

Automata Design Document

CS4031 — Compiler Construction | Assignment 01 | Spring 2026

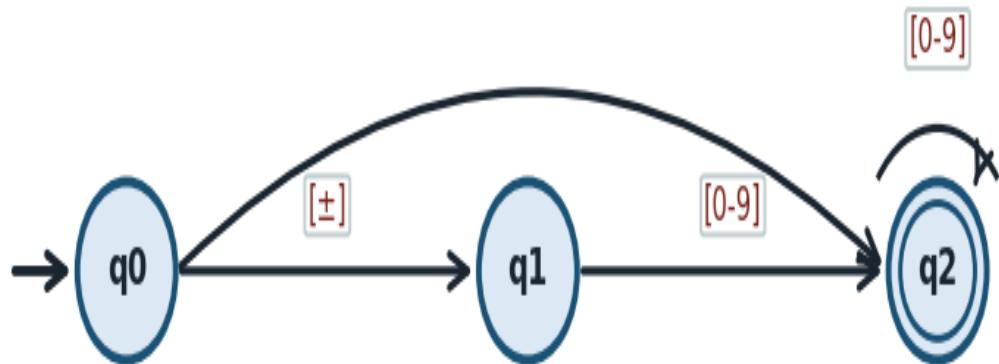
1. Regular Expression Specifications

Token Type	Regular Expression
Keywords	(start finish loop condition declare output input function return break continue else)
Identifier	[A-Z][a-zA-Z0-9_]{0,30}
Integer Literal	[+-]?[0-9]+
Float Literal	[+-]?[0-9]+\.[0-9]{1,6}([eE][+-]?[0-9]+)?
String Literal	"([^\\"\\n] \\\"[^\\ntr])*"
Char Literal	'([^\'\\n] \\'[\\ntr])'
Boolean Literal	(true false)
Arith. Operator	(** [\+\-*/%])
Relational Op	(== != <= >= < >)
Logical Op	(&& \\ ! !)
Assignment Op	(\+= -= *= /= =)
Increment Op	\+\+
Decrement Op	--
Punctuators	[(){}[],;:]
Single Comment	##[^\\n]*
Multi Comment	#*([^*] *+[^*#])**+#
Whitespace	[\\t\\r\\n]+ (skipped)

