Automata Design Document

Muhammad Shaffan Ahmad 23i-0673

Zaid Bin Umer 23i-0671

1. Overview

The Custom Language Scanner uses a Deterministic Finite Automaton (DFA) generated by JFlex to recognize tokens. This document describes the regular expressions, individual NFAs, and the final optimized DFA.

2. Regular Expressions (Lexical Specification)

The following regular expressions define the token classes for the language:

Token Class	Regular Expression	Description
Single Line Comment	##[^\n]*	Starts with ##, followed by non-newline chars
Boolean Literal	(true|false)	
Identifier	[A-Z][a-z0-9]{0,30}	Uppercase start, mixed case tail (Max 31 chars)
Integer Literal	[+-]?[0-9]+	Optional sign, followed by digits
Floating Point	[+-]?[0-9]+\.[0-9]{1,6}([eE][+-]?[0-9]+)?	Decimal required, optional exponent. Max 6 decimal digits.
Operator	[+\-*/%<>=!]	Single character operators
Punctuator	[(){}[\][,;:]]	Braces, parentheses, and delimiters

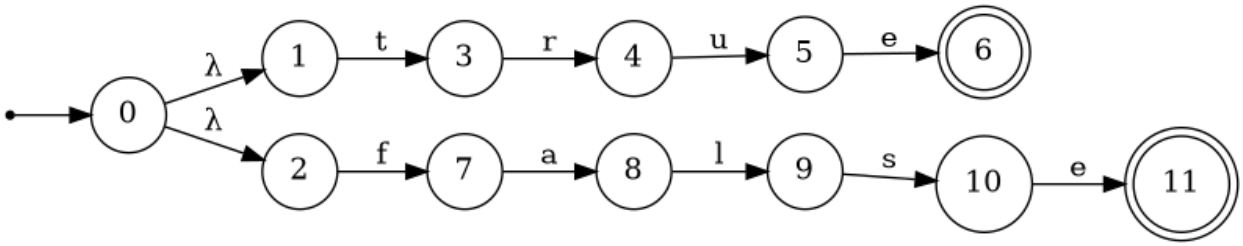
3. Individual NFA Transition Tables

These tables represent the Non-Deterministic Finite Automata for each individual token class before combination.

3.1 Boolean Literal NFA

Regex: true | false

Accept States: [6, 11]

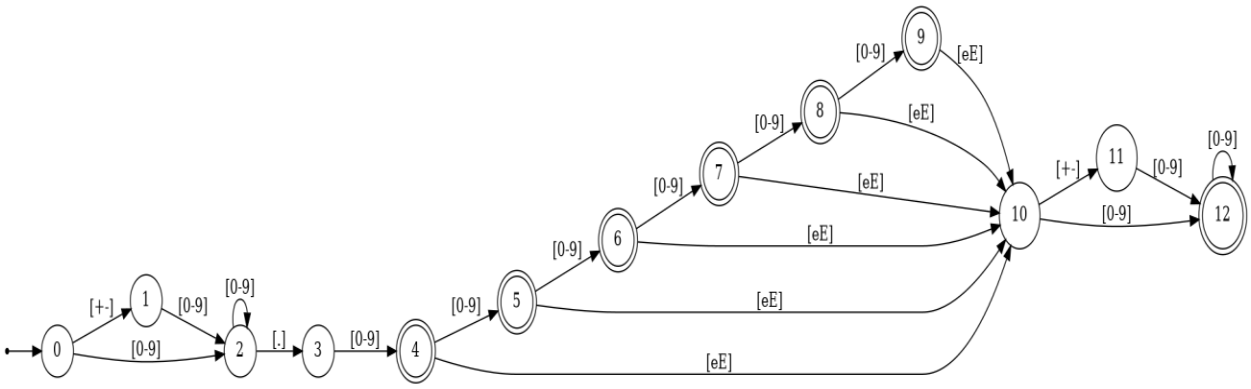


State	Input	Next State
q0	λ	q1, q2
q1	t	q3
q2	f	q7
q3	r	q4
q4	u	q5
q5	e	q6 (Accept)
q7	a	q8
q8	l	q9
q9	s	q10
q10	e	q11 (Accept)

3.2 Floating Point Literal NFA

Regex: [+-]?[0-9]+\.[0-9]+([eE][+-]?[0-9]+)?

