Compiler for Simplified C language

1.Regular Expression (Lexical Analyzer)

Sub regular expressions

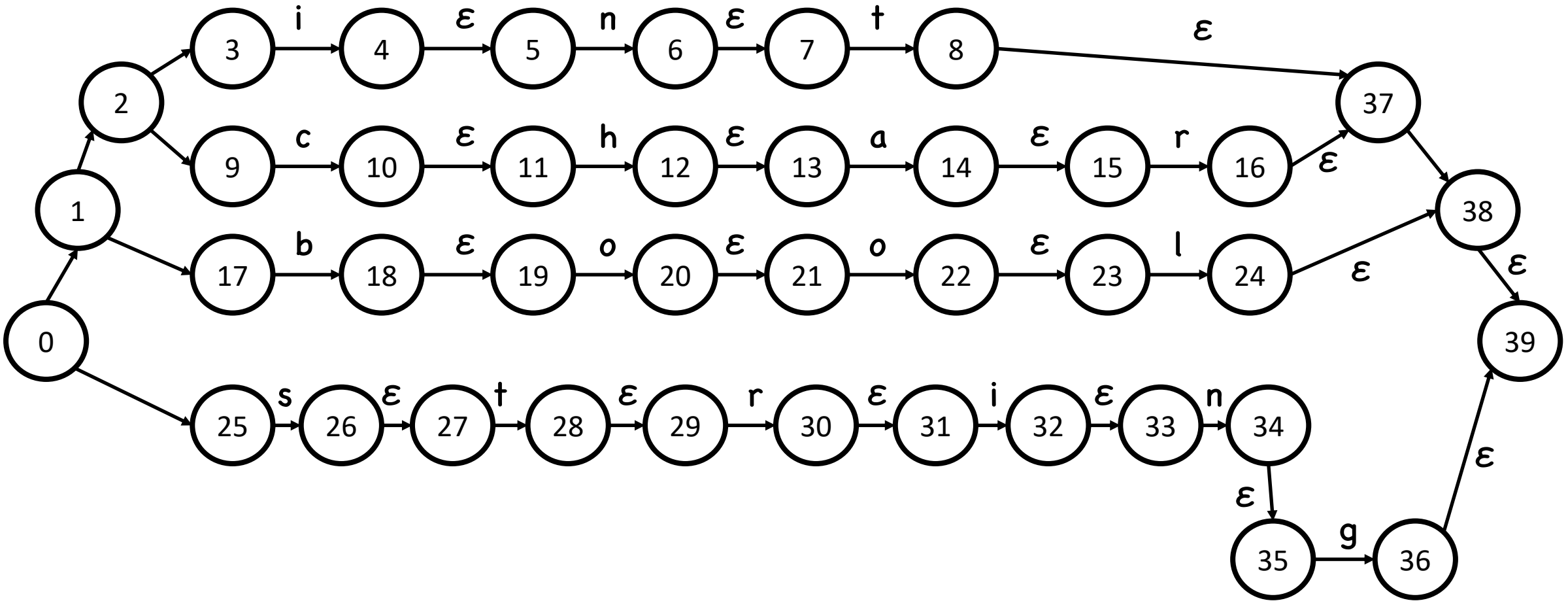
- digit : 0|1|2|3|4|5|6|7|8|9

- digit_ : 1|2|3|4|5|6|7|8|9

- Letter : a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

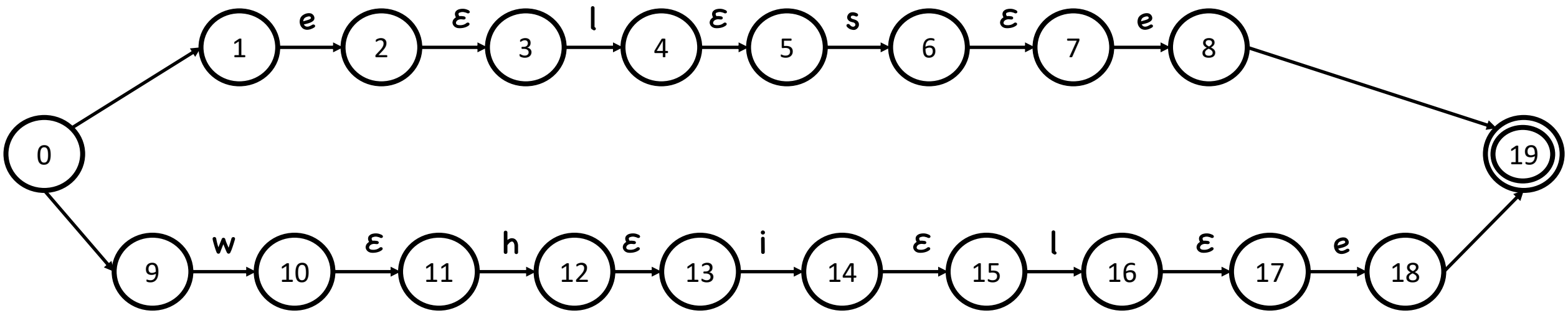
vtype

- alphabet: [a, b, c, h, i, l, n, o, r, s, t]
- regular expression: int|char|bool|string



