1. Define Tokens

Type : int | float | bool | string

Id : identifier

keyword : if | else | while | for | return

A-O : + | - | * | /

B-O : << | >> | & | |

Assign O : =

C-O : < | > | == | != | <= | >=

terminating : ;

LBarket : {

RBarket : }

Lparen : (

Rparen :)

seperating : ,

whitespaces : tot | \n | blanks

- INT :
- FLOAT :
- BOOL : TRUE | FALSE
- STRING :

2. Regular Expression

1) Type : (int | float | bool | string)

- INT : $(0 | (-|\varepsilon)(\text{non-zero digit})(\text{digit}^*))$
- float : (INT) . $(0 | (\text{non-zero digit}) | \text{digit}^*(\text{non-zero digit}))$
- String : $(\text{“}(\text{digit} | \text{letter} | \text{whitespace})^*\text{”})$

- bool : (true | false)

2) Id
- Id : (Letter | _)(Letter | Digit | _)*

3) keywords.
- keywords : (if | else | while | for | return)

4) A-O
- A-O : (+ | - | * | /)

5) B-O
- B-O : (<< | >> | & | |)

6) Assign O
- Assign O : (=)

7) C-O
- C-O : (<(= | ε) | >(= | ε) | (= | !) =)

8) terminate
- terminate : ;

9) LBarket
- LBarket : ({)

10) RBarket
- RBarket : (})

11) Lparen
- Lparen : [(]

12) Rparen
- Rparen : [)]

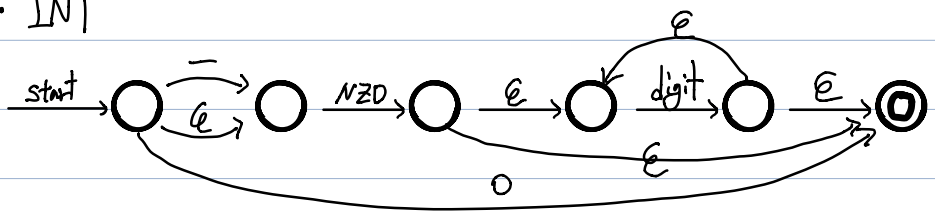
13) Seperating
- Seperating : [,]

14) whitespaces
- whitespaces : [wt | wn | blank]*

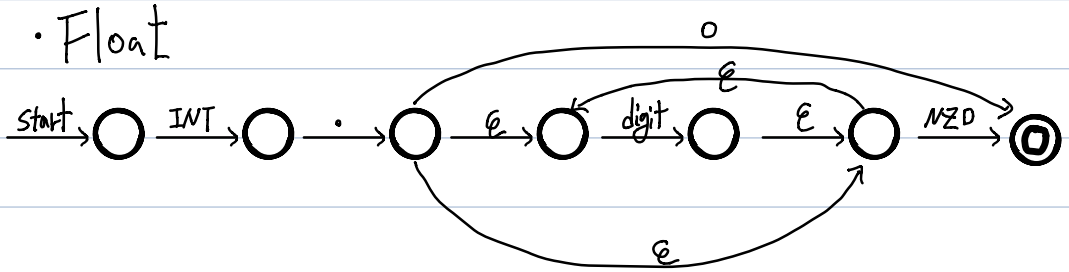
3. NFA

Non-zero digit = NZD

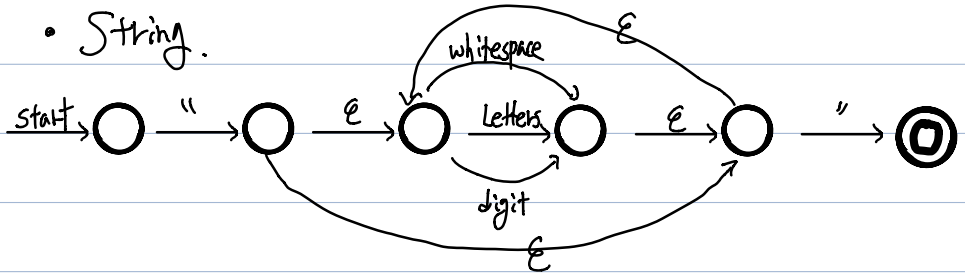
- INT



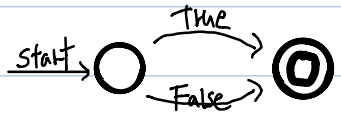
- Float



- String.