Accept States: [4, 5, 6, 7, 8, 9, 12] (Enforcing precision limits)



State Input Next State

---

q0 [+-] q1

q0 [0-9] q2

q1 [0-9] q2

q2 [0-9] q2

q2 . q3

q3 [0-9] q4

q4 [0-9] q5

q4 [eE] q10

q5 [0-9] q6

q5 [eE] q10

... (states q6-q9 enforce max 6 decimal digits) ...

q10 [+-] q11

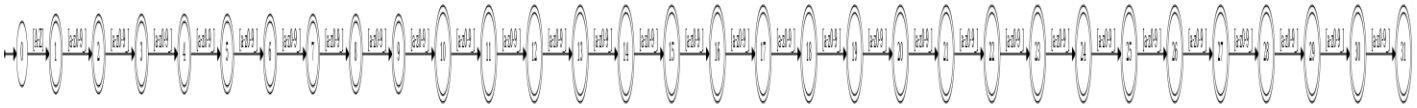
q10 [0-9] q12

q12 [0-9] q12 (Accept)

3.3 Identifier NFA

Regex: [A-Z][a-z0-9_]* (Max 31 chars)

Accept States: [1..31]



State Input Next State

---

q0 [A-Z] q1

q1 [a-z0-9_] q2

q2 [a-z0-9_] q3

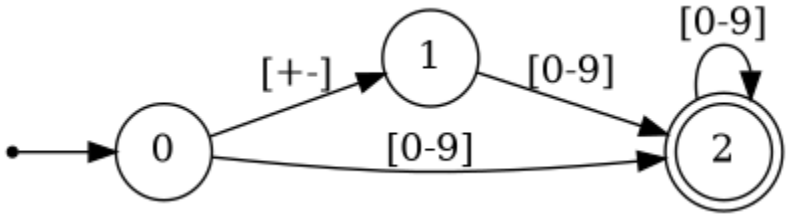
...

q30 [a-z0-9_] q31 (Accept, max length reached)

3.4 Integer Literal NFA

Regex: [+-]?[0-9]+

Accept States: [2]



State Input Next State

---

q0 [+-] q1

q0 [0-9] q2

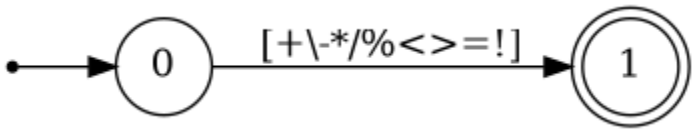
q1 [0-9] q2

q2 [0-9] q2 (Accept)

3.5 Single Char Operator NFA

Regex: [+\-*/%<>=!]

Accept States: [1]



State Input Next State

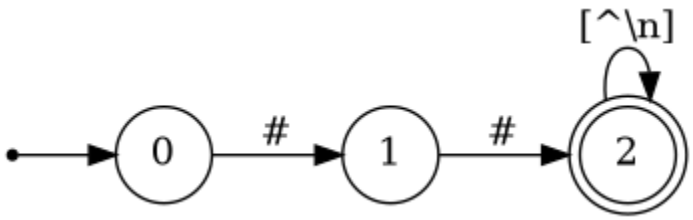
---

q0 [+\-*/%<>=!] q1 (Accept)

3.6 Single Line Comment NFA

Regex: ##[^\n]*

Accept States: [2]



State Input Next State

---

q0 # q1

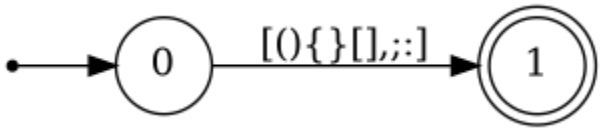
q1 # q2

q2 [^\n] q2 (Accept loop)

3.7 Punctuator NFA

Regex: [(){}\[\],;:]

Accept States: [1]



State	Input	Next State
q0	[(){}\[\],;:]	q1 (Accept)

