CS1602 - COMPILER DESIGN

Lex – Look Ahead Operator and
Conflict Resolution

Session Meta Data

Author	**Dr. B. Bharathi**
Reviewer	
Version Number	1.2
Release Date	3 February 2021

Session Objectives

- Understanding the concept of look ahead operator and conflict resolution in Lex tool

Session Outcomes

- At the end of this session, participants will be able to
  - Understand look ahead operator and conflict resolution in Lex tool.

Agenda

- Lex
  - look ahead operator
  - conflict resolution

Conflict Resolution

- Conflict arises when several prefixes of input matches one or more patterns. This can be resolved by the following:
  - Always prefer a longer prefix than a shorter prefix.
  - If two or more patterns are matched for the longest prefix, then the first pattern listed in lex program is preferred.

Conflict Resolution

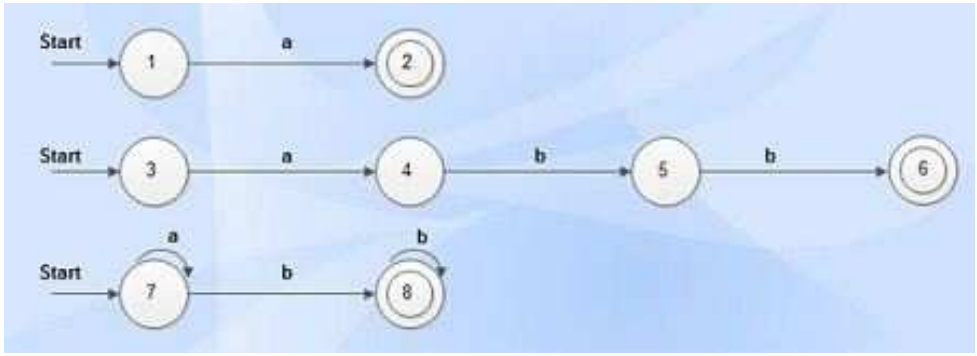
- *a* {action A_1 for pattern P_l}
- *abb* { action A_2 for pattern P_2 }
- *a*b+* { action A_3 for pattern P_3 }

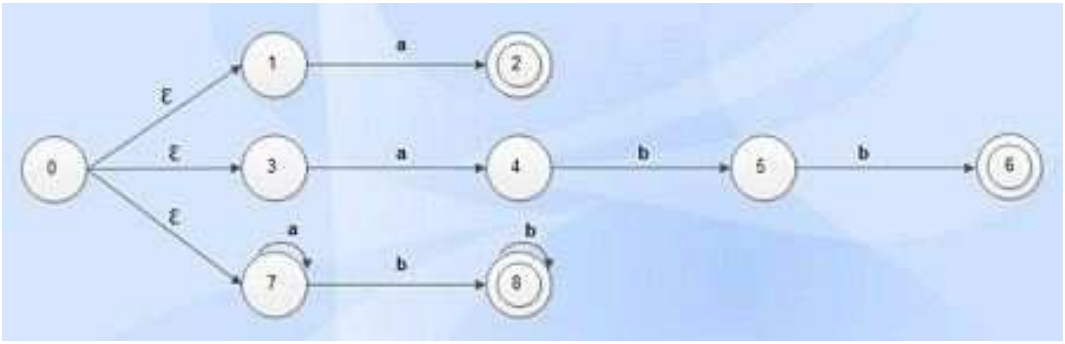
- For string a*bb,* pattern P_2 and pattern p_3 matches. But the pattern P_2 will be taken into account as it was listed first in lex program.

- For string *aabbb* · · · , matches pattern p_3 as it has many prefixes.

- abbbb ?

Design of Lexical Analyzer





8

Lookahead Operator

- Lookahead operator is the additional operator that is read by Lex in order to distinguish additional pattern for a token.
- Lexical analyzer is used to read one character ahead of valid lexeme and then retracts to produce token.
- At times, it is needed to have certain characters at the end of input to match with a pattern. In such cases, slash (/) is used to indicate end of part of pattern that matches the lexeme.

Lookahead Operator

- (eg.) In some languages keywords are not reserved. So the statements
- IF (I, J) = 5 and IF(condition) THEN results in conflict whether to produce IF as an array name or a keyword.
- To resolve this the Lex rule for keyword IF can be written as,
  - IF/\ (.* \) { keyword }
  - IF \(.*\) { array name}

Lookahead Operator

- Rule : {IDENT}/ab
- Input : bracadabra
  - Lex will return bracad for yytext

- {IDENT}
- {IDENT}/ab
- Input : bracadabra
  - Lex will return bracadabra for yytext

Summary

- Conflict resolution
  - Longest prefix
  - Order of rule
- Look ahead operator
  - ‘\’ operator

Check your understanding

1. Rule: a*b/cc, input: aaabcc, what is yytext?
2. Rule: x*/xy, input: xxxy, what is yytext?