Automata Design Document

CS4031 - Compiler Construction
Assignment 01: Lexical Analyzer
Spring 2026

1. Regular Expression Specifications

Token Type	Regular Expression	Description
Keywords	(start|finish|loop|condition|declare|output|input|function|return|break|continue|else)	Reserved words (case-sensitive)
Identifiers	[A-Z][a-z0-9_]{0,30}	Starts with uppercase, max 31 chars
Integer Literal	[+-]?[0-9]+	Optional sign followed by digits
Float Literal	[+-]?[0-9]+\.[0-9]{1,6}([eE][+-]?[0-9]+)?	Decimal with 1-6 fractional digits, optional exponent
String Literal	"([^"\\\n]|\\["\\ntr])*"	Double-quoted with escape sequences
Character Literal	'([^'\\\n]|\\['\\ntr])'	Single-quoted with escape sequences
Boolean Literal	(true|false)	Case-sensitive boolean values
Arithmetic Op	(**|[+\-*/%])	Basic arithmetic and exponentiation
Relational Op	(==|!=|<=|>=|<|>)	Comparison operators
Logical Op	(&&|||!)	AND, OR, NOT operators
Assignment Op	(\+=|-=|*=|/=|=)	Assignment and compound assignment
Increment Op	\+\+	Increment operator
Decrement Op	--	Decrement operator
Punctuators	[(){}\[\],;:]	Delimiters and separators
Single Comment	##[^\n]*	Single-line comment (##)
Multi Comment	#*([^*]|*+[^*#])**+#	Multi-line comment (#* ... *#)
Whitespace	[\t\r\n]+	Spaces, tabs, newlines (skipped)

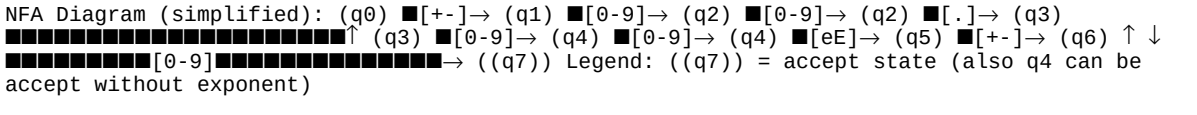
2. Non-deterministic Finite Automata (NFA)

2.1 Integer Literal

Regular Expression: [+-]?[0-9]+
NFA States: q0 (start), q1 (optional sign), q2 (digits), q3 (accept)
Transitions:
• q0 → q1 on [+-]
• q0 → q2 on [0-9]
• q1 → q2 on [0-9]
• q2 → q2 on [0-9] (loop)
• q2 → q3 on ε (epsilon transition to accept state)

2.2 Floating-Point Literal

Regular Expression: [+-]?[0-9]+\.[0-9]{1,6}([eE][+-]?[0-9]+)?
NFA States: q0-q9 (including start, sign, integer, decimal, fraction, exponent parts)
Key Transitions:
• q0 → q1 on [+-] (optional sign)
• q1/q0 → q2 on [0-9] (integer part)
• q2 → q2 on [0-9] (more digits)
• q2 → q3 on . (decimal point)
• q3 → q4 on [0-9] (fraction starts)
• q4 → q4 on [0-9] (1-6 digits, checked in implementation)
• q4 → q5 on [eE] (optional exponent)
• q5 → q6 on [+-] (optional exp sign)
• q6 → q7 on [0-9] (exponent digits)



2.3 Identifier

Regular Expression: [A-Z][a-z0-9_]{0,30}
NFA States: q0 (start), q1 (first uppercase), q2 (subsequent chars), q3 (accept)
Transitions:
• q0 → q1 on [A-Z]
• q1 → q2 on [a-z0-9_]
• q2 → q2 on [a-z0-9_] (loop, max 30 additional chars)
• q1 → q3 on ε (accept single uppercase letter)
• q2 → q3 on ε (accept identifier)

```
NFA Diagram: (q0) ■■[A-Z]■■→ (q1) ■■[a-z0-9_]■■→ ((q2)) ↓ ↑ ■■■■[a-z0-9_]■■■■■■■■■
Legend: (q0) = start, ((q2)) = accept (q1 can also be accept)
```