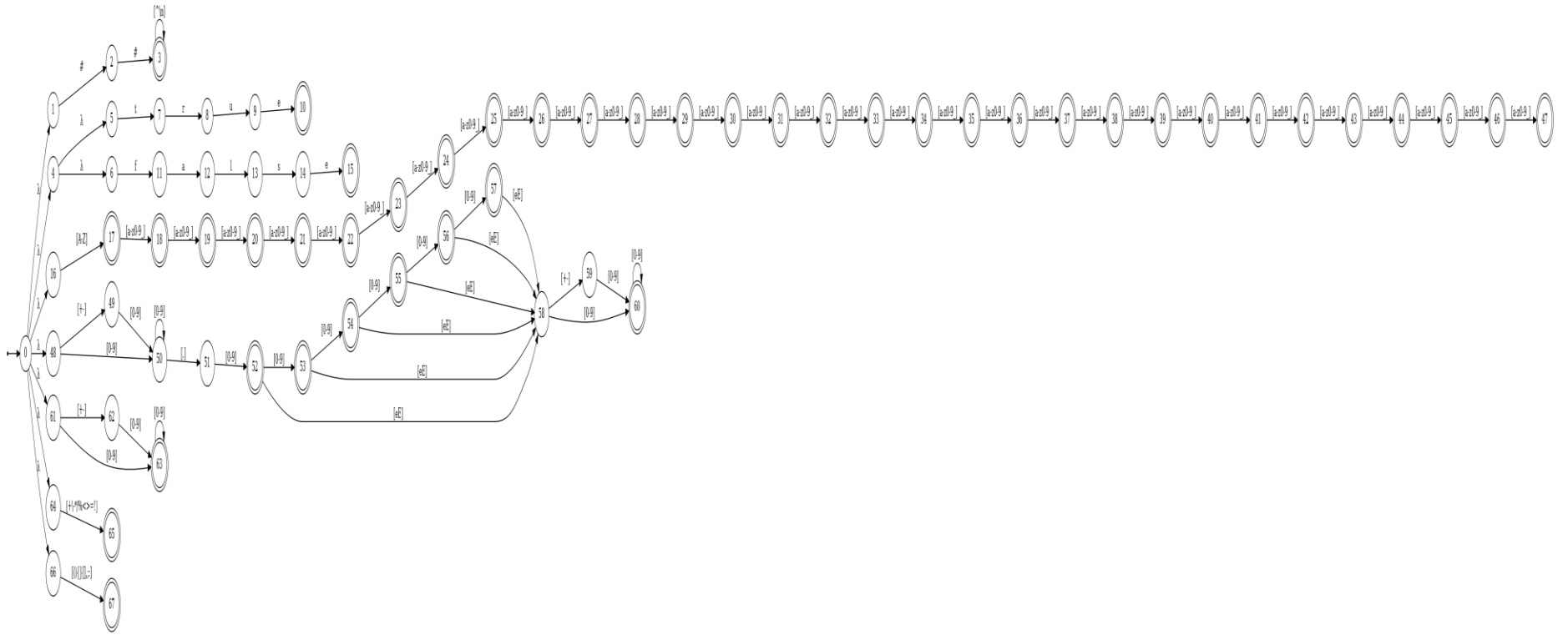
4. Combined NFA

The individual NFAs are combined into a single machine using epsilon (λ) transitions from a new start state.

Start State: 0

Accept States: [3, 10, 15, 17…56] (Union of all NFA accept states)

Priority Order: Comment > Boolean > Identifier > Float > Integer > Operator > Punctuator


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
58
59
60
61
62
63
64
65
66
67
[^\n]
#
#
l
t
r
u
e
l
f
a
l
s
e
[A-Z]
[a-z0-9_]
[+-]
[0-9]
[0-9]
[.]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[eE]
[eE]
[eE]
[eE]
[eE]
[+-]
[0-9]
[0-9]
[+-]
[0-9]
[0-9]
[0-9]
[(){}[],;]

State Input Next State

---

q0 λ q1 (Comment Start)

q0 λ q4 (Boolean Start)

q0 λ q16 (Identifier Start)

q0 λ q48 (Float Start)

q0 λ q61 (Integer Start)