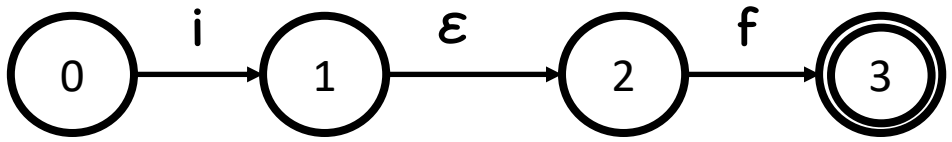
boolstr

- alphabet: [a, e, f, l, r, s, t, u]
- regular expression: true|false



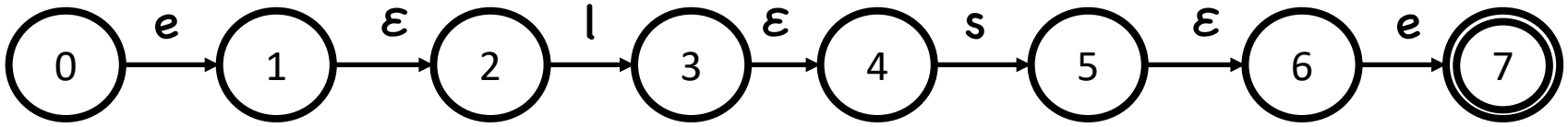
if

- alphabet: [f, i]
- regular expression: if



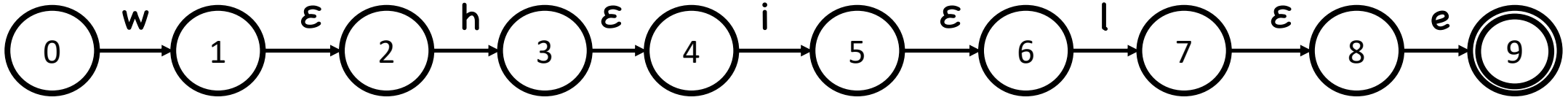
else

- alphabet: [e, l, s]
- regular expression: else



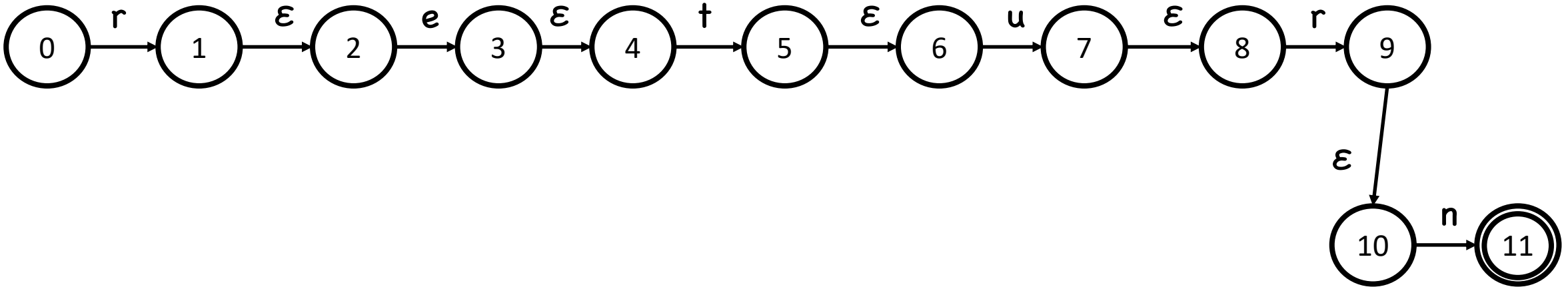
while

- alphabet: [e, h, i, l, w]
- regular expression: while



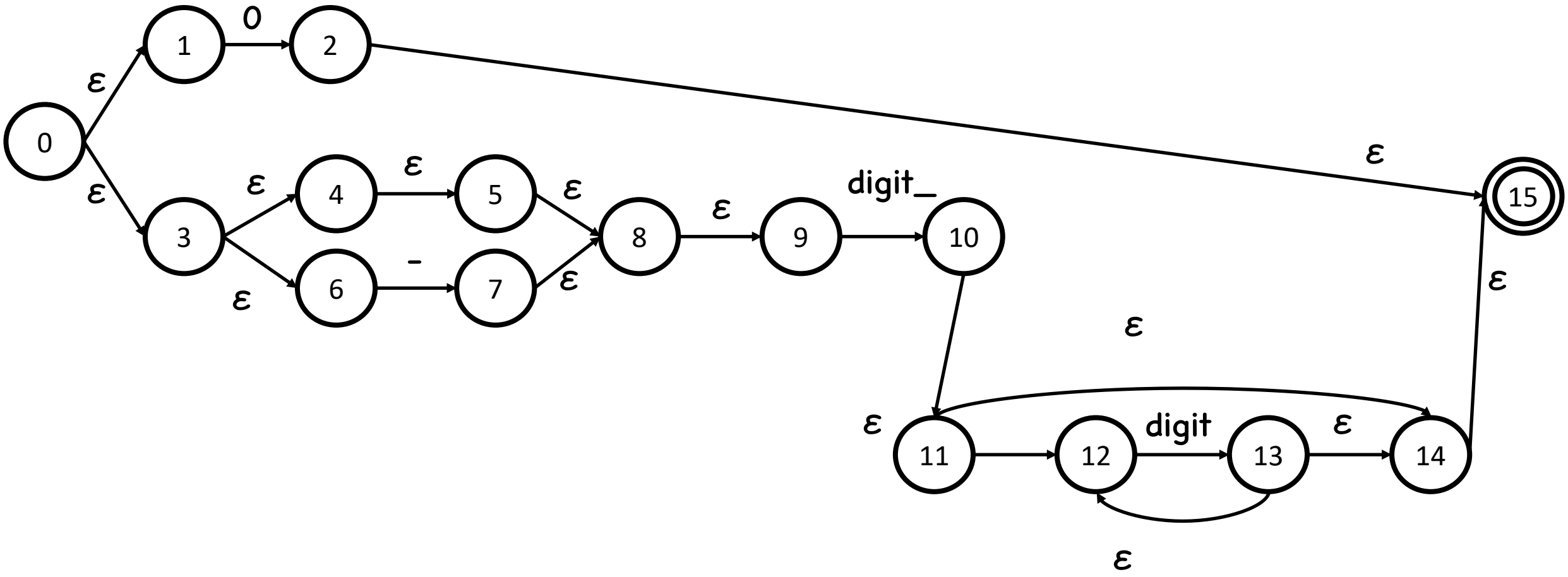
return

- alphabet: [e, n, r, t, u]
- regular expression: return



num

- alphabet: [-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε]
- regular expression: 0|(ε|-)(digit_)(digit)*
- NFA



