Digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Nonzero_digit = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Letter = A | B | C | ··· | Z | a | b | c | d | ··· | z

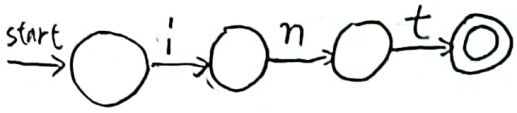
Blank는 한칸의 공백을 의미

편의상 위와 같이 정의하고 시작한다.

○ Variable type

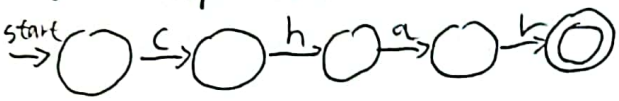
〈INT〉

regular expression: int



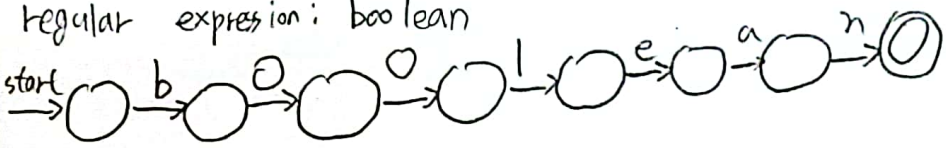
〈CHAR〉

regular expression: char



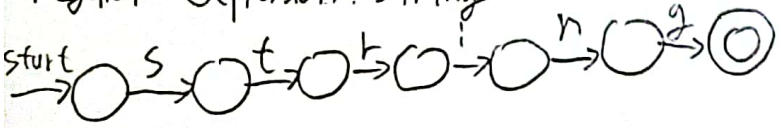
〈BOOL〉

regular expression: boolean



〈STR〉

regular expression: string

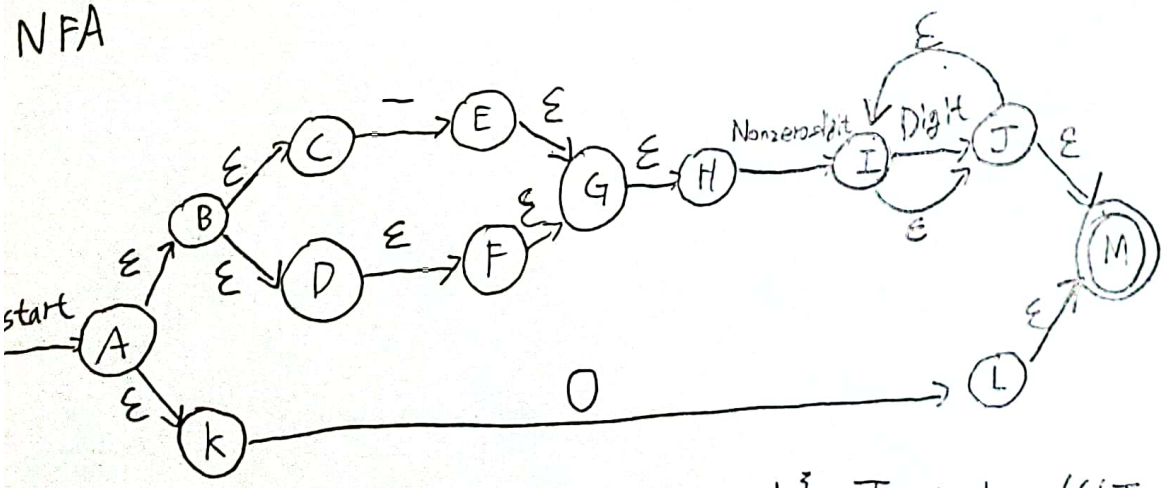


○ Signed Integer : 0 이거나 0으로 시작하지않는 양, 혹은 음의 정수 ex) 0, 51, -10, -100

〈INTEGER〉

regular expression: $((-|\varepsilon)\ Nonzero\_digit\ (Digit)^*)\ |\ 0$

NFA



$T_0 = \varepsilon\text{-closure}(A) = \{A, B, C, D, F, G, H, K\}$

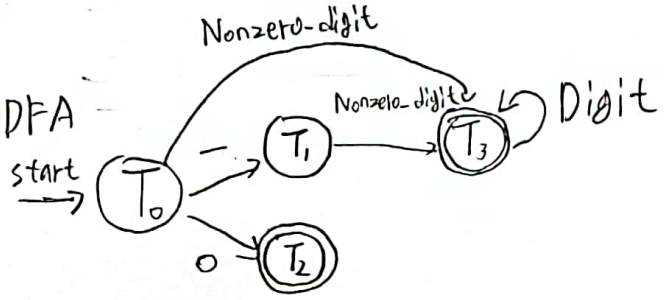
$T_1 = \varepsilon\text{-closure}(\delta(T_0, -)) = \{E, G, H\}$