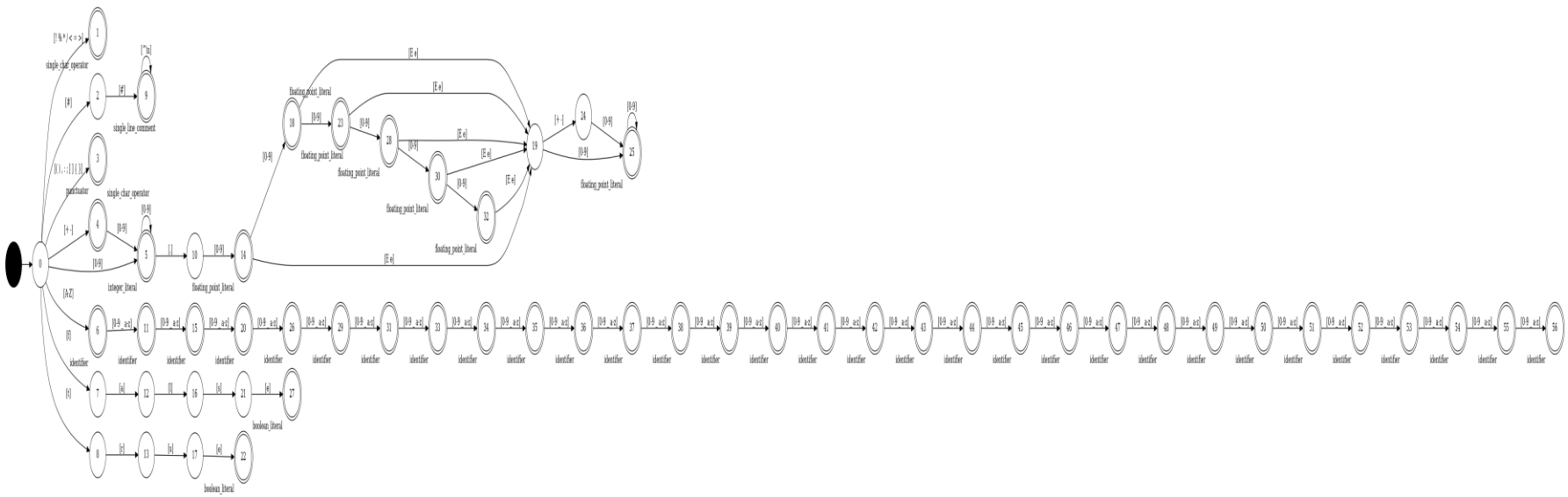
q0 λ q64 (Operator Start)

q0 λ q66 (Punctuator Start)

... (Internal transitions from individual NFAs) ...

5. Minimized DFA

After subset construction (NFA → DFA)


0
1
[!%*/<=>]
single char operator
2
[#]
[^\n]
9
single_line_comment
3
[(),.:;][]{}
punctuator
single char operator
4
[+-]
[0-9]
5
[0-9]
integer_literal
[.]
10
[0-9]
floating_point_literal
16
[0-9]
18
floating_point_literal
[E e]
[0-9]
23
floating_point_literal
[E e]
[0-9]
28
floating_point_literal
[E e]
[0-9]
30
floating_point_literal
[E e]
[0-9]
32
floating_point_literal
[E e]
19
[+-]
24
[0-9]
25
[0-9]
floating_point_literal
[0-9]
[A-Z]
6
identifier
[0-9 _ a-z]
11
identifier
[0-9 _ a-z]
15
identifier
[0-9 _ a-z]
20
identifier
[0-9 _ a-z]
26
identifier
[0-9 _ a-z]
29
identifier
[0-9 _ a-z]
31
identifier
[0-9 _ a-z]
33
identifier
[0-9 _ a-z]
34
identifier
[0-9 _ a-z]
35
identifier
[0-9 _ a-z]
36
identifier
[0-9 _ a-z]
37
identifier
[0-9 _ a-z]
38
identifier
[0-9 _ a-z]
39
identifier
[0-9 _ a-z]
40
identifier
[0-9 _ a-z]
41
identifier
[0-9 _ a-z]
42
identifier
[0-9 _ a-z]
43
identifier
[0-9 _ a-z]
44
identifier
[0-9 _ a-z]
45
identifier
[0-9 _ a-z]
46
identifier
[0-9 _ a-z]
47
identifier
[0-9 _ a-z]
48
identifier
[0-9 _ a-z]
49
identifier
[0-9 _ a-z]
50
identifier
[0-9 _ a-z]
51
identifier
[0-9 _ a-z]
52
identifier
[0-9 _ a-z]
53
identifier
[0-9 _ a-z]
54
identifier
[0-9 _ a-z]
55
identifier
[0-9 _ a-z]
56
identifier
[t]
7
[a]
12
[l]
16
[s]
21
[e]
27
boolean_literal
[f]
8
[r]
13
[u]
17
[e]
22
boolean_literal

and minimization, we achieve the optimal state machine.