2. NFA Diagrams

2.1 NFA — Integer Literal

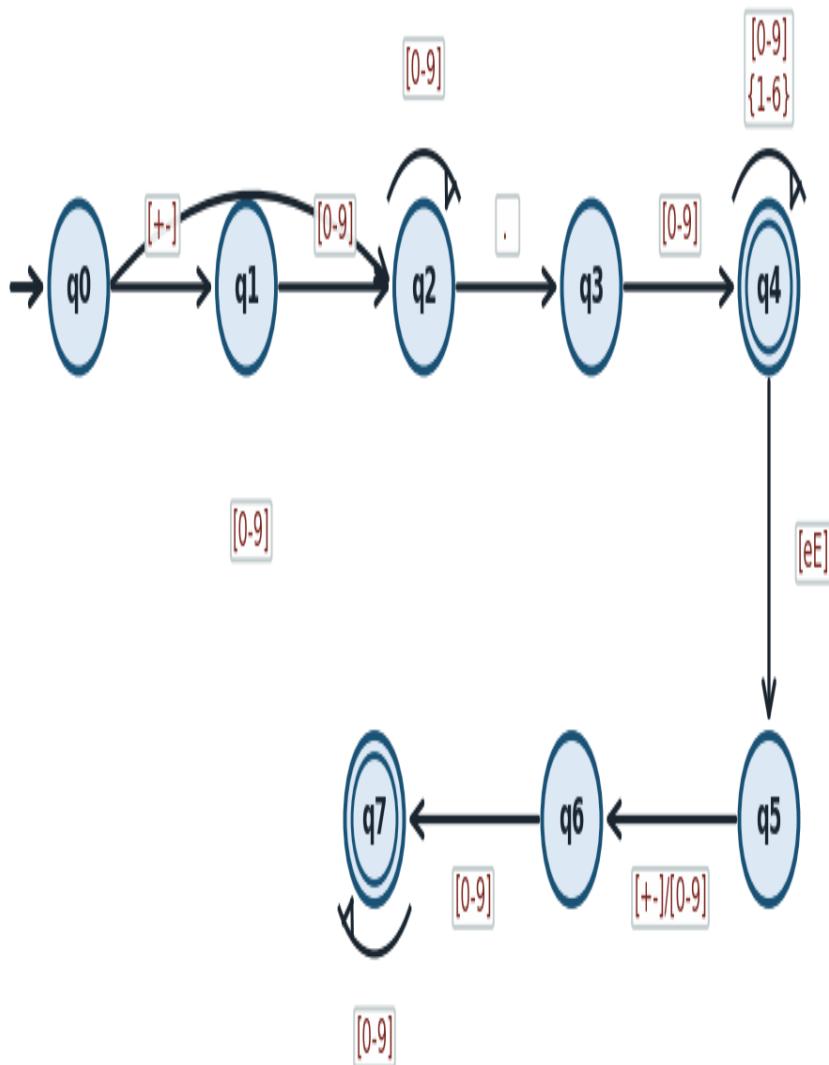
NFA — Integer Literal $[+ -] ? [0-9]^+$



$[0-9]$
q0=start q2=accept $\pm=[+ -]$ double circle=accept state

2.2 NFA — Floating-Point Literal

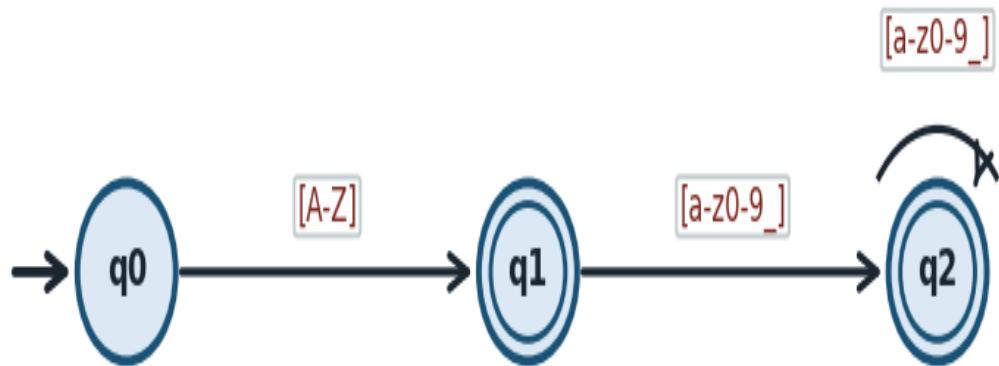
NFA – Floating-Point Literal $[+ -]?[0-9]+.[0-9]\{1,6\}([eE][+ -]?[0-9]+)?$



$q_0 = \text{start}$ $q_4 = \text{accept (no exponent)}$ $q_7 = \text{accept (with exponent)}$ $\{1-6\} = \text{fractional digit count}$

2.3 NFA — Identifier

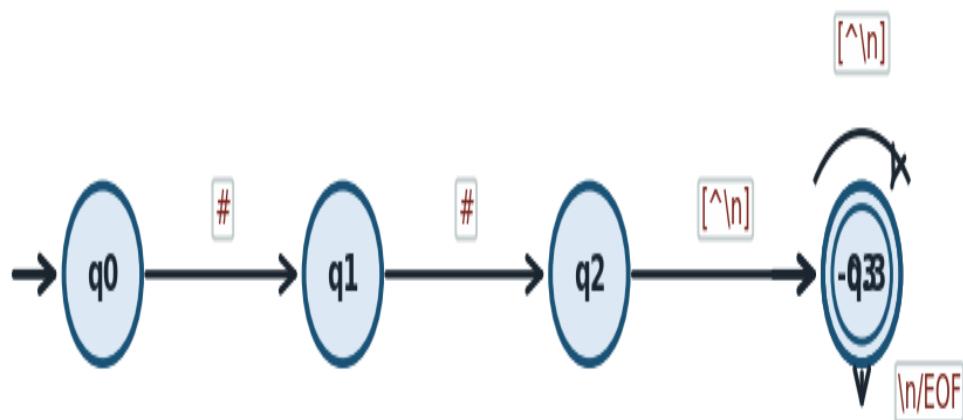
NFA – Identifier [A-Z][a-z0-9_]{0,30}



q_0 =start q_1 =accept (single uppercase valid) q_2 =accept (body chars, max 30 more)