$T_2 = \varepsilon\text{-closure}(\delta(T_0, 0)) = \{L, M\}$

$T_3 = \varepsilon\text{-closure}(\delta(T_1, Nonzero\_digit)) = \{I, J, M\}$

	—	0	Nonzero_digit	Digit
T_0	T_1	T_2	T_3	∅
T_1	∅	∅	T_3	∅
T_2	∅	∅	∅	∅
T_3	∅	∅	∅	T_3

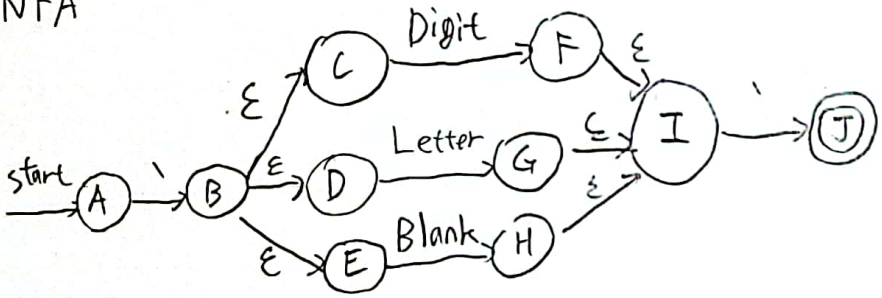
DFA



o single character & 한 글자의 숫자, 알파벳 공백이 ' ' 사이에 있는

<CHARACTER> Regular expression: `(Letter | Digit | Blank)`

NFA



$T_0 = \varepsilon\text{-closure}(A) = \{A\}$

$T_1 = \varepsilon\text{-closure}(\delta(T_0, \text{'}')) = \{B, C, D, E\}$

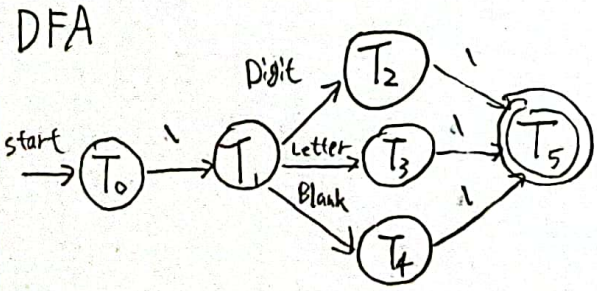
$T_2 = \varepsilon\text{-closure}(\delta(T_1, \text{Digit})) = \{F, I\}$

$T_3 = \varepsilon\text{-closure}(\delta(T_1, \text{Letter})) = \{G, I\}$

$T_4 = \varepsilon\text{-closure}(\delta(T_1, \text{Blank})) = \{H, I\}$

$T_5 = \varepsilon\text{-closure}(\delta(T_2, \text{'}')) = \{J\}$

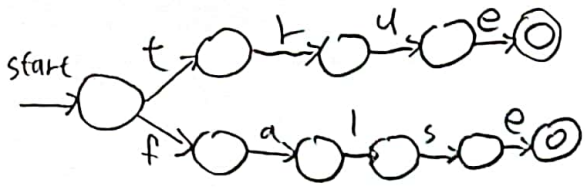
DFA



○ Boolean string : true 혹은 false

<BOOLEAN>

regular expression : (true) | (false)