num

ε-closure(0) = {0, 1, 3, 4, 5, 6, 8, 9} = T0

ε-closure(δ(T0, 0)) = {2, 15} = T1

ε-closure(δ(T0, digit_)) = {10, 11, 14, 15} = T2

ε-closure(δ(T0, -)) = {7, 8, 9} = T3

ε-closure(δ(T0, digit)) = ∅

ε-closure(δ(T1, 0)) = ∅

ε-closure(δ(T1, digit_)) = ∅

ε-closure(δ(T1, -)) = ∅

ε-closure(δ(T1, digit)) = ∅

ε-closure(δ(T2, 0)) = ∅

ε-closure(δ(T2, digit_)) = ∅

ε-closure(δ(T2, -)) = ∅

ε-closure(δ(T2, digit)) = {12, 13, 14 ,15} = T4

num

ε-closure(δ(T3, 0)) = ∅

ε-closure(δ(T3, digit_)) = T2

ε-closure(δ(T3, -)) = ∅

ε-closure(δ(T3, digit)) = ∅

ε-closure(δ(T4, 0)) = ∅

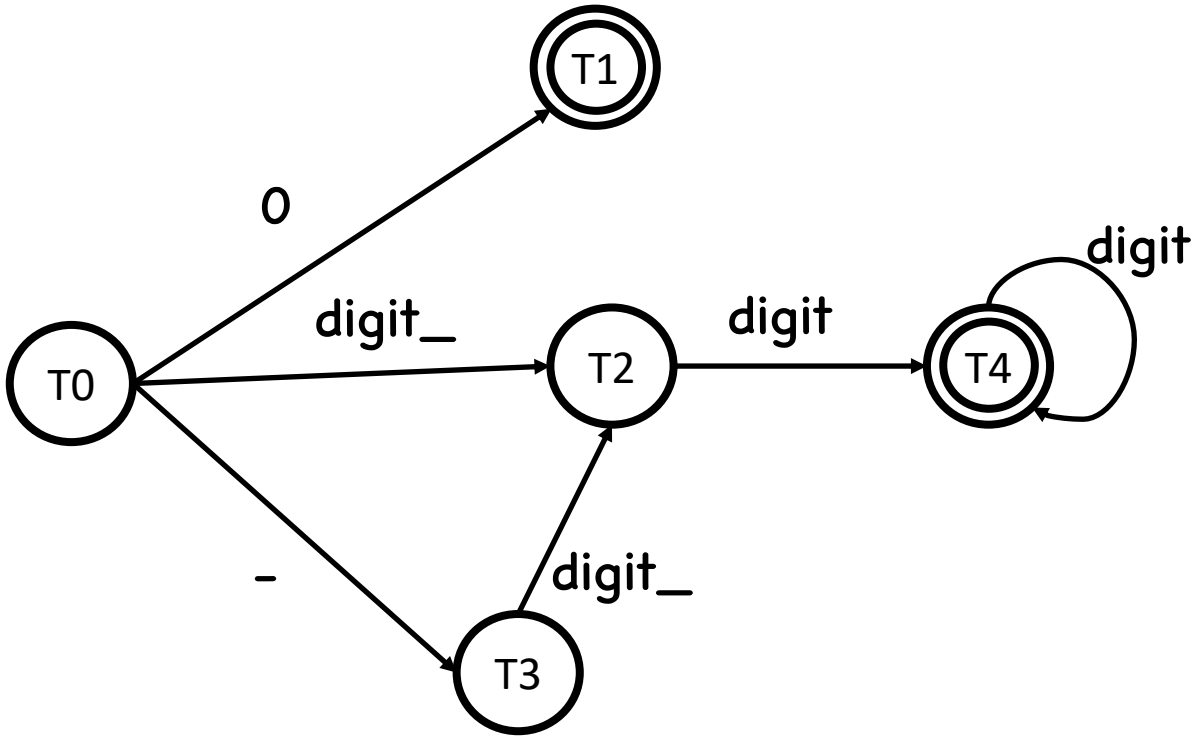
ε-closure(δ(T4, digit_)) = ∅

ε-closure(δ(T4, -)) = ∅

ε-closure(δ(T4, digit)) = T4

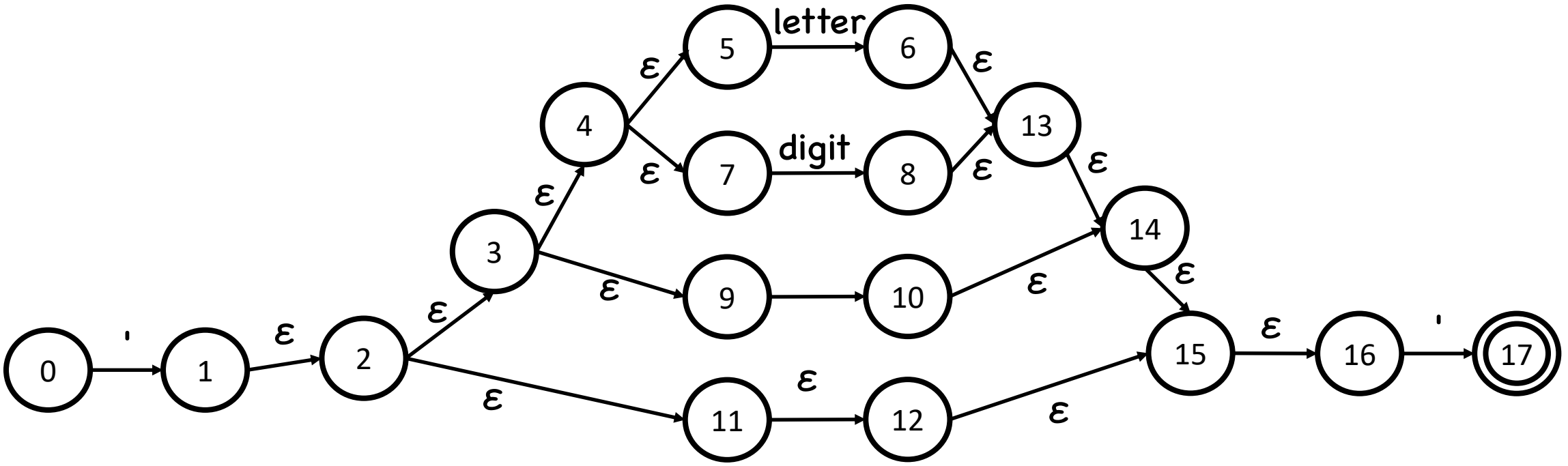
num

- DFA



character

- alphabet: [ε, , `, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
- regular expression: 'letter| |ε|digit'
- NFA



character

ε-closure(0) = {0} = T0

ε-closure(δ(T0, ')) = {0, 1, 2, 3, 4, 5, 7, 9, 11, 12, 15, 16} = T1

ε-closure(δ(T0, letter)) = ∅

ε-closure(δ(T0, digit)) = ∅

ε-closure(δ(T0,)) = ∅

ε-closure(δ(T1, ')) = {17} = T2

ε-closure(δ(T1, letter)) = {6, 13, 14, 15, 16, 17} = T3

ε-closure(δ(T1, digit)) = {8, 13, 14, 15, 16, 17} = T4

ε-closure(δ(T1,)) = {10, 14, 15, 16, 17} = T5

ε-closure(δ(T2, ')) = ∅

ε-closure(δ(T2, letter)) = ∅

ε-closure(δ(T2, digit)) = ∅

ε-closure(δ(T2,)) = ∅

character

ε-closure(δ(T3, ')) = {17} = T2

ε-closure(δ(T3, letter)) = ∅

ε-closure(δ(T3, digit)) = ∅

ε-closure(δ(T3,)) = ∅

ε-closure(δ(T4, ')) = {17} = T2

ε-closure(δ(T4, letter)) = ∅

ε-closure(δ(T4, digit)) = ∅

ε-closure(δ(T4,)) = ∅

ε-closure(δ(T5, ')) = {17} = T2

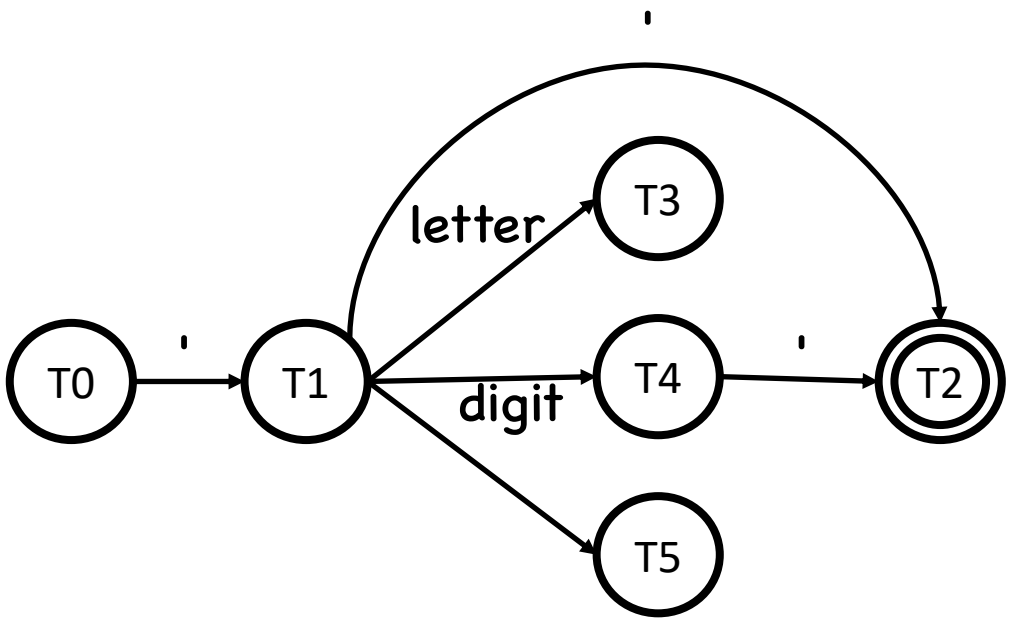
ε-closure(δ(T5, letter)) = ∅

ε-closure(δ(T5, digit)) = ∅

ε-closure(δ(T5,)) = ∅

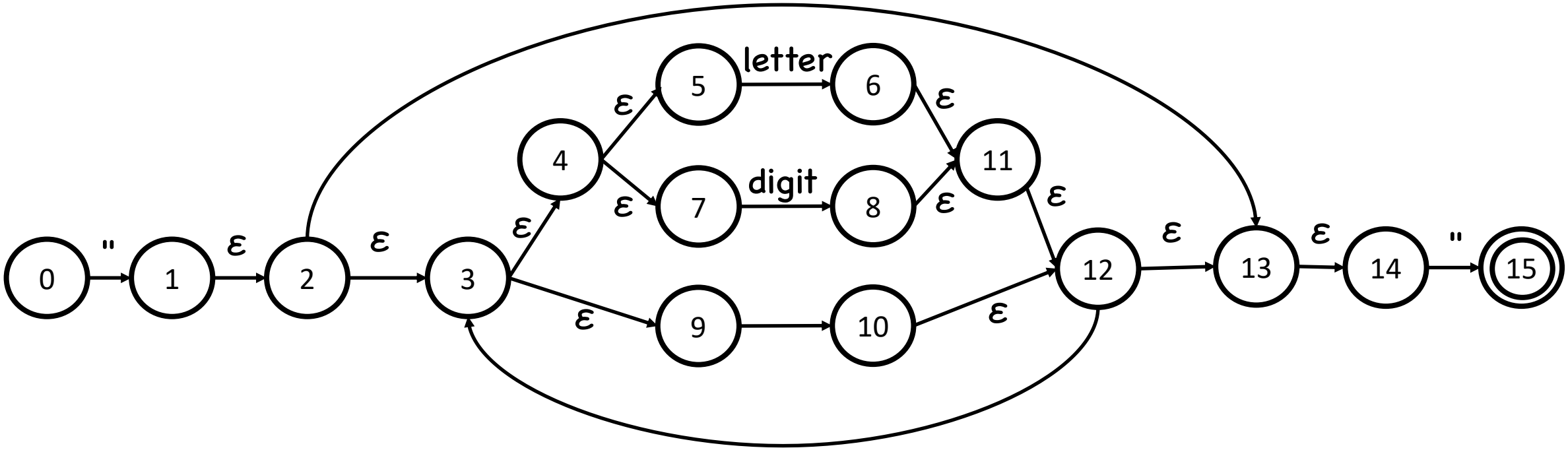
character

- DFA



string

- alphabet: [, ", 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
- regular expression: "(letter| |digit)*"



string

ε-closure(0) = {0} = T0

ε-closure(δ(T0, ")) = {1, 2, 3, 4, 5, 7, 9, 13, 14} = T1

ε-closure(δ(T1, letter)) = {3, 4, 5, 6, 7, 9, 11, 12} = T2
ε-closure(δ(T1, digit)) = {3, 4, 5, 7, 8, 9, 11, 12} = T3
ε-closure(δ(T1,)) = {3, 4, 5, 7, 9, 10, 12} = T4
ε-closure(δ(T1, ")) = {15} = T5