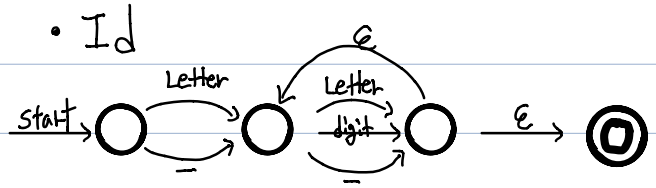


- bool



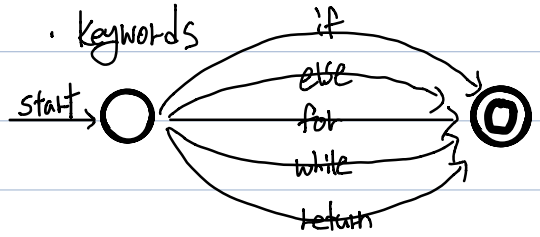
2) Id

• Id



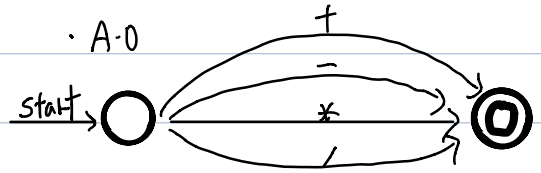
3) keywords

• keywords



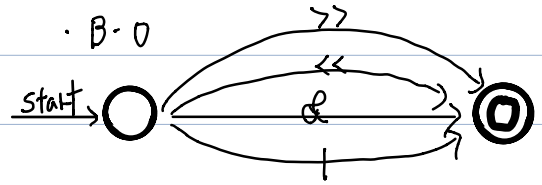
4) A·O

• A·O



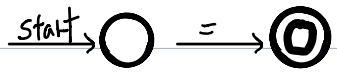
5) B·O

• B·O

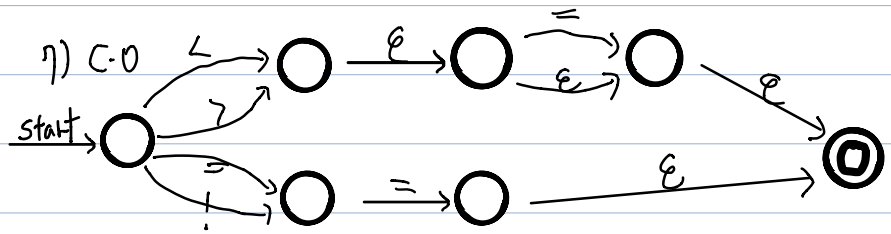


6) Assign O

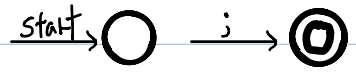
• Assign O



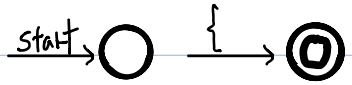
7) C·O



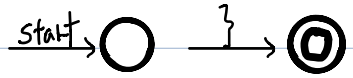
8) terminate



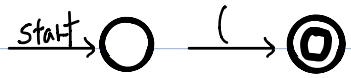
9) LBarket



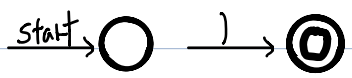
10) RBarket