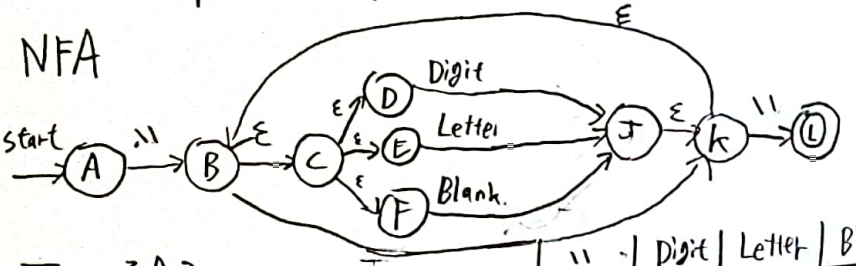


○ Literal string : Digit, Letter, Blank로 이루어진 문자열이 " " 사이에 있음.

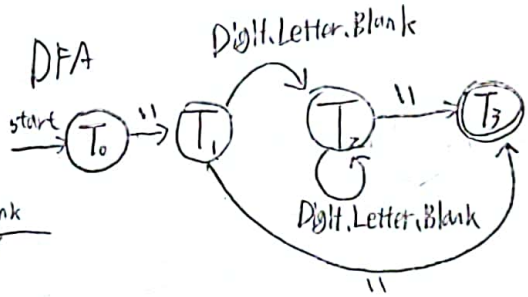
<STRING>

regular expression : " (Digit | Letter | Blank)* "

NFA



DFA



$T_0 = \{A\}$

$T_1 = \{B, C, D, E, F, K\}$

$T_2 = \{J, K, B, C, D, E, F\}$

$T_3 = \{L\}$

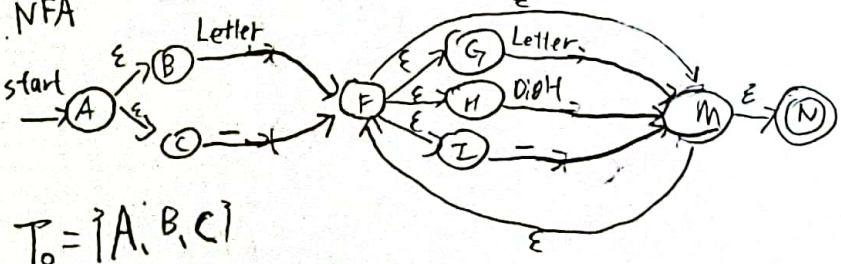
	"	Digit	Letter	Blank
T_0	T_1	∅	∅	∅
T_1	T_3	T_2	T_2	T_2
T_2	T_3	T_2	T_2	T_2
T_3	∅	∅	∅	∅

● An identifier of variables and function.