2.4 NFA — Single-Line Comment

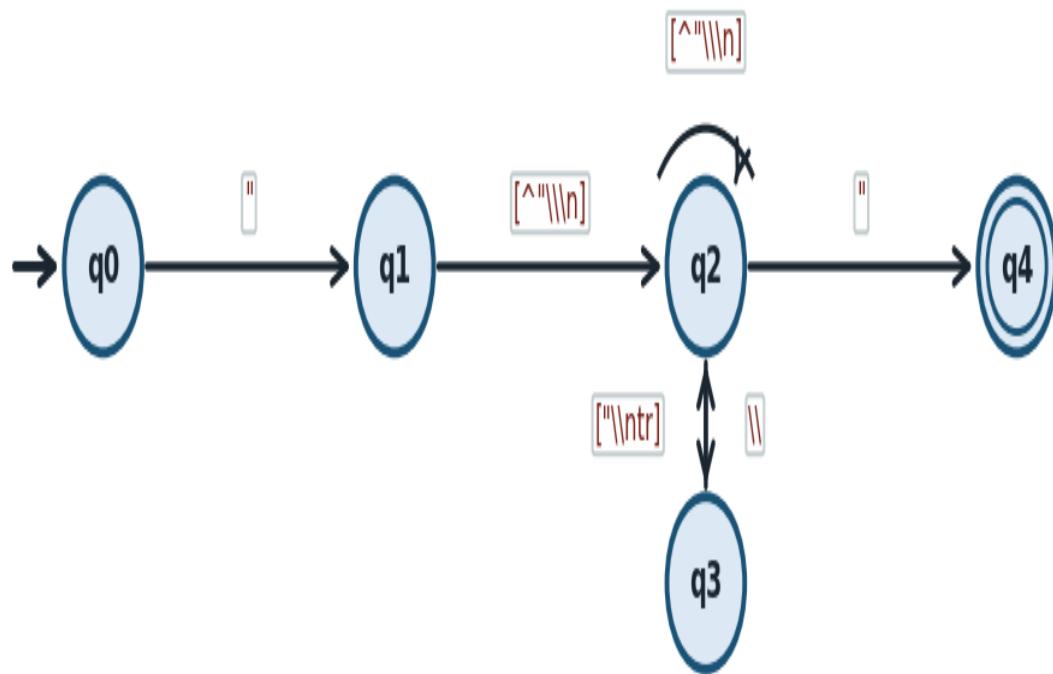
NFA – Single-Line Comment $\#\#[^\n]*$



q_0 =start q_1 =after $\#\#$ q_2 =reading body (accept via ϵ at end) q_3 =accept on \n or EOF

2.5 NFA — String Literal

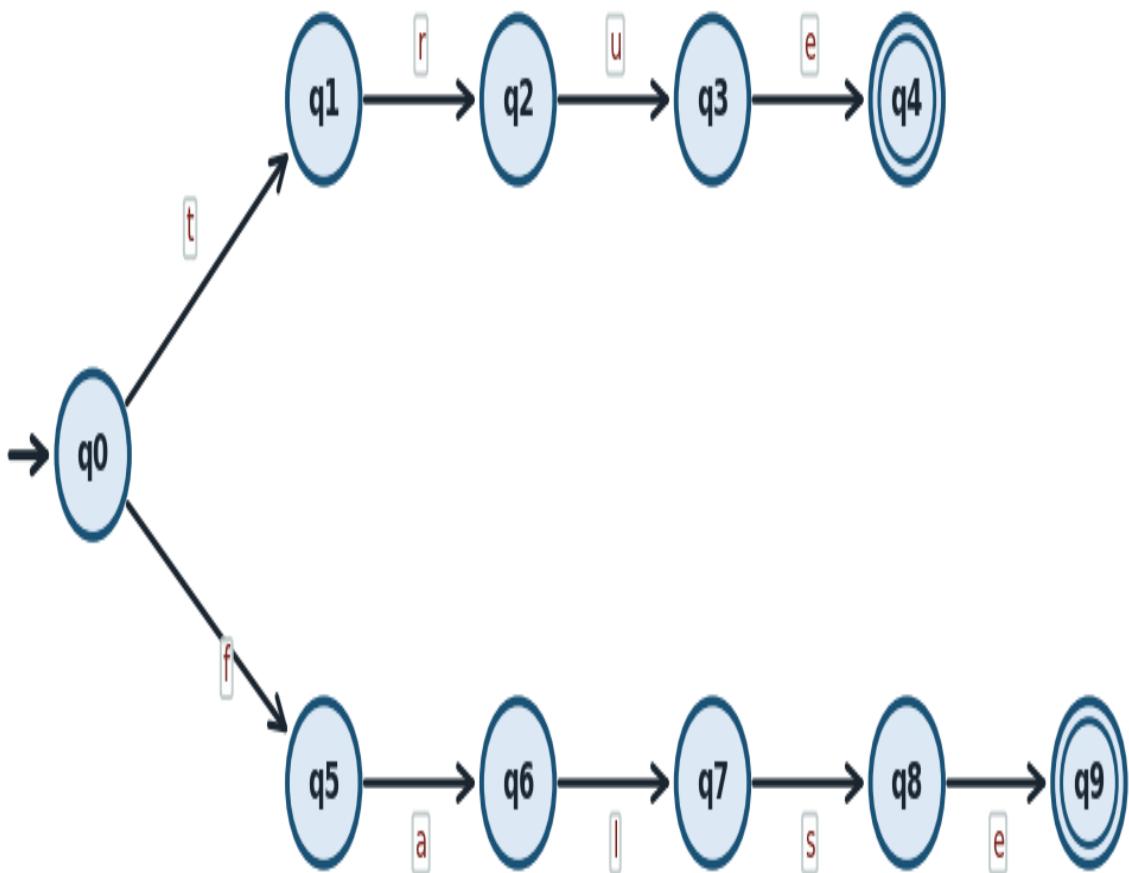
NFA – String Literal ([^\n][\\n])^*



$q_0 = \text{start}$ $q_1 = \text{after "}$ $q_2 = \text{inside string}$ $q_3 = \text{after } | \text{ (escape)}$ $q_4 = \text{accept (closing " reached)}$

