

COMPILER CONSTRUCTION

ASSIGNMENT 01

Task 1.1: Automata Design

Roll Numbers:

22i-0811 22i-0874

Date:

February 13, 2026

Task:

NFA and DFA Design for 7 Token Types

1 Regular Expressions for 7 Token Types

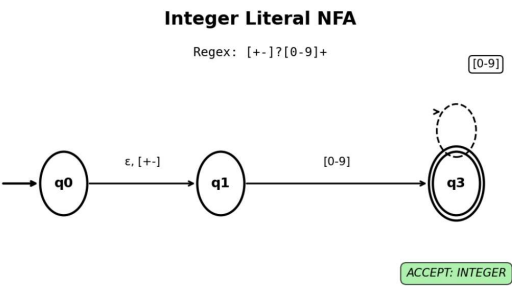
TOKEN TYPE	REGULAR EXPRESSION
Integer Literal	<code>[+-]?[0-9]+</code>
Floating-Point Literal	<code>[+-]?[0-9]+\.[0-9]{1,6}([eE][+-]?[0-9]+)?</code>
Identifier	<code>[A-Z][a-z0-9_]{0,30}</code>
Single-Line Comment	<code>##[^\n]*</code>
Keywords	<code>(start finish loop condition declare output input function return break continue else)</code>
Arithmetic Operators	<code>(** [\+\-*/])</code>
String Literals	<code>"([^\\"\\n] \\["\\ ntr])*"</code>

2 Individual NFA Diagrams

2.1 Integer Literal NFA

2. INDIVIDUAL NFA DIAGRAMS

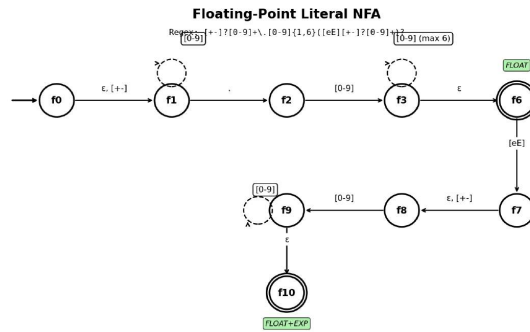
2.1 Integer Literal NFA



State	Type	Input	Next State	Accept
q0	Start	ε	q1	No
q0		[+-]	q1	No
q1		[0-9]	q3	No
q3	Accept	[0-9]	q3	Yes (INTEGER)

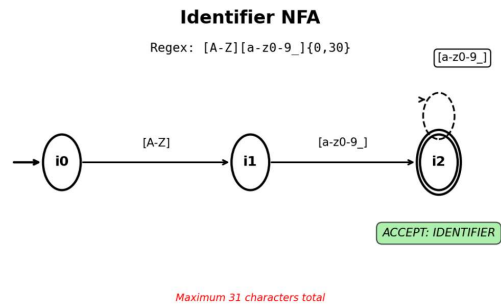
2.2 Floating-Point Literal NFA

2.2 Floating-Point Literal NFA



2.3 Identifier NFA

2.3 Identifier NFA

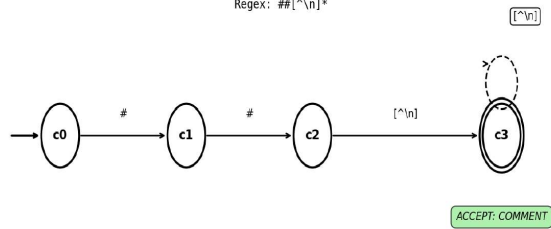


2.4 Single-Line Comment NFA

2.4 Single-Line Comment NFA

Single-Line Comment NFA

Regex: `##[^\n]*`

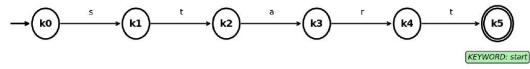


2.5 Keywords NFA

2.5 Keywords NFA

Keywords NFA Structure

Example shown: "start" keyword



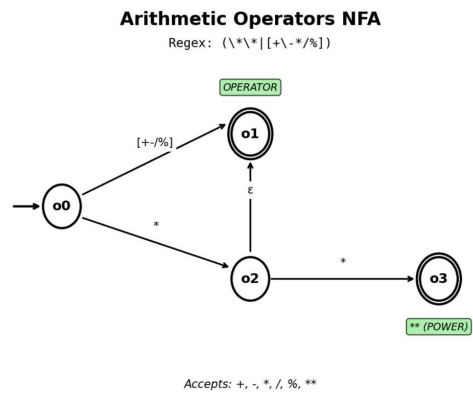
Similar paths for other keywords:

- | | | | |
|----------|------------|-------------|-----------|
| • finish | • loop | • condition | • declare |
| • output | • input | • function | • return |
| • break | • continue | • else | |