ε-closure(δ(T2, letter)) = {3, 4, 5, 6, 7, 9, 11, 12} = T2
ε-closure(δ(T2, digit)) = {3, 4, 5, 7, 8, 9, 11, 12} = T3
ε-closure(δ(T2,)) = {3, 4, 5, 7, 9, 10, 12} = T4
ε-closure(δ(T2, ")) = {15} = T5

string

ε-closure(δ(T3, letter)) = {3, 4, 5, 6, 7, 9, 11, 12} = T2

ε-closure(δ(T3, digit)) = {3, 4, 5, 7, 8, 9, 11, 12} = T3

ε-closure(δ(T3,)) = {3, 4, 5, 7, 9, 10, 12} = T4

ε-closure(δ(T3, ")) = {15} = T5

ε-closure(δ(T4, letter)) = {3, 4, 5, 6, 7, 9, 11, 12} = T2

ε-closure(δ(T4, digit)) = {3, 4, 5, 7, 8, 9, 11, 12} = T3

ε-closure(δ(T4,)) = {3, 4, 5, 7, 9, 10, 12} = T4

ε-closure(δ(T4, ")) = {15} = T5

ε-closure(δ(T5, letter)) = {3, 4, 5, 6, 7, 9, 11, 12} = ∅

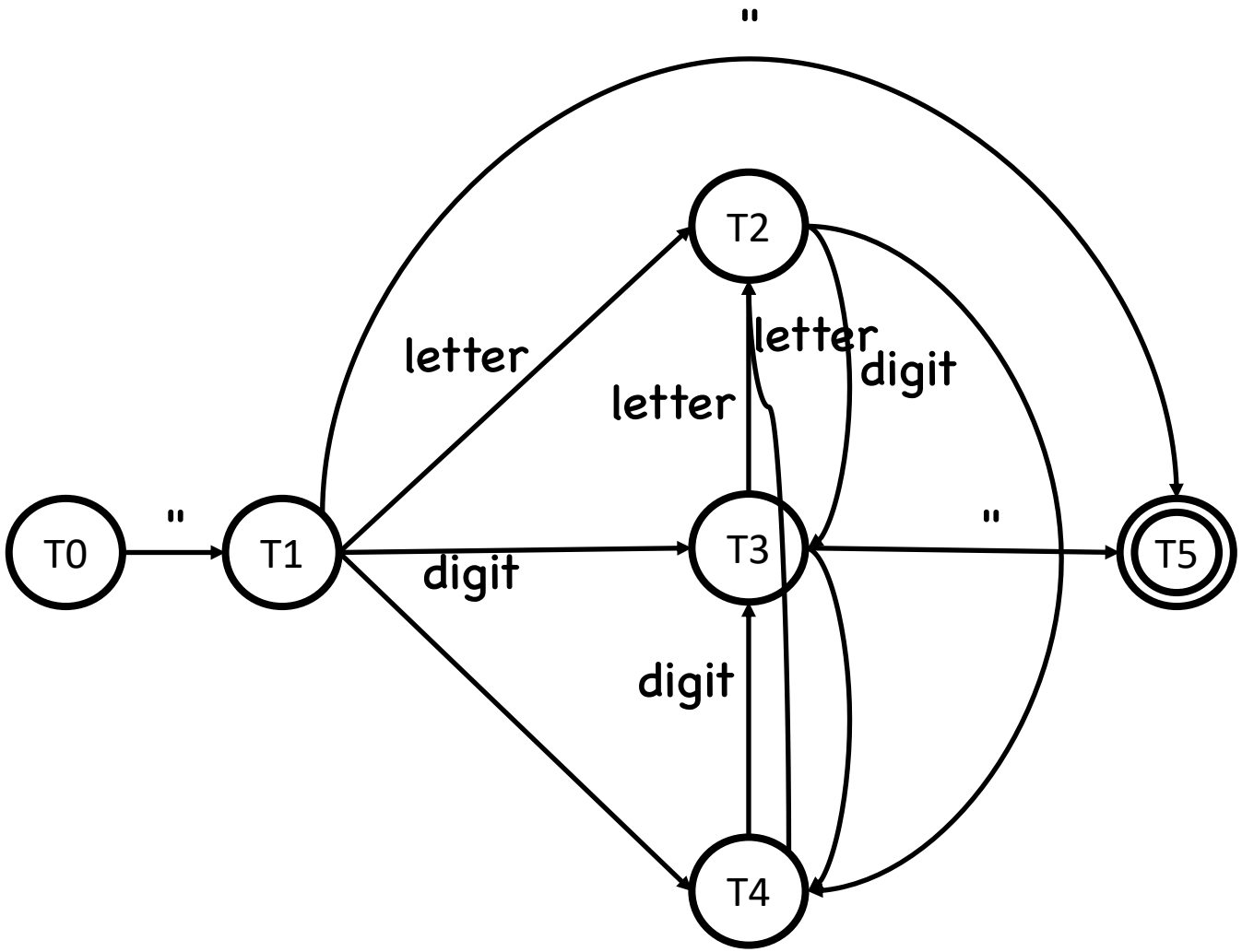
ε-closure(δ(T5, digit)) = {3, 4, 5, 7, 8, 9, 11, 12} = ∅

ε-closure(δ(T5,)) = {3, 4, 5, 7, 9, 10, 12} = ∅

ε-closure(δ(T5, ")) = {15} = ∅

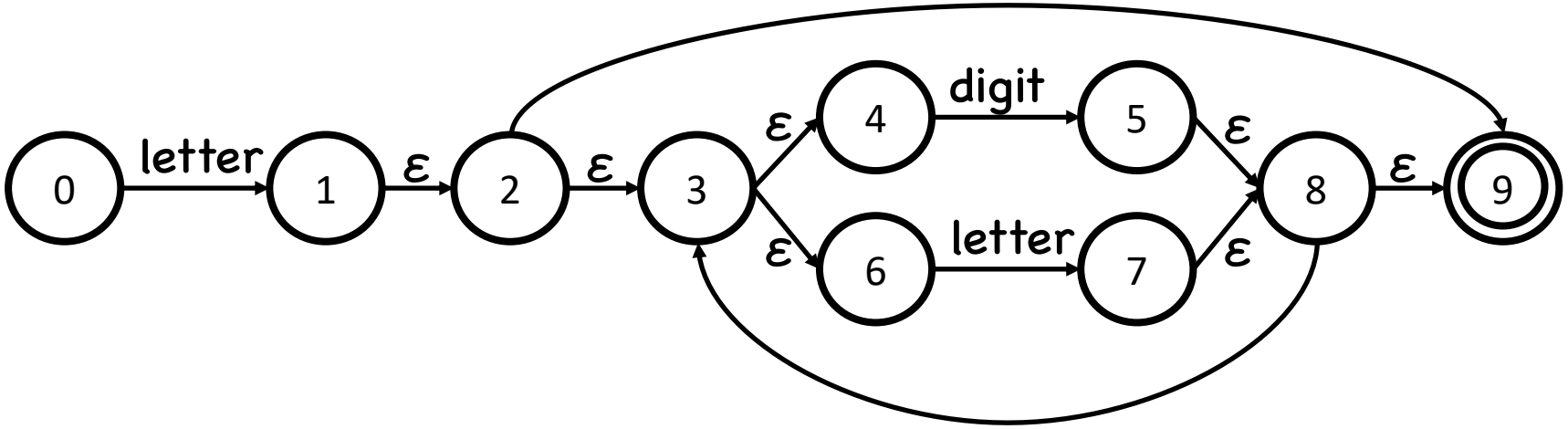
string

- DFA



id

- alphabet: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
- regular expression: letter(digit|letter)*