2.6 NFA — Boolean Literal

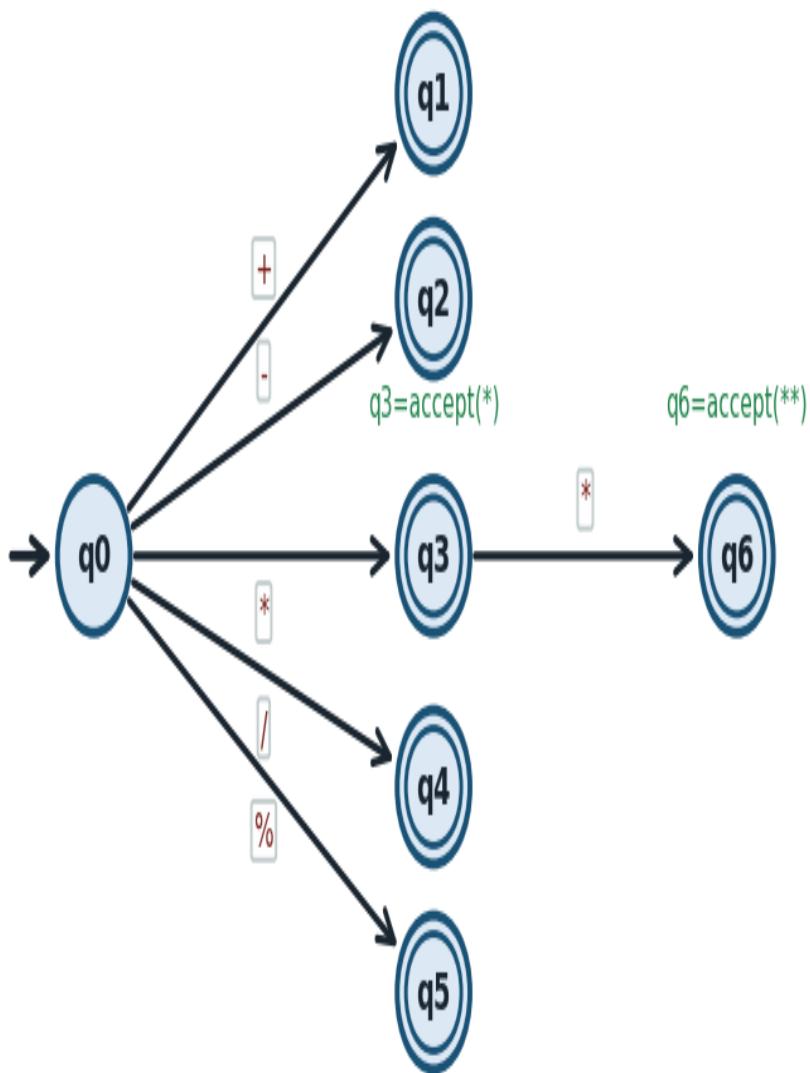
NFA – Boolean Literal (true | false)



q_0 =start q_4 =accept ("true") q_9 =accept ("false") Two non-deterministic branches from q_0

2.7 NFA — Arithmetic Operator

NFA – Arithmetic Operator (**|[+|-*/%])

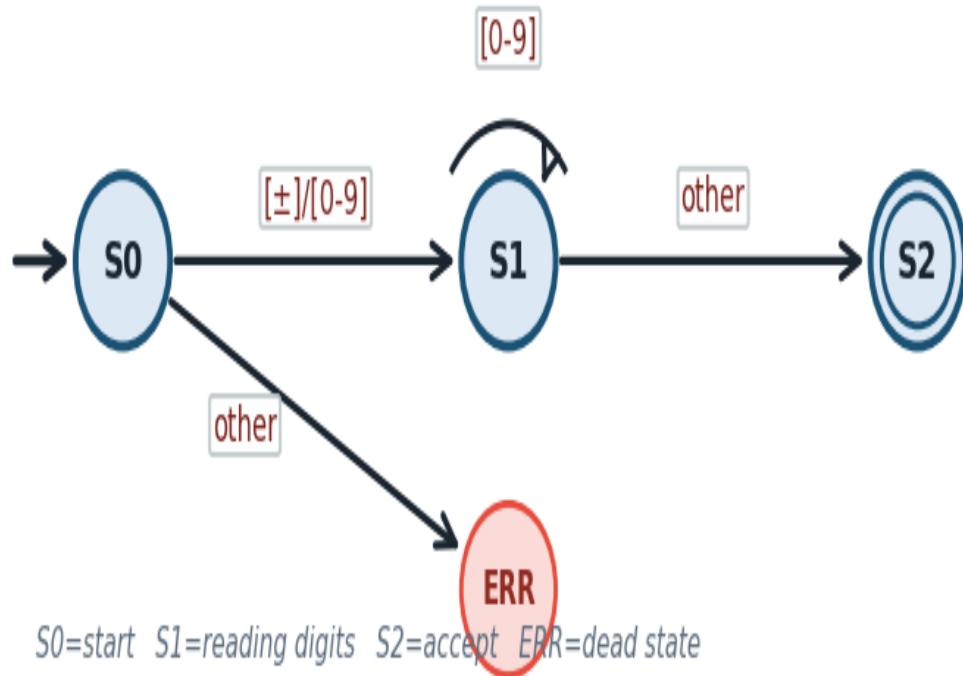


$q0=\text{start}$ $q1=\text{accept}(+)$ $q2=\text{accept}(-)$ $q3=\text{accept}(*) \rightarrow q6 \text{ on } 2\text{nd } *$ $q4=\text{accept}(/)$ $q5=\text{accept}(\%)$ $q6=\text{accept}(**)$