Start State: 0

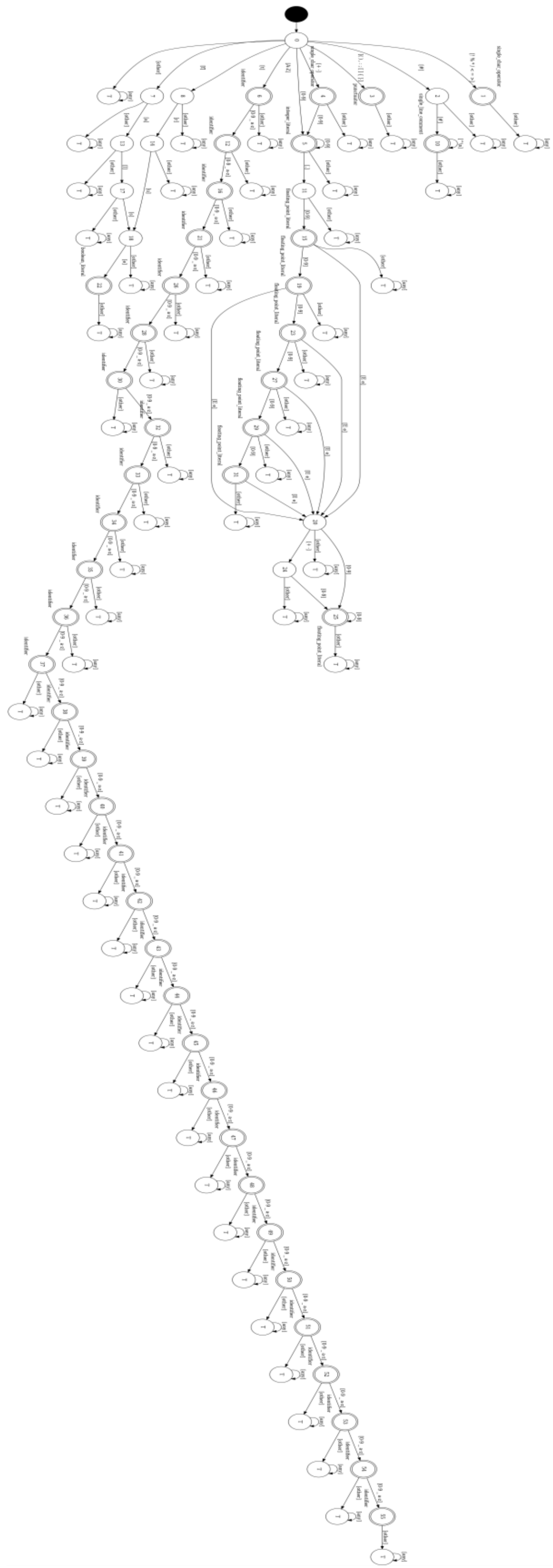
Total States: 56

Accept States: [1, 3, 4, 5, 6, 10, 12, 15, 16, 19, 21…55]

5.1 Acceptance State Labels

State	Token Recognized
q1	Single Char Operator
q3	Punctuator
q5	Integer Literal
q6, q12, q16…	Identifier
q10	Single Line Comment
q15, q19, q23…	Floating Point Literal
q22	Boolean Literal (true or false)

5.2 Transition Table (Min DFA)



State Input Next State

---

q0 [!%*/<>=] q1

q0 [#] q2

q0 [(){}[],;:] q3

q0 [+-] q4

q0 [0-9] q5

q0 [A-Z] q6

q0 [f] q7

q0 [t] q8

q0 [other] q9 (Dead State)

q2 [#] q10 (Comment Start)

q4 [0-9] q5 (Signed Int)

q5 [.] q11 (Float Start)

q6 [a-z0-9_] q12 (Identifier)

q7 [a] q13 (Start of 'false')

q8 [r] q14 (Start of 'true')

q13 [l] q17

q14 [u] q18

q17 [s] q18 (Merge 'false'/'true' suffix)

q18 [e] q22 (Boolean Accept)

q11 [0-9] q15 (Float digit 1)

q15 [0-9] q19 (Float digit 2)

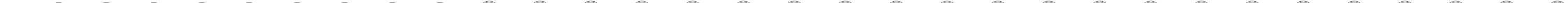
...