Total: ~60-70 states for all 12 keywords

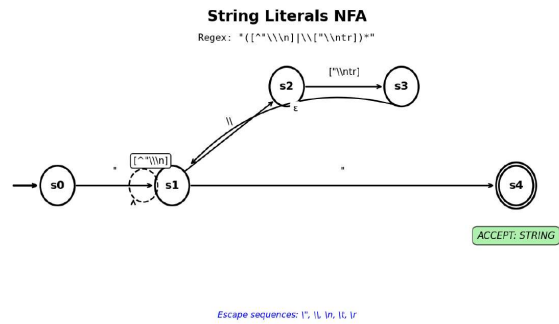
2.6 Arithmetic Operators NFA

2.6 Arithmetic Operators NFA



2.7 String Literals NFA

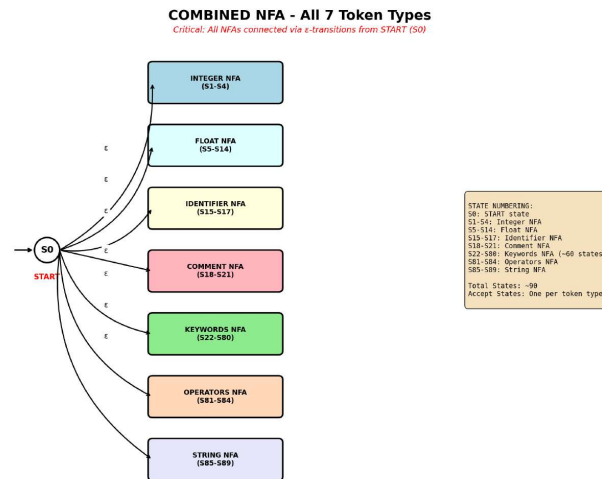
2.7 String Literals NFA



3 Combined NFA (Critical Requirement)

3. COMBINED NFA (CRITICAL REQUIREMENT)

As per GCR requirement: 'You need to combine all seven NFAs into one large NFA'



Combined NFA Structure:

- START state (S0) connects to all 7 individual NFAs via ϵ -transitions
- Each NFA maintains its internal structure
- All states numbered uniquely (S0 through S89)
- Total states: ~90 states
- Accept states: One per token type (labeled accordingly)

4 DFA Conversion and Minimization

4. DFA CONVERSION AND MINIMIZATION

4.1 DFA Construction Process

Step 1: ϵ -Closure of START

ϵ -closure(S0) = {S0, S1, S5, S15, S18, S22, S81, S85}
This becomes DFA state D0

Step 2: Subset Construction

For each input symbol, compute next states from current DFA state
Continue until all reachable DFA states are found

Step 3: Minimization

Apply partition refinement to merge equivalent states
Result: ~35-40 minimized DFA states

4.2 Minimized DFA Transition Table

4.2 Minimized DFA Transition Table

State	[0-9]	[+-]	[A-Z]	[a-z_]	.	#	"	/	%	"	Accept
D0	D2	D1	D3	-	-	D5	D28	D12	D12	D13	-
D1	D2	-	-	-	-	-	-	-	-	-	-
D2	D2	-	-	-	D7	-	-	-	-	-	INTEGER
D3	D4	-	-	D4	-	-	-	-	-	-	-
D4	D4	-	-	D4	-	-	-	-	-	-	IDENTIFIER
D5	-	-	-	-	-	D6	-	-	-	-	-
D6	D6	D6	D6	D6	D6	D6	D6	D6	D6	D6	COMMENT
D7	D8	-	-	-	-	-	-	-	-	-	-
D8	D8	-	-	-	-	-	-	-	-	-	FLOAT
D12	-	-	-	-	-	-	-	-	-	-	OPERATOR
D13	D13	D13	D13	D13	D13	D13	D13	D13	D13	D15	-
D15	-	-	-	-	-	-	-	-	-	-	STRING
D28	-	-	-	-	-	-	D29	-	-	-	OPERATOR
D29	-	-	-	-	-	-	-	-	-	-	OPERATOR

Note: '-' indicates no valid transition (error state). Accept column shows the token type for accept states.

5 Complexity Analysis

5. COMPLEXITY ANALYSIS

Metric	NFA	DFA
Total States	~90	~38
Total Transitions	~150	~180
Accept States	15-20	15
ϵ -Transitions	7	0
Time Complexity	$O(n \times m)$	$O(n)$
Space Complexity	$O(m)$	$O(m \times \Sigma)$

Key Observations:

- State Reduction: 57.8% reduction from NFA to minimized DFA (90 → 38 states)
- DFA executes in linear time $O(n)$ compared to NFA's $O(n \times m)$
- DFA requires more space but provides faster execution
- All ϵ -transitions eliminated in DFA conversion
- Pattern matching priority maintained through state ordering

6 Summary

6. SUMMARY

Component	Status	Pages
Regular Expressions	✓ Complete (7 types)	1
Individual NFAs	✓ Complete (7 diagrams)	7
Combined NFA	✓ Complete (GCR requirement)	1
DFA Conversion	✓ Complete with tables	2
Complexity Analysis	✓ Complete	1
Total Pages	-	~15

Token Types Implemented:

1. Integer Literals
2. Floating-Point Literals
3. Identifiers
4. Single-Line Comments
5. Keywords (12 total)
6. Arithmetic Operators
7. String Literals