3. DFA Diagrams — Minimized

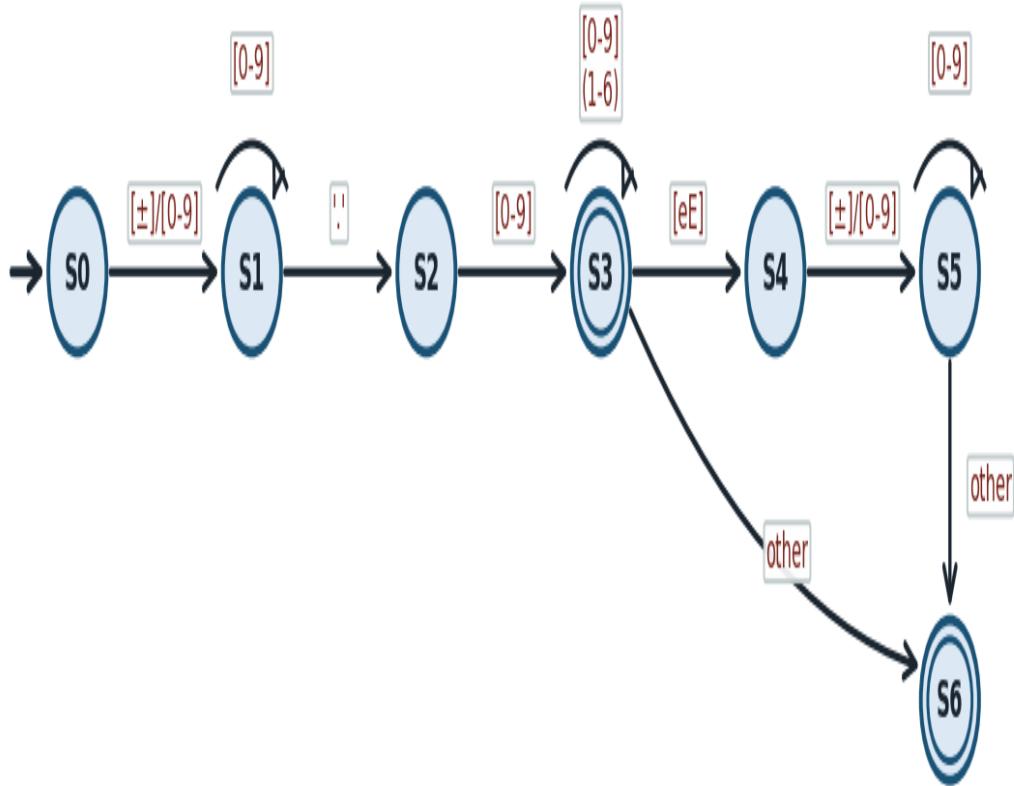
3.1 DFA — Integer Literal

DFA (Minimized) – Integer Literal



3.2 DFA — Floating-Point Literal

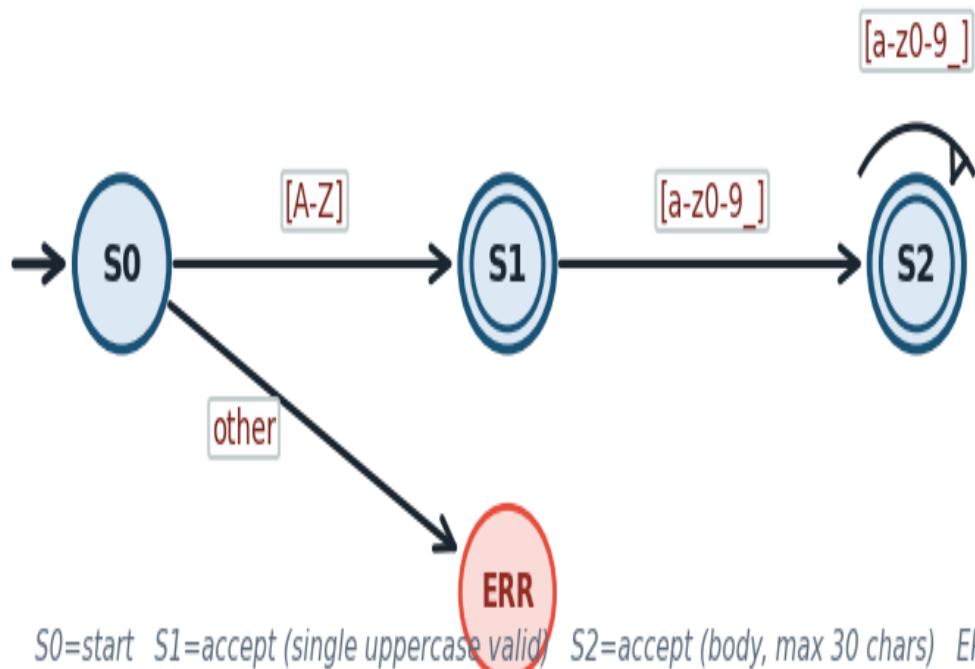
DFA (Minimized) – Floating-Point Literal



