ .
2.  $F \subseteq E$ , if  $x \in F$ , then  $x \in E$ .

### 1.3.2 The Contrapositive

“if not  $C$ , then not  $H$ ”

Four cases

$H$	$C$	$H \rightarrow C$	$\bar{C} \rightarrow \bar{H}$	$C \rightarrow H$	$H \leftrightarrow C$
$T$	$T$	$T$	$T$	$T$	$T$
$T$	$F$	$F$	$F$	$T$	$F$
$F$	$T$	$T$	$T$	$F$	$F$
$F$	$F$	$T$	$T$	$T$	$T$

*contrapositive    converse    equivalence*

### ***1.3.3 Proof by Contradiction***

*“H and not C implies false”*

### ***1.3.4 Counterexample***

*To prove a statement is false.*

*disproof*

*observation*

*induction*

## 1.4 Inductive proof

### *Inductive(Recursive) definition of natural numbers*

**Basis:** 0 is a natural number.

**Induction:** If  $n$  is a natural number,  
then  $n+1$  is *also* a natural number.

### *Inductive proof on natural numbers*

*To prove a predicate  $p(n)$  is true for all natural number  $n(n \geq 0)$ ,*

**basis**      *prove  $p(0)$  is true,*

**induction**      *assume  $p(n)$  is true, prove  $p(n+1)$  is true.*

***Induction principle on natural numbers***

***basis***                    *prove  $p(i)$ ,*  
***induction***                *prove  $p(n)$  implies  $p(n+1)$  for all  $n \geq i$ ,*  
***conclusion***                 *$p(n)$  for all  $n \geq i$ .*

***General induction principle on natural numbers***

***basis***                    *prove  $p(i), p(i+1), \dots p(i+j)$  for some  $j \geq 0$ ,*  
***induction***                *prove  $p(n)$  implies  $p(n+j+1)$  for all  $n \geq i$ ,*  
***conclusion***                 *$p(n)$  for all  $n \geq i$ .*



### 1.4.3 Structural Induction

**Structural recursive definition of a set  $X$ .**

**(finite) basis**  $b_1, b_2, \dots, b_k \in X$ .

**(finite) recursion** If  $x_1, x_2, \dots, x_n \in X$ , then  $f(x_1, x_2, \dots, x_n) \in X$ .

**Example 1.19** A rooted tree  $T = (V, E, r)$  is defined as follows:

i)  $V$  is a set of vertices,

ii)  $E$  is a set of  $n$  edges where  $(u, v) \in E$  denotes an edge from the vertex  $u$  to the vertex  $v$ , and

iii)  $r \in V$  is a root vertex of the rooted tree  $T$

**basis**  $(\{v\}, \{\}, v)$  is a rooted tree

**recursion** If  $T_1=(V_1, E_1, r_1)$ ,  $T_2=(V_2, E_2, r_2)$ , ...,  $T_k=(V_k, E_k, r_k)$  are rooted trees, Then  $(V, E, r)$  is a rooted tree

where  $V = V_1 \cup V_2 \cup \dots \cup V_k \cup \{r\}$ ,

$E = E_1 \cup E_2 \cup \dots \cup E_k \cup \{(r, r_i) \mid 1 \leq \forall i \leq k\}$ , and

$$r \notin V_1 \cup V_2 \cup \dots \cup V_k.$$

*Example 1.20(expression)*

**basis** number and letter(variable) are expression.

**recursion**  $exp + exp$ ,  $exp - exp$ ,  $exp \times exp$ ,  $exp \div exp$ , and  $(exp)$   
are  $exp$ .

i) What are  $f$  and  $X$ ?

ii) Are the definition of  $X$  and the defined set  $X$  finite or infinite?

**Structural recursive proof on the set  $X$ .**

To prove  $p(X) \equiv \forall x \in X, p(x)$ .

**basis**  $p(b_1), p(b_2), \dots, p(b_k)$ .

**induction** If  $p(x), p(x_2), \dots, p(x_n)$ , then  $p(f(x_1, x_2, \dots, x_n))$ .

*Example 1.21 (rooted tree), Example 1.22(expression)*

What is  $p$ ?



**Empty string**

a string of **zero(no)** occurrence of symbols, denoted as,  $\epsilon$ (epsilon).  
 The **empty string** may be chosen from **any** alphabet whatsoever.