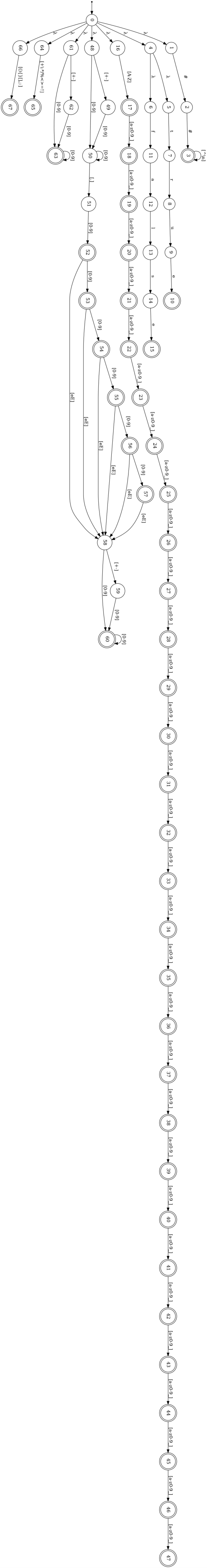
q31 [0-9] q9 (Precision limit exceeded -> Error)

6. Optimization Summary

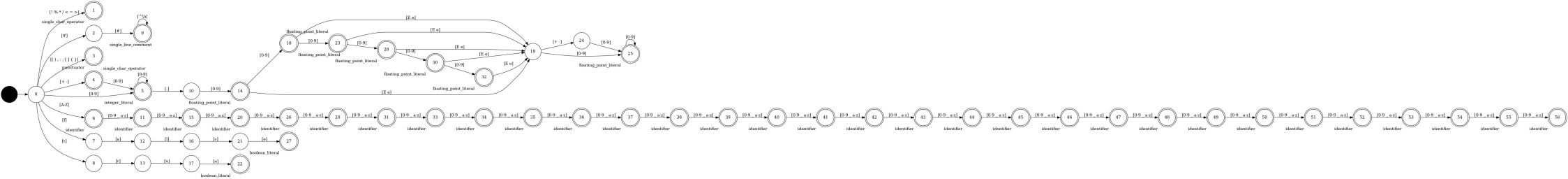
1. **Boolean Literal Merging:** The suffixes of true and false were effectively merged by the minimization algorithm. State q18 handles the e transition for both.
2. **Precision Enforcement:** The DFA structure (states q11 → q31) explicitly enforces the 6-digit limit for floating-point numbers. Any digit after the 6th transitions to the dead state (q9) or implies the end of the token.
3. **Ambiguity Resolution:**
    - + leads to q4 (Operator Accept).
    - + followed by [0-9] leads to q5 (Integer Accept).
    - This structure enforces the "Longest Match" rule natively.

For Clarity Purposes, pictures of Identifier's NFA, Combined NFA, Combined DFA, and Minimized DFA are attached at the end:


0
[A-Z]
1
[a-z0-9_]
2
[a-z0-9_]
3
[a-z0-9_]
4
[a-z0-9_]
5
[a-z0-9_]
6
[a-z0-9_]
7
[a-z0-9_]
8
[a-z0-9_]
9
[a-z0-9_]
10
[a-z0-9_]
11
[a-z0-9_]
12
[a-z0-9_]
13
[a-z0-9_]
14
[a-z0-9_]
15
[a-z0-9_]
16
[a-z0-9_]
17
[a-z0-9_]
18
[a-z0-9_]
19
[a-z0-9_]
20
[a-z0-9_]
21
[a-z0-9_]
22
[a-z0-9_]
23
[a-z0-9_]
24
[a-z0-9_]
25
[a-z0-9_]
26
[a-z0-9_]
27
[a-z0-9_]
28
[a-z0-9_]
29
[a-z0-9_]
30
[a-z0-9_]
31



λ
#
[^\n]
t
r
u
e
f
a
l
s
[A-Z]
[a-z0-9_]
[+-]
[0-9]
[.]
[eE]


0
1
{ ! % */ < = >}
single_char_operator
2
[#]
[#]
9
[^\n]
single_line_comment
3
[( ) , ;] { }]
single_char_operator
4
[+ -]
[0-9]
integer_literal
5
[0-9]
[.]
10
[0-9]
floating_point_literal
14
[0-9]
18
floating_point_literal
[0-9]
23
floating_point_literal
[0-9]
28
floating_point_literal
[0-9]
30
floating_point_literal
[0-9]
32
floating_point_literal
[E e]
[E e]
[E e]
[E e]
19
[+ -]
24
[0-9]
floating_point_literal
[0-9]
25
[0-9]
[E e]
[A-Z]
[f]
[t]
6
[0-9 _ a-z]
identifier
7
[a]
12
[l]
16
[s]
21
[e]
boolean_literal
8
[r]
13
[u]
17
[e]
22
boolean_literal
11
15
20
26
29
31
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56