S0=start S1=integer part S2=after . S3=fraction (accept=no exp) S4=after e/E S5=exp digits S6=accept

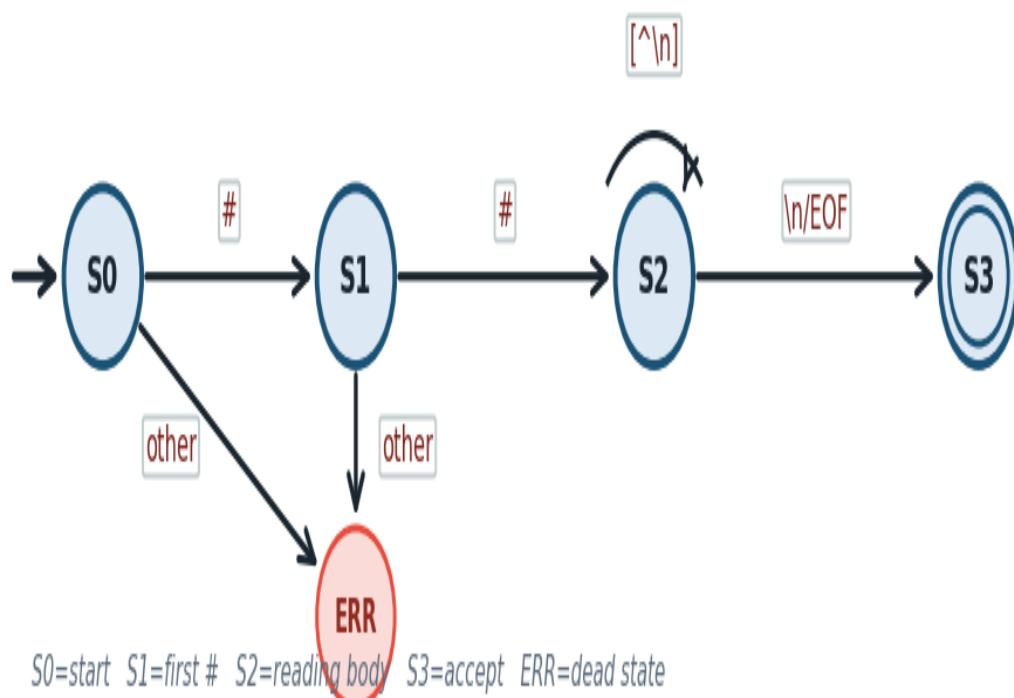
3.3 DFA — Identifier

DFA (Minimized) – Identifier



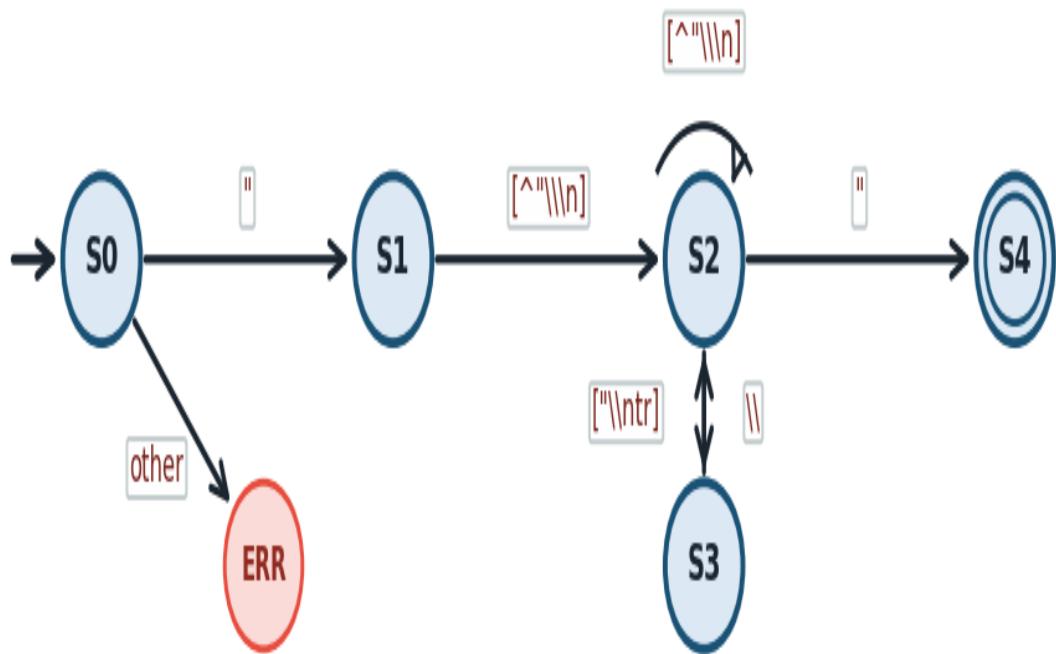
3.4 DFA — Single-Line Comment

DFA (Minimized) – Single-Line Comment