<IDENTIFIER>

regular expression : (Letter | _) (Letter | Digit | _)*

NFA

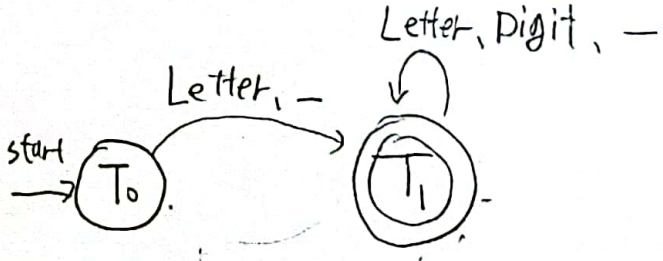


$T_0 = \{A, B, C\}$

$T_1 = \{F, G, H, IM, N\}$

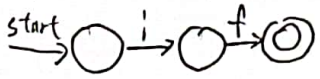
DFA

	Letter	Digit	_
T_0	T_1	∅	T_1
T_1	T_1	T_1	T_1

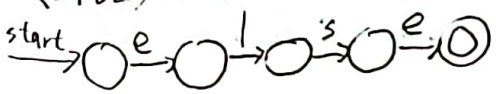


○ Key words for special statements

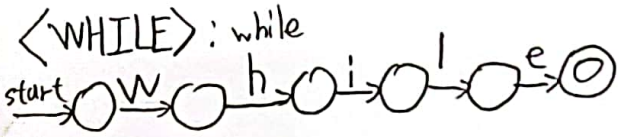
<IF> : if



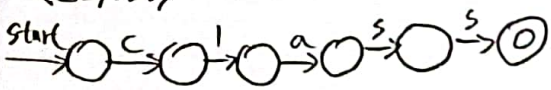
<ELSE> : else



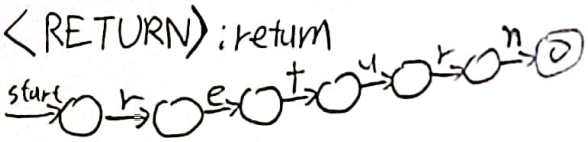
<WHILE> : while



<CLASS> : class

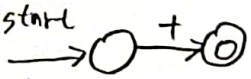


<RETURN> : return



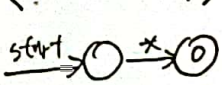
○ Arithmetic operators : 사칙연산자

<PLUS> : +

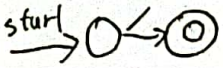


<MINUS> : -

<MULTIFLY> : *

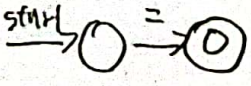


<DIVISION> : /



○ Assignment operator

<ASSIGNMENT> : =

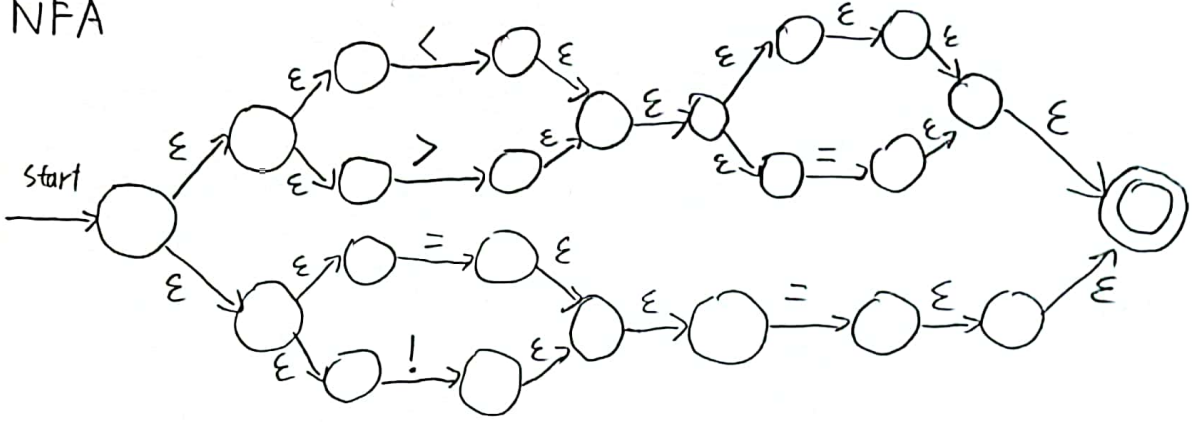


∝Comparison operators

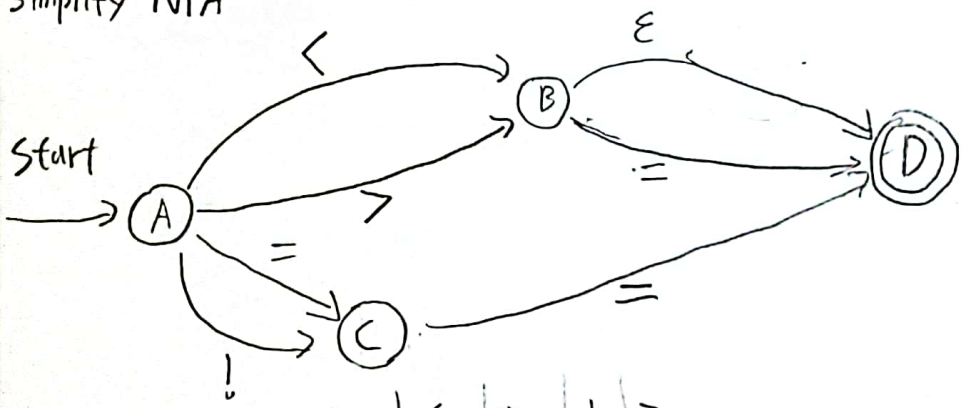
<COMPARE> : <、>、<=、>=、==、!=

regular expression : $((<|>)(\varepsilon|=))|((=|!)=)$

NFA



Simplify NFA



$T_0 = \{A\}$