**Length of a string**

the **number of positions** for symbols in the string

If  $w$  is a string, then the **length** of the string  $w$  is denoted as  $|w|$ .

$$|01101| = 5, |111| = 3$$

$$|\epsilon| = 0$$

**Concatenation of strings**

Let  $x$  and  $y$  be strings, the **concatenation** of string, denoted  $xy$ ,

If  $x = a_1a_2 \dots a_n$ ,  $1 \leq \forall i \leq n$ ,  $a_i$  is a symbol, and

$y = b_1b_2 \dots b_m$ ,  $1 \leq \forall j \leq m$ ,  $b_j$  is a symbol, then

$$xy = a_1a_2 \dots a_nb_1b_2 \dots b_m$$

$$|xy| = |x| + |y| = n + m$$

**Concatenation is associative but noncommutative.**

Let  $x, y, z$  be a string over some alphabet  $\Sigma$ .

$$(xy)z = x(yz) \qquad xy \neq yx$$

**Empty string is the identity element for the concatenation**  
for any string  $w$ ,  $x\varepsilon = \varepsilon x = x$ .

### **Power of an alphabet**

We define  $\Sigma^k$  to be the set of strings of length  $k$   
each of whose symbol is in  $\Sigma$ .

Note that  $\Sigma^0 = \{\varepsilon\}$ , regardless of what alphabet  $\Sigma$  is.

Example) If  $\Sigma = \{0, 1\}$ , then

$$\Sigma^0 = \{\varepsilon\} \quad \Sigma^1 = \{0, 1\} \quad \Sigma^2 = \{00, 01, 10, 11\}$$

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

$$|\Sigma^k| = |\Sigma|^k.$$

The set of **all** strings over an alphabet  $\Sigma$ , denoted as,  $\Sigma^*$ ,

$$\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{i \in N_0} \Sigma^i.$$

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots = \bigcup_{i \in N_1} \Sigma^i.$$

$$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$$

$\Sigma^*$  contains **any** string over  $\Sigma$ .

$x$  is a **string** over  $\Sigma \Leftrightarrow x \in \Sigma^*$ .

$\Sigma^*$  is the **universe(type)** of string  $x$ .

### 1.5.3 Language

A set of strings *all* of which are chosen from some  $\Sigma^*$ ,  
where  $\Sigma$  is a particular alphabet.

*Language is a set of strings.*

If  $\Sigma$  is an alphabet, and  $L \subseteq \Sigma^*$ , then  $L$  is a *language over*  $\Sigma$ .  
 $\Sigma^*$ , is the *universe* for *any* languages over  $\Sigma$ .

$L$  is a *language over*  $\Sigma \Leftrightarrow L \subseteq \Sigma^*$  or  $L \in 2^{\Sigma^*}$ .

#### Example of languages

1.  $\{\epsilon, 01, 0011, 000111, \dots\} = \{0^n 1^n \mid n \geq 0\}$
2. The set of all strings of 0's and 1's equal numbers of each.  
 $\{\epsilon, 01, 10, 0011, 0101, 0110, 1001, \dots\}$
3. The set of binary numbers(non leading zero) whose value is prime.  
 $\{10, 11, 101, 111, 1011, \dots\}$

4.  $\Sigma^*$  is a language for alphabet  $\Sigma$ .
5.  $\emptyset$ , the **empty language**, is a language over **any** alphabet.
6.  $\{\varepsilon\}$ , the language consisting of only **empty string**,  
is also a language over **any** alphabet.  
 $\emptyset \neq \{\varepsilon\}$

### **Four terminologies**

	<i>element</i>	<i>set</i>
<i>single</i>	<i>symbol</i> $a \in \Sigma$	<b>alphabet</b> $\Sigma$
<u><i>sequence</i></u>	<u><i>string</i></u> $x \in \Sigma^*$	<u><b>language</b></u> $L \subseteq \Sigma^*$ or $L \in 2^{\Sigma^*}$ .

**A symbol is a string of length one.**

**An alphabet is a language whose elements are of length one.**



*Type conventions*

$a, b, c, \dots, 0, 1, \dots \in \Sigma$	<i>symbols</i>
$u, v, w, x, y, z, \dots \in \Sigma^*$	<i>strings</i>
$L, S, T, \dots \subseteq \Sigma^*$	<i>languages</i>
$\Sigma, V, \dots$	<i>alphabets, vocabularies</i>

$\Sigma^*$  is countable.

*strings are countable*

But is  $2^{\Sigma^*}$  uncountable.

*languages are uncountable*

## 1.5.4 Problems

### *membership problem*

*a question of deciding whether a given string is a member of particular language*

*Let  $\Sigma$  be an alphabet and  $L$  be a language over  $\Sigma^*$ , ( $L \subseteq \Sigma^*$ )*

*Given  $w \in \Sigma^*$ , decide whether  $w \in L$  or not.*

### *decision problem*

*a question of deciding yes or no.*

### *Example 1.26*

*membership problem vs. decision problem*