id

ε-closure(0) = {0} = T0

ε-closure(δ(T0, letter)) = {1, 2, 3, 4, 6, 9} = T1

ε-closure(δ(T0, digit)) = ∅

ε-closure(δ(T1, letter)) = {3, 4, 6, 7, 8, 9} = T2

ε-closure(δ(T1, digit)) = {3, 4, 5, 6, 8, 9} = T3

ε-closure(δ(T2, letter)) = {3, 4, 6, 7, 8, 9} = T2

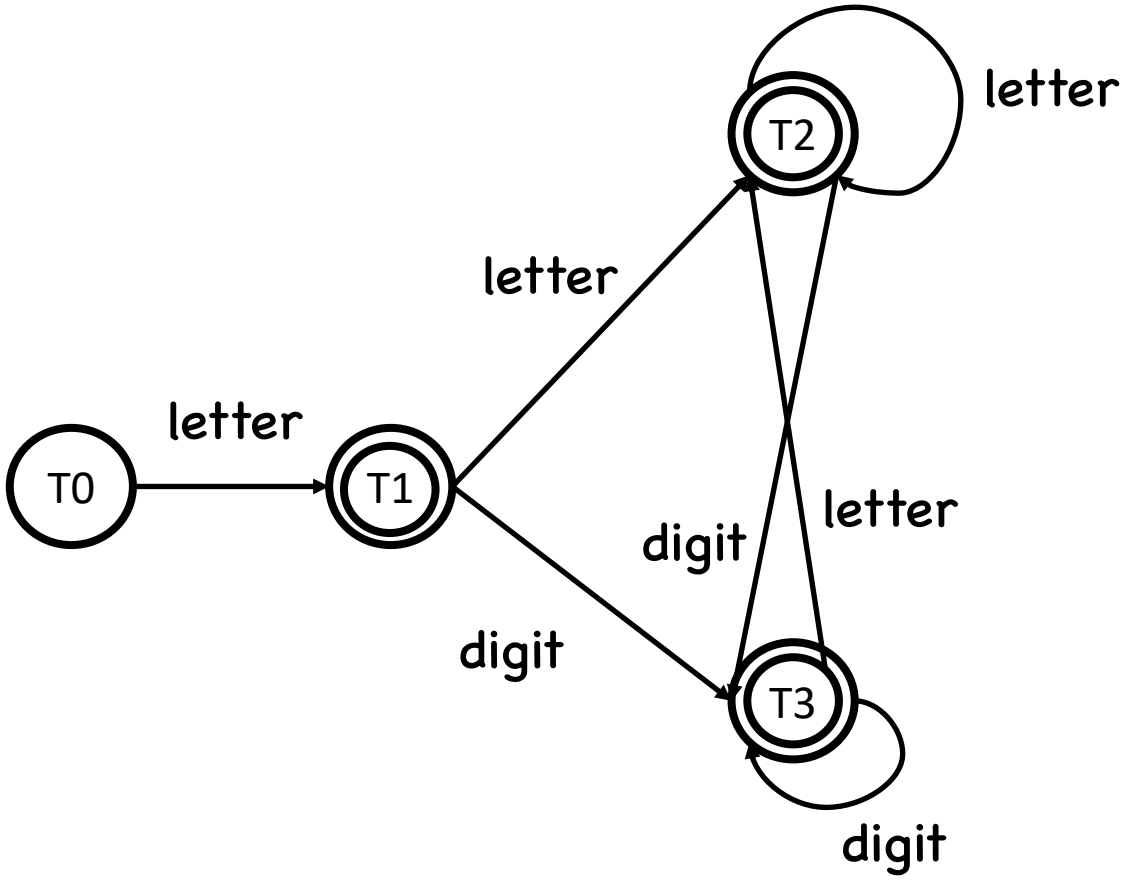
ε-closure(δ(T2, digit)) = {3, 4, 5, 6, 8, 9} = T3