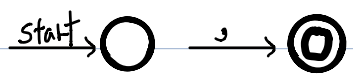
11) LParen



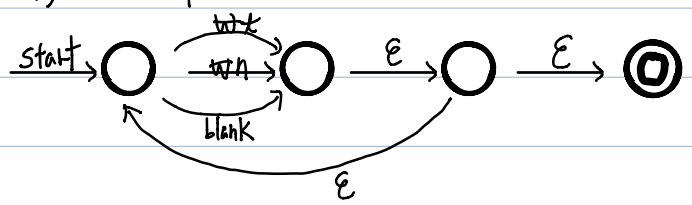
12) RParen



13) Seperating

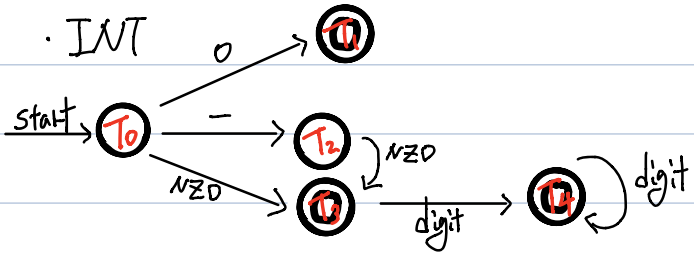


14) whitespaces



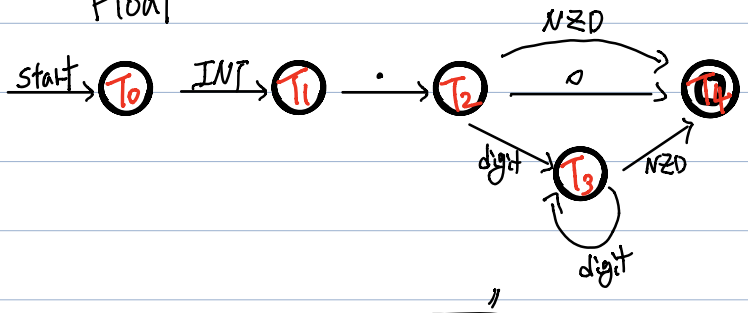
4. DFA

· INT



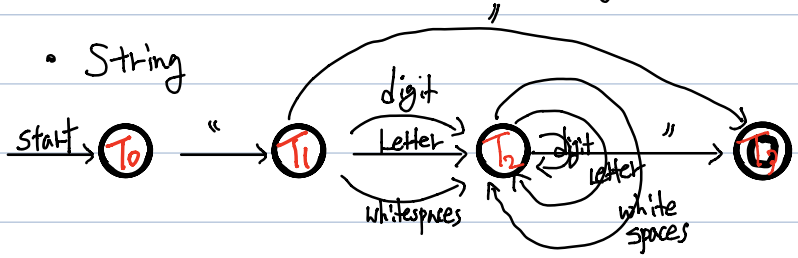
	0	–	NZD	digit
T_0	T_1	T_2	T_3	X
T_1	X	X	X	X
T_2	X	X	T_3	X
T_3	X	X	X	T_4
T_4	X	X	X	T_4

· Float



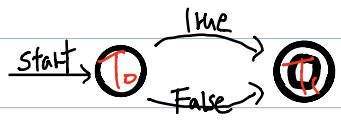
	INT	.	NZD	0	digit
T_0	T_1	X	X	X	X
T_1	X	T_2	X	X	X
T_2	X	X	T_4	T_4	T_3
T_3	X	X	T_4	X	T_3
T_4	X	X	X	X	X

· String



	"	digit	Letter	white spaces
T_0	T_1	X	X	X
T_1	T_3	T_2	T_2	T_2
T_2	T_3	T_2	T_2	T_2
T_3	X	X	X	X