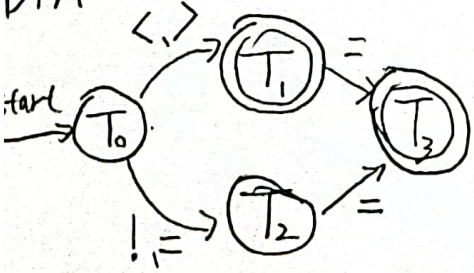
$T_1 = \{B, D\}$

$T_2 = \{C\}$

$T_3 = \{D\}$

	<	>	!	=
T_0	T_1	T_1	T_2	T_2
T_1	∅	∅	∅	T_3
T_2	∅	∅	∅	T_3
T_3	∅	∅	∅	∅

DFA

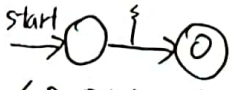


- A terminating symbol of statements

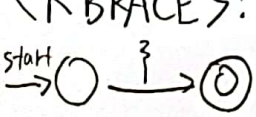
<SEMI> : ;

- A pair of symbols for defining area/scope of variables and function

<LBRACE> : {

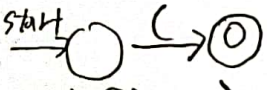


<RBRACE> : }

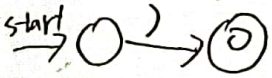


- A pair of symbols for indicating a function/statement

<LPAREN> : (

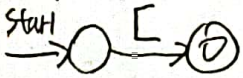


<RPAREN> :)

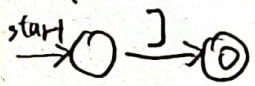


- A pair of symbols for using an array

<LBRACKET> : [

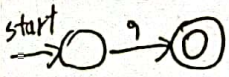


- <RBRACKET> :]



- A symbol for separating input arguments in functions

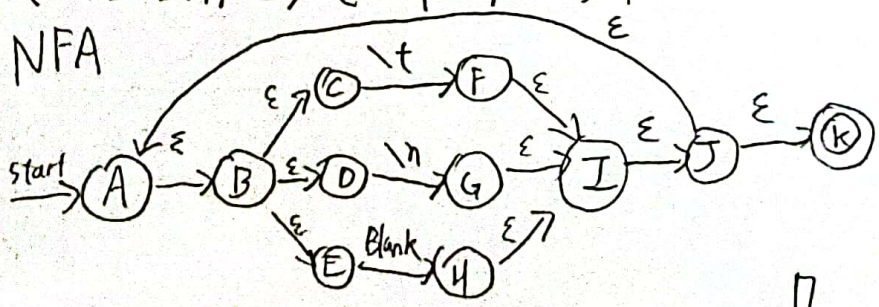
<COMMA> : ,



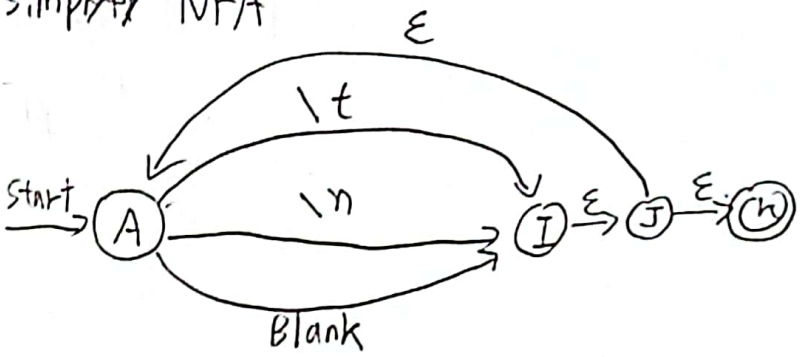
- Whitespaces

<WHITESPACE> (\t | \n | Blank)+

NFA



Simplify NFA



$T_0 = \{A\}$

$T_1 = \{I, J, K, A\}$

	\n	\t	Blank
T_0	T_1	T_1	T_1
T_1	T_1	T_1	T_1

DFA