ε-closure(δ(T3, letter)) = {3, 4, 6, 7, 8, 9} = T2

ε-closure(δ(T3, digit)) = {3, 4, 5, 6, 8, 9} = T3

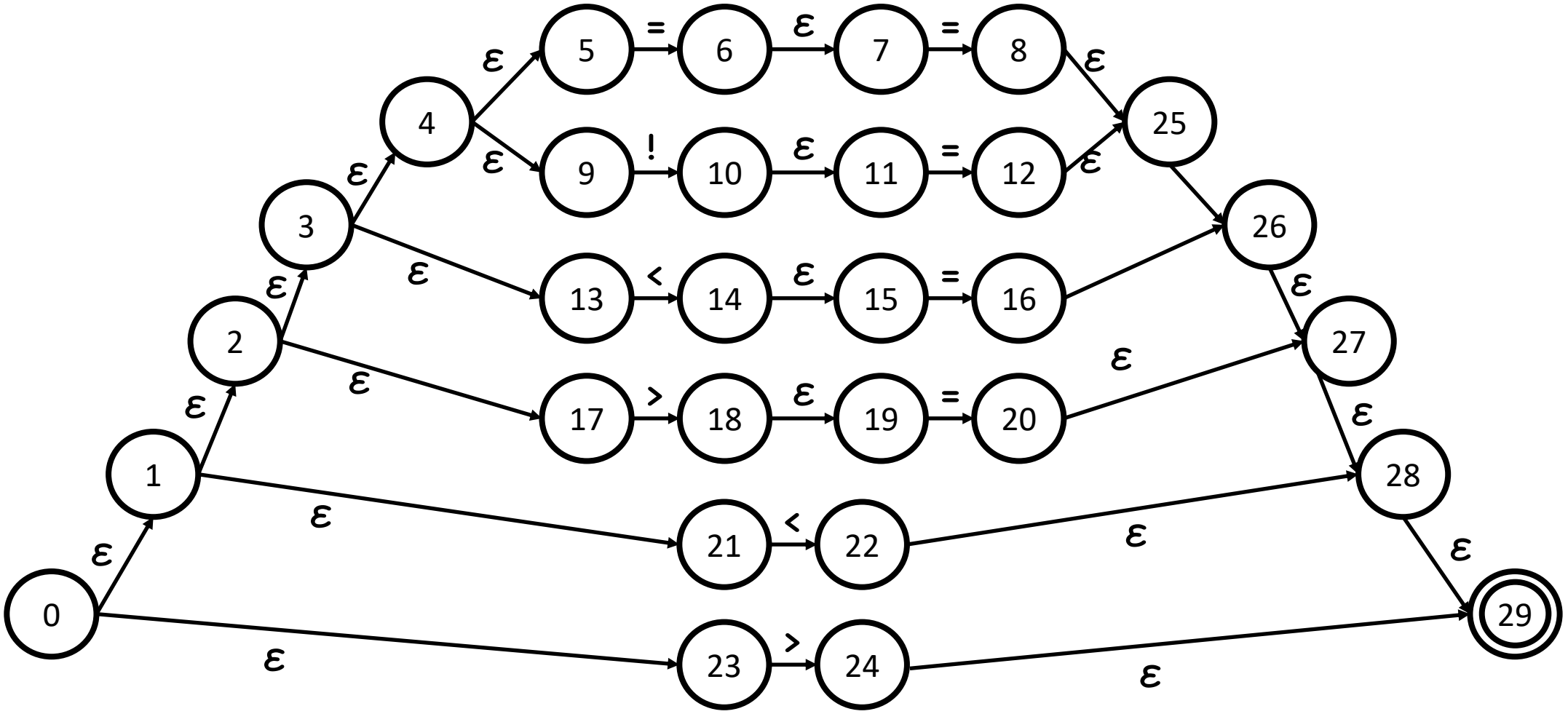
id

- DFA



comp

- alphabet: [!, <, =, >]
- regular expression: ==|!=|<|>|<=|>=



comp

ε-closure(0) = {0, 1, 2, 3, 4, 5, 9, 13, 17, 21, 23} = T0

ε-closure(δ(T0, =)) = {6, 7} = T1

ε-closure(δ(T0, !)) = {10, 11} = T2

ε-closure(δ(T0, <)) = {14, 15, 22, 28, 29} = T3

ε-closure(δ(T0, >)) = {18, 19, 24, 29} = T4

ε-closure(δ(T1, =)) = {8, 25, 26, 27, 28, 29} = T5

ε-closure(δ(T1, !)) = ∅

ε-closure(δ(T1, <)) = ∅

ε-closure(δ(T1, >)) = ∅

ε-closure(δ(T2, =)) ={12, 25, 26, 27, 28, 29} = T6

ε-closure(δ(T2, !)) = ∅

ε-closure(δ(T2, <)) = ∅

ε-closure(δ(T2, >)) = ∅

comp

ε-closure(δ(T3, =)) = {16, 26, 27, 28, 29} = T7

ε-closure(δ(T3, !)) = ∅

ε-closure(δ(T3, <)) = ∅

ε-closure(δ(T3, >)) = ∅

ε-closure(δ(T4, =)) = {20, 27, 28, 29} = T8

ε-closure(δ(T4, !)) = ∅

ε-closure(δ(T4, <)) = ∅

ε-closure(δ(T4, >)) = ∅

ε-closure(δ(T5, =)) = ∅

ε-closure(δ(T5, !)) = ∅

ε-closure(δ(T5, <)) = ∅

ε-closure(δ(T5, >)) = ∅

comp

ε-closure(δ(T6, =)) = ∅

ε-closure(δ(T6, !)) = ∅

ε-closure(δ(T6, <)) = ∅

ε-closure(δ(T6, >)) = ∅

ε-closure(δ(T7, =)) = ∅

ε-closure(δ(T7, !)) = ∅

ε-closure(δ(T7, <)) = ∅

ε-closure(δ(T7, >)) = ∅

ε-closure(δ(T8, =)) = ∅

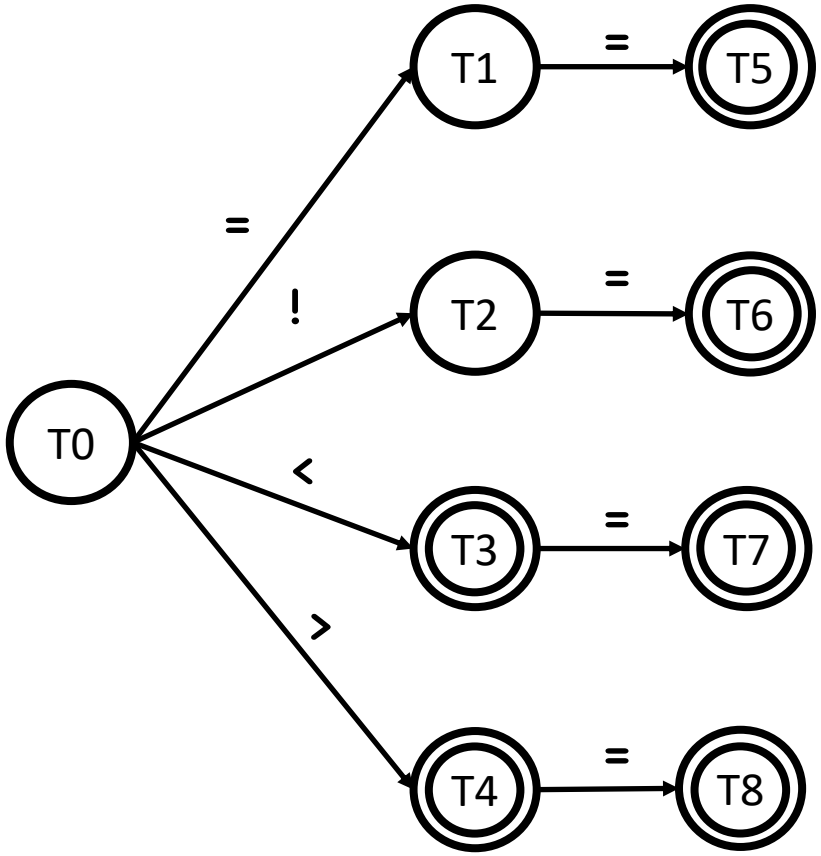
ε-closure(δ(T8, !)) = ∅

ε-closure(δ(T8, <)) = ∅

ε-closure(δ(T8, >)) = ∅

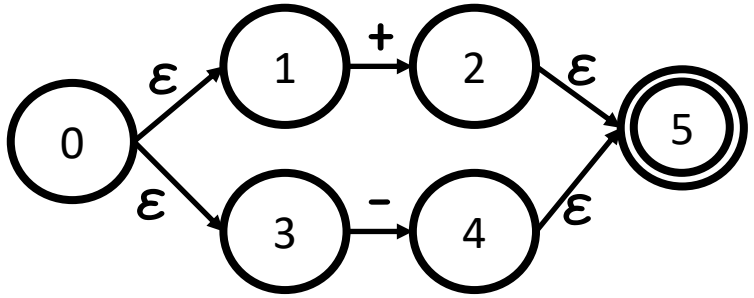
comp

- DFA



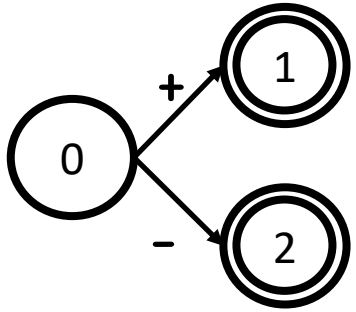
addsub

- alphabet: [+, -]
- regular expression: +|-



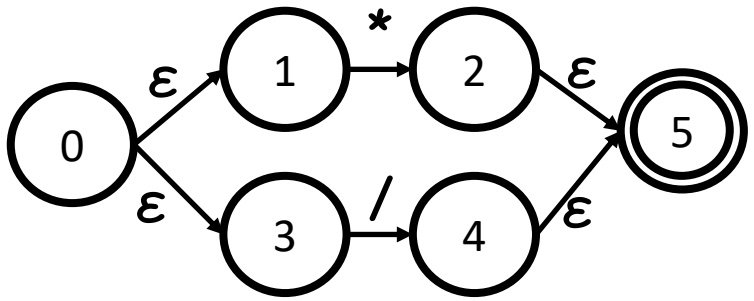
addsub

- DFA



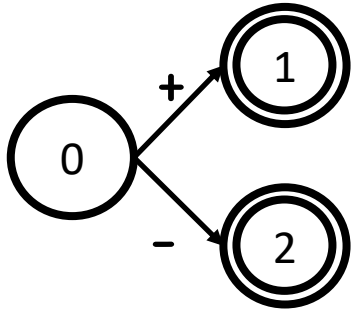
multidiv

- alphabet: [*, /]
- regular expression: *|/



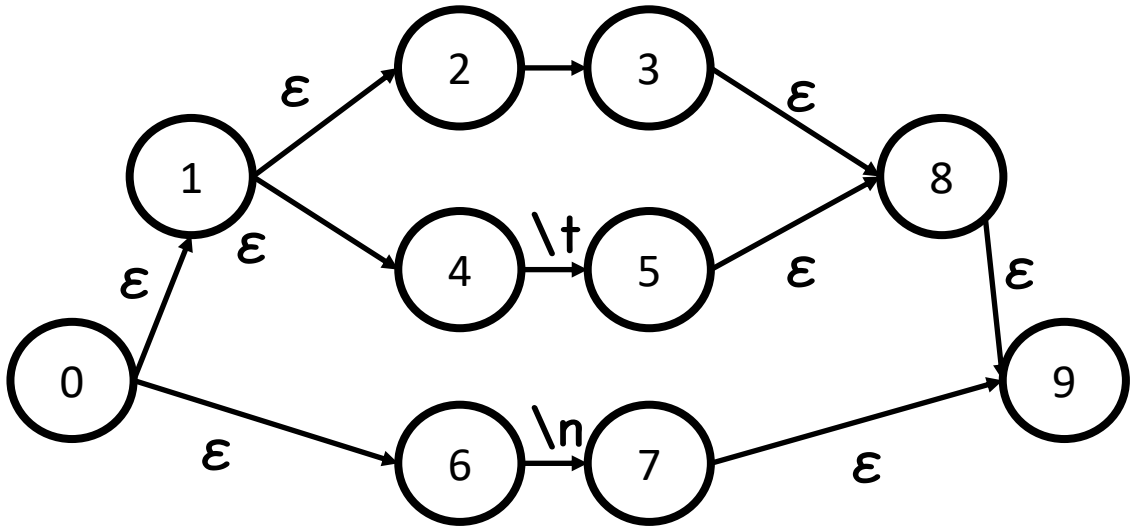
multidiv

- DFA



whitespace

- alphabet:[, \t, \n]
- regular expression: \t| |\n



whitespace

ε-closure(0) = {0, 1, 2, 4, 6} = T0

ε-closure(δ(T0,)) = {3, 8, 9} = T1
ε-closure(δ(T0, \n)) = {5, 8, 9} = T2
ε-closure(δ(T0, \t)) = {7, 9} = T3

