CS 348: Module 02: Compilers

Lexical Analysis

Prof. Sukumar Nandi

Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

sukumar@iitg.ac.in

February 10, 2025

Module Objectives

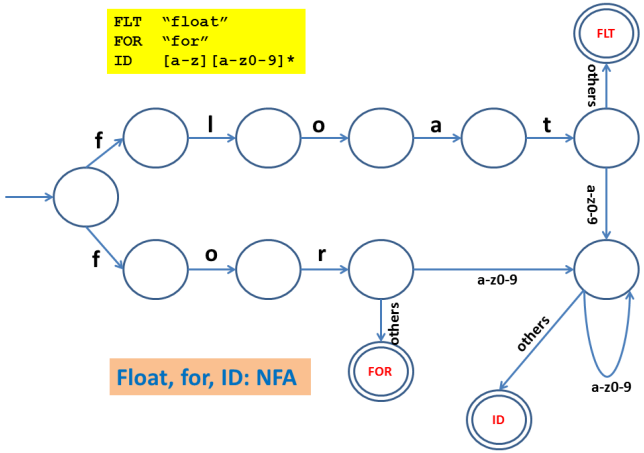
- Understand Lexical Analysis
- Understand Flex Specification

Lexical Analysis Algorithm

- RE[1] for every Token Class
- Convert Regular Expression to an NFA[2]
- Convert NFA to DFA[3]
- Lexical Action for every final state of DFA

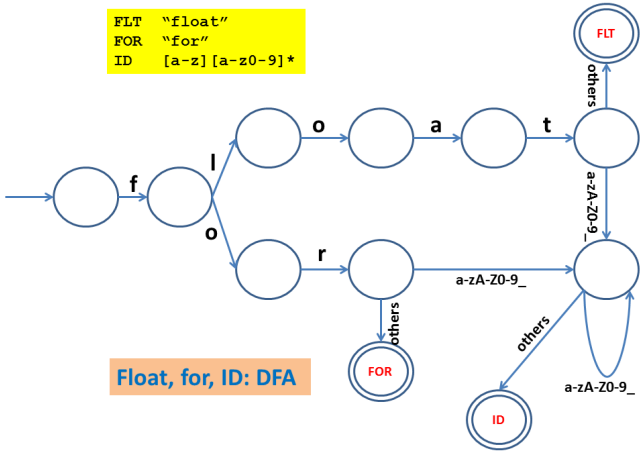
[1] Regular Expression
[2] Non-deterministic Finite Automata
[3] Deterministic Finite Automata

Lexical Analysis Algorithm



NFA Recognizer for a language having keywords "float" and "for" and identifiers starting with 'float' or 'for' (restrictive). Note that transitions on 'others' are look-ahead while all others are consumption.

Lexical Analysis Algorithm



DFA Recognizer for a language having keywords "float" and "for" and identifiers starting with 'float' or 'for' (restrictive). Note that transitions on 'others' are look-ahead while all others are consumption.

Lexical Analysis Rules

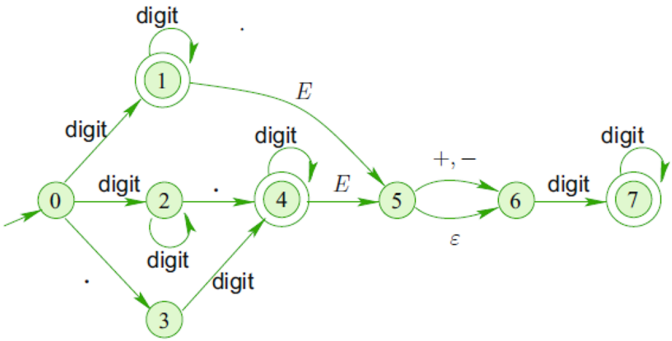
number → digits optFrac optExp
digit → 0 | 1 | 2 | …| 9
digits → digit digit*
optFrac → . digit | ε
optExp → (E (+ | - | ε) digit) | ε

integer and float constants

id → letter (letter | digit)*
letter → A | B | C … | Z | a | b | c … | z
digit → 0 | 1 | 2 | … | 9

Character class

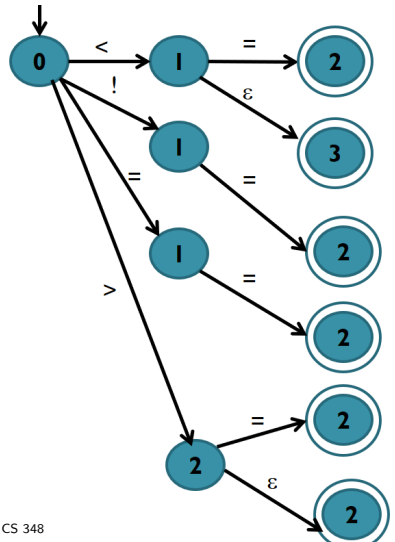
FSM for Integer and Floating Point Constants



Token Representation

Lexemes	Token Name	Attribute Value
Any ws	-	-
if	**if**	-
then	**then**	-
else	**else**	-
Any id	**id**	Pointer to ST
Any number	**number**	Pointer to ST
<	**relop**	LT
<=	**relop**	LE
==	**relop**	EQ
!=	**relop**	NE
>	**relop**	GT
>=	**relop**	GE

FSM for Logical Operators



return (relop, LE)

return (relop, LT)

return (relop, NE)

return (relop, EQ)

return (relop, GE)

return (relop, GT)