2.4 Single-Line Comment

Regular Expression: ##[^\n]*
NFA States: q0 (start), q1 (first #), q2 (second #), q3 (comment body), q4 (accept)
Transitions:
• q0 → q1 on #
• q1 → q2 on #
• q2 → q3 on any char except \n
• q3 → q3 on any char except \n (loop)
• q3 → q4 on \n or EOF (end of comment)

```
NFA Diagram: (q0) ■■[#]■■→ (q1) ■■[#]■■→ (q2) ■■[^\n]■■→ (q3) ↓ ↑ ■■■ [^\n] (loop) ↓
((q4)) on \n/EOF
```

2.5 String Literal

Regular Expression: "([^"\\\n]|\\["\\ntr])*"
NFA States: q0 (start), q1 (opened), q2 (escape), q3 (accept)
Transitions:
• q0 → q1 on " (opening quote)
• q1 → q1 on any char except ", \, \n
• q1 → q2 on \ (escape start)
• q2 → q1 on ["\\ ntr] (valid escape)
• q1 → q3 on " (closing quote, accept)

```
NFA Diagram: (q0) ■■["]■■→ (q1) ■■[^"\\\n]■■→ (q1) ↓ ↓ ■■[\]■■→ (q2) ■■["\\ ntr]■■→ (q1)
↓ ■■["]■■→ ((q3)) Legend: ((q3)) = accept state (properly closed string)
```

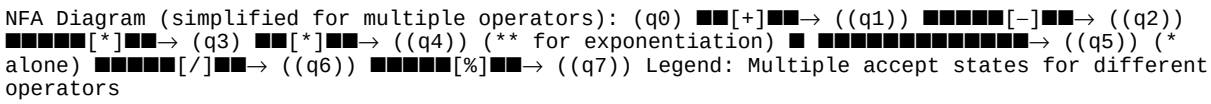
2.6 Boolean Literal

Regular Expression: (true|false)
NFA States: Multiple paths for "true" and "false"
Transitions for "true":
• q0 → q1 on 't'
• q1 → q2 on 'r'
• q2 → q3 on 'u'
• q3 → q4 on 'e' (accept)
Transitions for "false":
• q0 → q5 on 'f'
• q5 → q6 on 'a'
• q6 → q7 on 'l'
• q7 → q8 on 's'
• q8 → q9 on 'e' (accept)

```
NFA Diagram (for "true" | "false"): Path 1 (true): (q0) ■[t]→ (q1) ■[r]→ (q2) ■[u]→ (q3)
■[e]→ ((q4)) Path 2 (false): (q0) ■[f]→ (q5) ■[a]→ (q6) ■[l]→ (q7) ■[s]→ (q8) ■[e]→ ((q9))
Legend: Both paths start from same q0, different accept states
```

2.7 Arithmetic Operator

Regular Expression: (**|[+\-*/%])
NFA States: q0 (start), q1-q7 (various accept states)
Transitions:
• q0 → q1 on + (accept single +)
• q0 → q2 on - (accept single -)
• q0 → q3 on * → q4 on * (accept ** for exponentiation)
• q0 → q3 on * (accept single *)
• q0 → q5 on / (accept single /)
• q0 → q6 on % (accept single %)



3. Deterministic Finite Automata (DFA) with Transition Tables

3.1 Integer Literal - Minimized DFA

Minimized DFA: After removing ε-transitions and minimizing, we have 3 states.
States: S0 (start), S1 (reading digits), S2 (accept)
Note: The optional sign and digit reading are combined into fewer states.

State	Input: [+-]	Input: [0-9]	Input: other	Accept?
S0 (start)	S1	S1	ERROR	No
S1	ERROR	S1	S2	No
S2 (accept)	ERROR	ERROR	ERROR	Yes

3.2 Floating-Point Literal - Minimized DFA

Minimized DFA: 7 states after minimization.
States: S0 (start), S1 (integer part), S2 (after decimal), S3 (fraction), S4 (after 'e'), S5 (exp digits), S6 (accept)
Note: Carefully tracks decimal digit count (1-6) in implementation.