3.5 DFA — String Literal

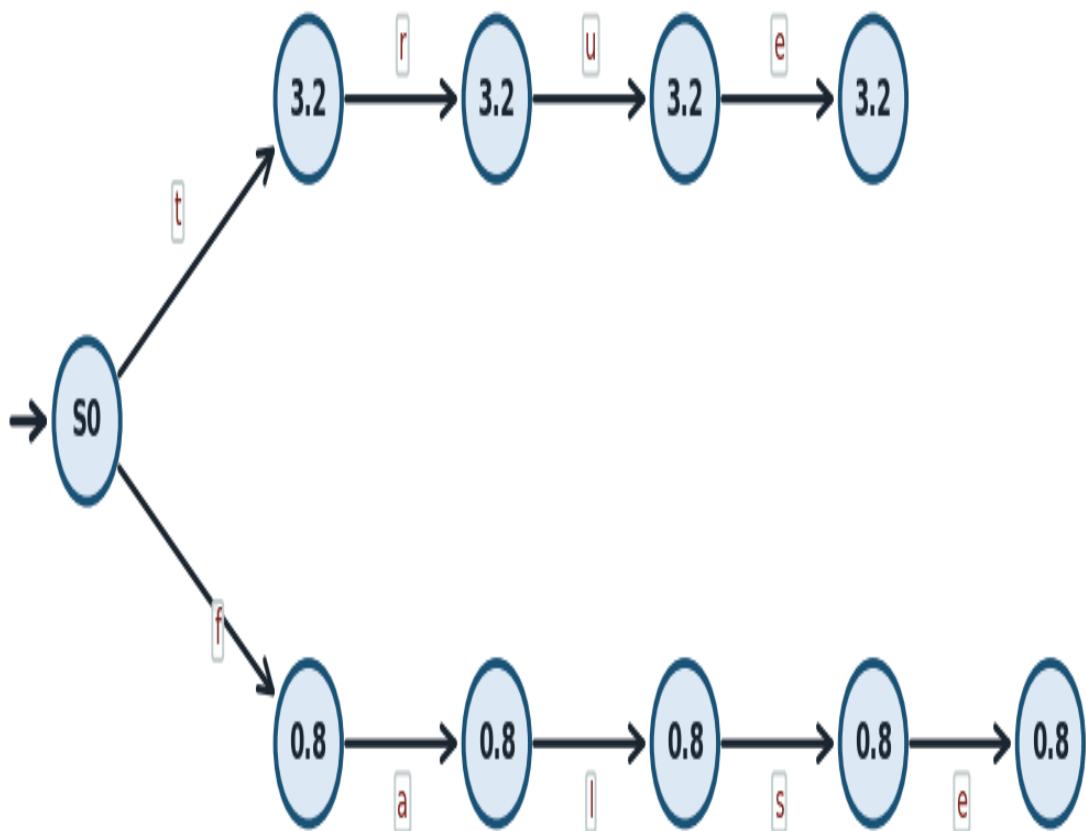
DFA (Minimized) – String Literal



S0=start S1=after " S2=inside string S3=after | escape S4=accept ERR=dead state

3.6 DFA — Boolean Literal

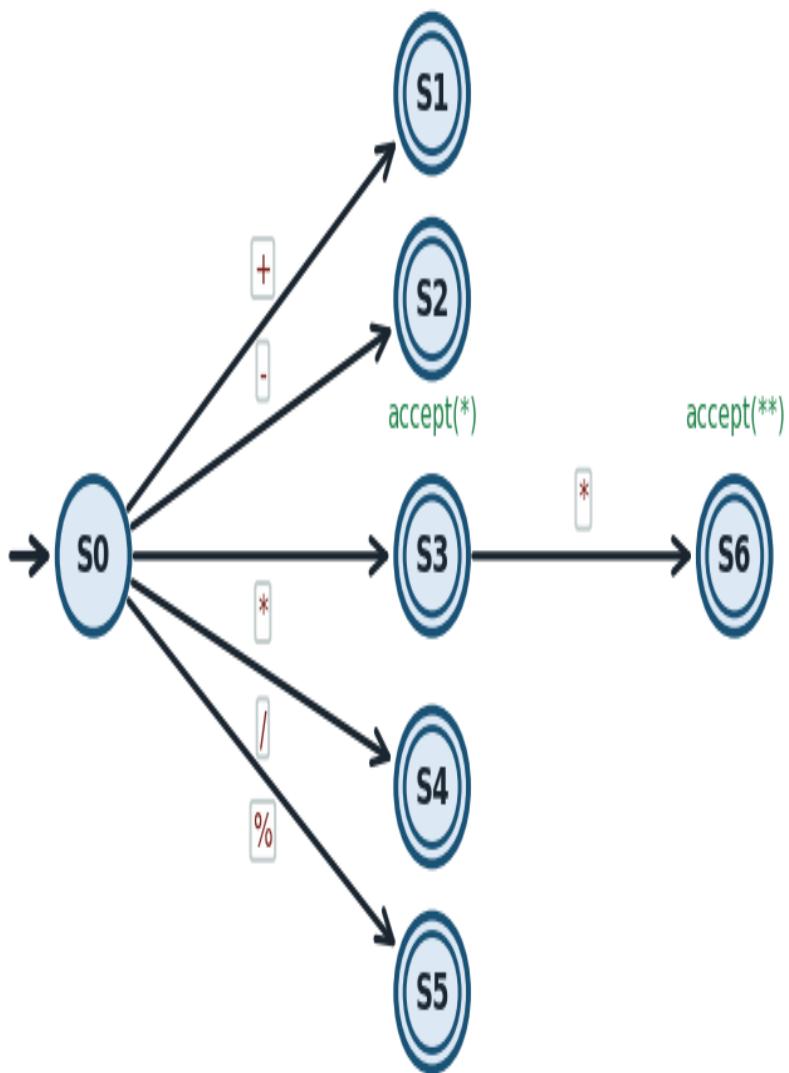
DFA (Minimized) – Boolean Literal (true | false)