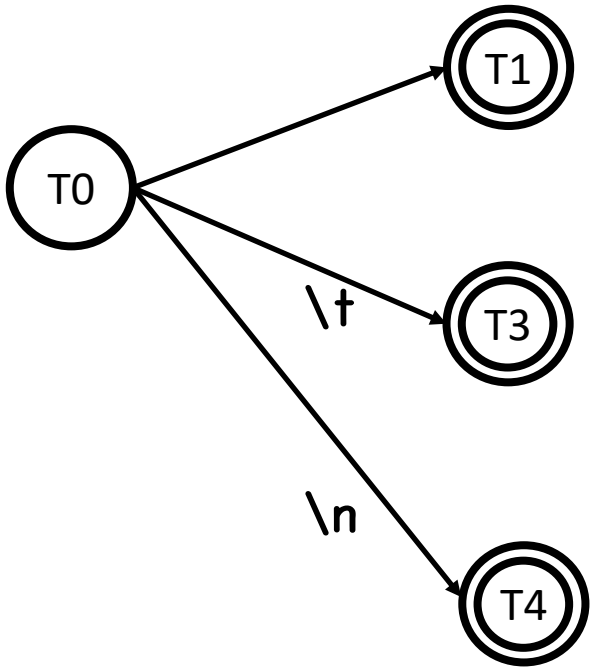
ε-closure(δ(T1,)) = ∅
ε-closure(δ(T1, \n)) = ∅
ε-closure(δ(T1, \t)) = ∅

ε-closure(δ(T2,)) = ∅
ε-closure(δ(T2, \n)) = ∅
ε-closure(δ(T2, \t)) = ∅

ε-closure(δ(T3,)) = ∅
ε-closure(δ(T3, \n)) = ∅
ε-closure(δ(T3, \t)) = ∅

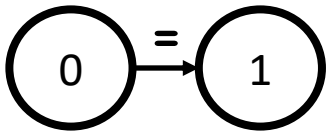
whitespace

- DFA

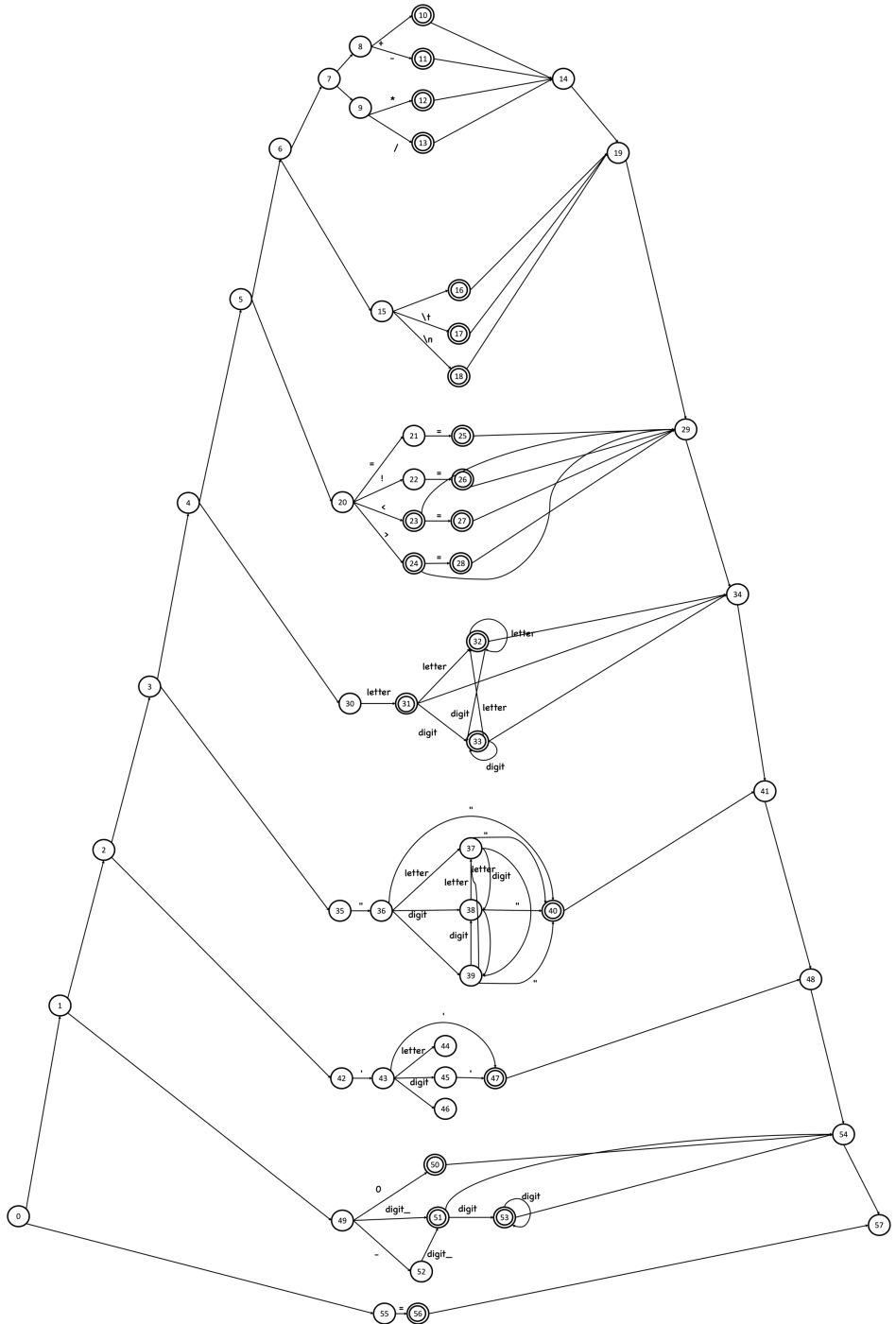


assign

- DFA



merged


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
+
-
*
/
\t
\n
=
!
<
>
letter
digit
"
'
0
digit_

merged

ε-closure(0) = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 15, 20, 30, 35, 42, 49, 55} = T0

ε-closure(δ(T0, +)) = {10, 14, 19, 29, 34, 41, 48, 54, 57} = T1 (addsub)

ε-closure(δ(T0, -)) = {11, 14, 19, 29, 34, 41, 48, 52, 54, 57 } = T2 (addsub)

ε-closure(δ(T2, digit_)) = {51, 54, 57} = T3 (num)

ε-closure(δ(T3, digit)) = {53, 54, 57} = T4 (num)

ε-closure(δ(T4, digit)) = {53, 54, 57} = T4 (num)

ε-closure(δ(T0, *)) = {12, 14, 19, 29, 34, 41, 48, 54, 57} = T5 (arithmetic)

ε-closure(δ(T0, /)) = {13, 14, 19, 29, 34, 41, 48, 54, 57} = T6 (arithmetic)

ε-closure(δ(T0,)) = {16, 19, 29, 34, 41, 48, 52, 54, 57 } = T7 (whitespace)

ε-closure(δ(T0, \t)) = {17, 19, 29, 34, 41, 48, 52, 54, 57 } = T8 (whitespace)

ε-closure(δ(T0, \n)) = {18, 19, 29, 34, 41, 48, 52, 54, 57 } = T9 (whitespace)

merged

ε-closure(δ(T0, =)) = {21, 56, 57} = T10 (assign)

ε-closure(δ(T10, =)) = {25, 29, 34, 41, 48, 54, 57} = T11 (comp)

ε-closure(δ(T0, !)) = {22} = T12

ε-closure(δ(T12, =)) = {26, 29, 34, 41, 48, 54, 57} = T13(comp)