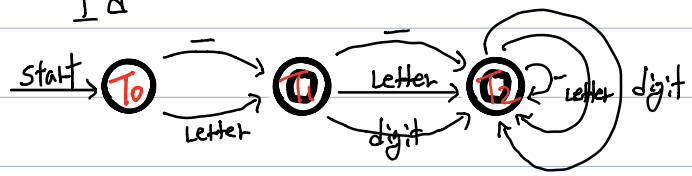
· bool



	TRUE	FALSE
T_0	T_1	T_1
T_1	X	X

2) Id

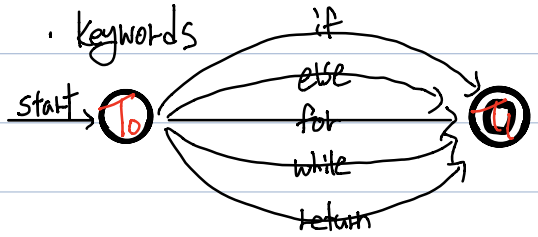
• Id



	_	letter	digit
T_0	T_1	T_1	x
T_1	T_2	T_2	T_2
T_2	T_3	T_3	T_3

3) keywords

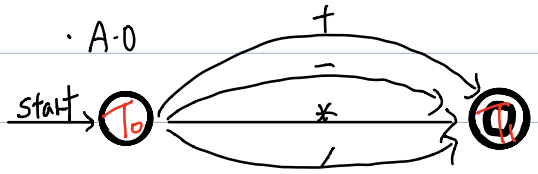
• keywords



	if	else	for	while	return
T_0	T_1	T_1	T_1	T_1	T_1
T_1	x	x	x	x	x

4) A·O

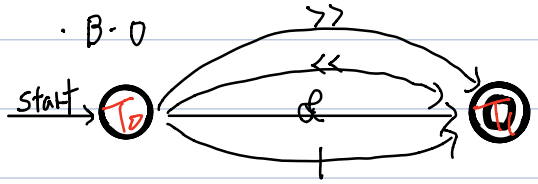
• A·O



	+	−	*	/
T_0	T_1	T_1	T_1	T_1
T_1	x	x	x	x

5) B·O

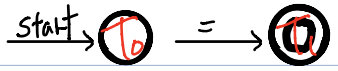
• B·O



	>>	<<	&	|
T_0	T_1	T_1	T_1	T_1
T_1	x	x	x	x

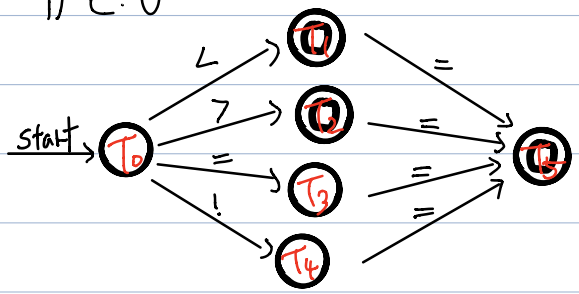
6) Assign O

• Assign O



	=
T_0	T_1
T_1	x

7) C·O



	<	>	=	!
T_0	T_1	T_2	T_3	T_4
T_1	x	x	T_5	x
T_2	x	x	T_5	x
T_3	x	x	T_5	x
T_4	x	x	T_5	x
T_5	x	x	x	x

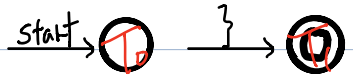
8) terminate

	;
T_0	T_1
T_1	X

9) LBarket

	{
T_0	T_1
T_1	X

10) RBarket

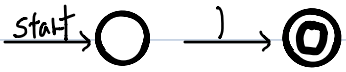


	}
T_0	T_1
T_1	X

11) LParen

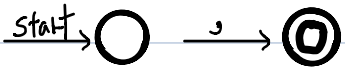
	(
T_0	T_1
T_1	X

12) RParen



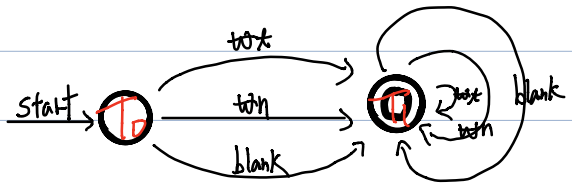
	)
T_0	T_1
T_1	X

13) Seperating



	,
T_0	T_1
T_1	X

14) Whitespaces.



	~~tab~~	~~blank~~	~~blank~~
T_0	T_1	T_1	T_1
T_1	T_1	T_1	T_1