S0=start S4=accept ("true") S9=accept ("false") Invalid input from any state → ERR (dead state, not shown)

3.7 DFA — Arithmetic Operator

DFA (Minimized) – Arithmetic Operator $(^{**}|[+|-*/%])$



$S_0 = \text{start}$ $S_1 = \text{accept}(+)$ $S_2 = \text{accept}(-)$ $S_3 = \text{accept}(*)$ also $\rightarrow S_6$ on second $*$ $S_4 = \text{accept}(/)$ $S_5 = \text{accept}(\%)$ $S_6 = \text{accept}(^{**})$

4. DFA Transition Tables

4.1 Integer Literal

State	[±]	[0-9]	other	Accept?
S0 (start)	S1	S1	ERR	—
S1	ERR	S1	S2	—
S2 (accept)	ERR	ERR	ERR	✓
ERR (dead)	ERR	ERR	ERR	—

4.2 Floating-Point Literal

State	[±]	[0-9]	[.]	[eE]	other	Accept?
S0 (start)	S1	S1	ERR	ERR	ERR	—
S1 (int part)	ERR	S1	S2	ERR	ERR	—
S2 (after .)	ERR	S3	ERR	ERR	ERR	—
S3 (frac)	ERR	S3	ERR	S4	S6	✓
S4 (after e/E)	S5	S5	ERR	ERR	ERR	—
S5 (exp)	ERR	S5	ERR	ERR	S6	—
S6 (accept)	ERR	ERR	ERR	ERR	ERR	✓

4.3 Identifier

State	[A-Z]	[a-z0-9_]	other	Accept?
S0 (start)	S1	ERR	ERR	—
S1 (1st char)	ERR	S2	ERR	✓
S2 (body)	ERR	S2	ERR	✓
ERR (dead)	ERR	ERR	ERR	—

4.4 Single-Line Comment

State	#	[^\\n]	\\n / EOF	other	Accept?
S0 (start)	S1	ERR	ERR	ERR	—
S1 (one #)	S2	ERR	ERR	ERR	—
S2 (body)	S2	S2	S3	ERR	—
S3 (accept)	ERR	ERR	ERR	ERR	✓

4.5 String Literal

State	"	\\"	[^"\\\"\\n]	\\n	esc char	Accept?
S0 (start)	"→S1	ERR	ERR	ERR	ERR	—
S1 (opened)	S4	S3	S2	ERR	ERR	—
S2 (inside)	S4	S3	S2	ERR	ERR	—
S3 (escape \\")	S2	S2	S2	S2	S2	—
S4 (accept)	ERR	ERR	ERR	ERR	ERR	✓

4.6 Boolean Literal

State	t	r	u	e	f	a	l	s	Accept?
S0 (start)	S1	—	—	—	S5	—	—	—	—
S1	—	S2	—	—	—	—	—	—	—
S2	—	—	S3	—	—	—	—	—	—
S3	—	—	—	S4	—	—	—	—	—
S4 (accept)	—	—	—	—	—	—	—	—	✓
S5	—	—	—	—	—	S6	—	—	—
S6	—	—	—	—	—	—	S7	—	—
S7	—	—	—	—	—	—	—	S8	—
S8	—	—	—	S9	—	—	—	—	—
S9 (accept)	—	—	—	—	—	—	—	—	✓

4.7 Arithmetic Operator

State	+	-	*	/	%	Accept?
S0 (start)	S1	S2	S3	S4	S5	—
S1 (+)	ERR	ERR	ERR	ERR	ERR	✓
S2 (-)	ERR	ERR	ERR	ERR	ERR	✓
S3 (*)	ERR	ERR	S6	ERR	ERR	✓
S4 (/)	ERR	ERR	ERR	ERR	ERR	✓
S5 (%)	ERR	ERR	ERR	ERR	ERR	✓
S6 (**)	ERR	ERR	ERR	ERR	ERR	✓