ε-closure(δ(T0, <)) = {23, 29, 34, 41, 48, 54, 57} = T14 (comp)

ε-closure(δ(T14, =)) = {27, 29, 34, 41, 48, 54, 57} = T15 (comp)

ε-closure(δ(T0, >)) = {24, 29, 34, 41, 48, 54, 57} = T16 (comp)

ε-closure(δ(T16, =)) = {28, 29, 34, 41, 48, 54, 57} = T17 (comp)

merged

ε-closure(δ(T0, letter)) = {31, 34, 41, 48, 54, 57} = T18 (vtype)

ε-closure(δ(T18, letter)) = {32, 34, 41, 48, 54, 57} = T19 (vtype)

ε-closure(δ(T18, digit)) = {33, 34, 41, 48, 54, 57} = T20 (vtype)

ε-closure(δ(T19, letter)) = {32, 34, 41, 48, 54, 57} = T19 (vtype)

ε-closure(δ(T19, digit)) = {33, 34, 41, 48, 54, 57} = T20 (vtype)

ε-closure(δ(T20, letter)) = {32, 34, 41, 48, 54, 57} = T19 (vtype)

ε-closure(δ(T20, digit)) = {33, 34, 41, 48, 54, 57}} = T20 (vtype)

merged

ε-closure(δ(T0, ')) = {36} = T21

ε-closure(δ(T21, letter)) = {37} = T22
ε-closure(δ(T21, digit)) = {38} = T23
ε-closure(δ(T21,)) = {39} = T24

ε-closure(δ(T22, letter)) = {37} = T22
ε-closure(δ(T23, letter)) = {37} = T22
ε-closure(δ(T24, letter)) = {37} = T22

ε-closure(δ(T19, digit)) = {38} = T23
ε-closure(δ(T19, digit)) = {38} = T23
ε-closure(δ(T19, digit)) = {38} = T23

ε-closure(δ(T19,)) = {39} = T24
ε-closure(δ(T19,)) = {39} = T24
ε-closure(δ(T19,)) = {39} = T24

merged

ε-closure(δ(T21, ")) = {40, 41, 48, 54, 57} = T25 (string)

ε-closure(δ(T22, ")) = {40, 41, 48, 54, 57} = T25 (string)

ε-closure(δ(T23, ")) = {40, 41, 48, 54, 57} = T25 (string)

ε-closure(δ(T24, ")) = {40, 41, 48, 54, 57} = T25 (string)

merged

ε-closure(δ(T0, ')) = {43} = T26

ε-closure(δ(T26, letter)) = {44} = T27

ε-closure(δ(T26, digit)) = {45} = T28

ε-closure(δ(T26,)) = {46} = T29

ε-closure(δ(T27, ')) = {47, 48, 54, 57} = T30 (char)

ε-closure(δ(T28, ')) = {47, 48, 54, 57} = T30 (char)

ε-closure(δ(T29, ')) = {47, 48, 54, 57} = T30 (char)

merged

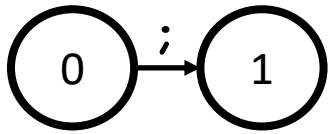
ε-closure(δ(T0, 0)) = {50, 54, 57} = T31 (num)

ε-closure(δ(T0, digit_)) = {51, 54, 57} = T32 (num)

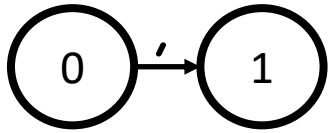
ε-closure(δ(T32, digit)) = {53, 54, 57} = T33 (num)

ε-closure(δ(T33, digit)) = {53, 54, 57} = T33 (num)

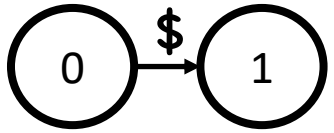
semi



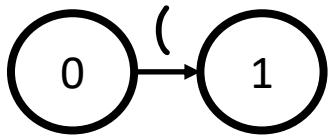
comma



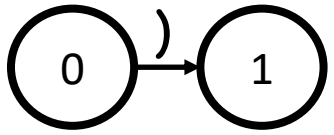
endmark



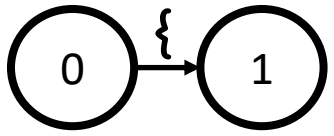
laparen



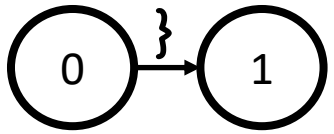
rparen



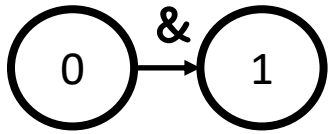
lbrace



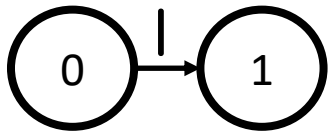
rbrace



and



or



2. Context-free Grammar (Syntax Analyzer)

Production Rules

1. S -> CODE
2. CODE -> VDECL CODE
3. CODE -> FDECL CODE
4. CODE -> ''
5. VDECL -> vtype id semi
6. VDECL -> vtype ASSIGN semi
7. ASSIGN -> id assign RHS
8. RHS -> EXPR
9. RHS -> char
10. RHS -> string
11. RHS -> boolstr
12. EXPR -> CALL
13. EXPR -> TERM addsub EXPR
14. EXPR -> TERM
15. TERM -> FACT multdiv TERM
16. TERM -> FACT
17. FACT -> num
18. FACT -> lparen EXPR rparen
19. FACT -> id
20. FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
21. ARG -> vtype id MOREARGS
22. ARG -> ''

Production Rules

23. MOREARGS -> comma vtype id MOREARGS
24. MOREARGS -> ''
25. BLOCK -> STMT BLOCK
26. BLOCK -> ''
27. STMT -> VDECL
28. STMT -> ASSIGN semi
29. STMT -> CALL semi
30. STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE
31. STMT -> while lparen COND rparen lbrace BLOCK rbrace
32. CALL -> id lparen ARGVAL rparen
33. ARGVAL -> EXPR MOREARGVAL
34. ARGVAL -> ''
35. MOREARGVAL -> comma EXPR MOREARGVAL
36. MOREARGVAL -> ''
37. COND -> COND_EXPR LOGIC_EXPR
38. COND_EXPR -> boolstr
39. COND_EXPR -> FACT comp FACT
40. LOGIC_EXPR -> ''
41. LOGIC_EXPR -> LOGIC_OP COND
42. LOGIC_OP -> and
43. LOGIC_OP -> or
44. ELSE -> else lbrace BLOCK rbrace
45. ELSE -> ”
46. RETURN -> return RHS semi

Terminals

1. **vtype** for the types of variables and functions
2. **num** for signed integers
3. **char** for a single character
4. **boolstr** for Boolean strings
5. **string** for literal strings
6. **id** for the identifiers of variables and functions
7. **if**, **else**, **while**, and **return** for if, else, while, and return statements respectively
8. **addsub** for +, and -
9. **multidiv** for *, and /
10. **assign** for assignment operators
11. **comp** for comparison operators
12. **semi** and **comma** for semicolons and commas respectively
13. **lparen**, **rparen**, **lbrace**, and **rbrace** for (,), {, and } respectively
14. **and**, and **or** for & and |

Non-termianls

EXPR, VDECL, LOGIC_EXPR, CODE, CALL, ELSE, ARGVAL, MOREARGVAL, BLOCK, ASSIGN,STMT, FACT, COND, MOREARGS, RHS, S, RETURN, COND_EXPR, TERM, FDECL, ARG, LOGIC_OP

Descriptions

1. A file of source code is constructed with zero or more declarations of functions or variables
2. Declaration of a variable is allowed regardless of initialization
3. Five types of the right-hand side of a variable
    a) Expression
    b) String
    c) Character
    d) Boolean
    e) Function return value
4. A function has zero or more arguments
5. A function block includes zero or more statements
6. Five types of statements
    a) Variable declaration
    b) Assignment operation
    c) Function call
    d) If-else statement
    e) While statement

Descriptions

7. If and while statements include one or more conditions that comprise Boolean values generated by comparison operations or logical operations
8. If statement stands alone or with an else statement
9. A return statement returns five types
    a) Expression
    b) String
    c) Character
    d) Boolean
    e) Function return value