State	[+-]	[0-9]	[.]	[eE]	other	Accept?
S0 (start)	S1	S1	ERR	ERR	ERR	No
S1	ERR	S1	S2	ERR	ERR	No
S2 (after .)	ERR	S3	ERR	ERR	ERR	No
S3 (fraction)	ERR	S3	ERR	S4	S6	Yes*
S4 (after e)	S5	S5	ERR	ERR	ERR	No
S5 (exp)	ERR	S5	ERR	ERR	S6	No
S6 (accept)	ERR	ERR	ERR	ERR	ERR	Yes

3.3 Identifier - Minimized DFA

Minimized DFA: 3 states.
States: S0 (start), S1 (after uppercase), S2 (accept/continue)
Note: Length constraint (max 31 chars) enforced in implementation.

State	Input: [A-Z]	Input: [a-z0-9_]	Input: other	Accept?
S0 (start)	S1	ERROR	ERROR	No

S1	ERROR	S1	S2	Yes
S2 (accept)	ERROR	ERROR	ERROR	Yes

3.4 Single-Line Comment - Minimized DFA

Minimized DFA: 4 states.
States: S0 (start), S1 (first #), S2 (second #, reading), S3 (accept)
Note: Stops at newline or EOF.

State	Input: #	Input: \n	Input: other	Accept?
S0 (start)	S1	ERROR	ERROR	No
S1 (one #)	S2	ERROR	ERROR	No
S2 (two ##)	S2	S3	S2	No
S3 (accept)	ERROR	ERROR	ERROR	Yes

3.5 String Literal - Minimized DFA

Minimized DFA: 4 states with error handling.
States: S0 (start), S1 (opened), S2 (escape), S3 (accept)
Note: Error state for unterminated strings.

State	Input: "	Input: \\	Input: \n	Input: other	Accept?
S0 (start)	S1	ERR	ERR	ERR	No
S1 (opened)	S3	S2	ERR	S1	No
S2 (escape)	S1	S1	S1	S1	No
S3 (accept)	ERR	ERR	ERR	ERR	Yes

3.6 Boolean Literal - Minimized DFA

Minimized DFA: Combined into single DFA with 10 states.
Note: Distinct paths for "true" (5 states) and "false" (6 states).

State	t	r	u	e	f	a	l	s	Accept?
S0	S1	-	-	-	S5	-	-	-	No
S1	-	S2	-	-	-	-	-	-	No
S2	-	-	S3	-	-	-	-	-	No

S3	-	-	-	S4	-	-	-	-	No
S4 (true)	-	-	-	-	-	-	-	-	Yes
S5	-	-	-	-	-	S6	-	-	No
S6	-	-	-	-	-	-	S7	-	No
S7	-	-	-	-	-	-	-	S8	No
S8	-	-	-	S9	-	-	-	-	No
S9 (false)	-	-	-	-	-	-	-	-	Yes

3.7 Arithmetic Operator - Minimized DFA

Minimized DFA: 5 states to handle both single and double-char operators.
States: S0 (start), S1 (after *), S2-S5 (various accept states)
Note: Special handling for ** (exponentiation) vs single *.

State	Input: +	Input: -	Input: *	Input: /	Input: %	Accept?
S0 (start)	S1	S2	S3	S4	S5	No
S1 (accept +)	ERR	ERR	ERR	ERR	ERR	Yes
S2 (accept -)	ERR	ERR	ERR	ERR	ERR	Yes
S3 (after *)	ERR	ERR	S6	ERR	ERR	Yes
S4 (accept /)	ERR	ERR	ERR	ERR	ERR	Yes
S5 (accept %)	ERR	ERR	ERR	ERR	ERR	Yes
S6 (accept **)	ERR	ERR	ERR	ERR	ERR	Yes