Compiler Construction

Lecture # 08

Mr. Usman Wajid

usman.wajid@nu.edu.pk

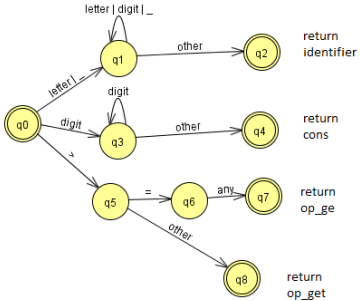
National University
of Computer & Emerging Sciences

Recognition of Tokens

- Lexical analysis can be performed with pattern matching through the use of regular expressions.
- Therefore, a lexical analyzer can be defined and represented as a DFA.
- Recognition of tokens implies a regular expression recognizer.

Example of a Lexical Analyzer

- to build a lexical analyzer for recognizing,
  - identifier
  - $>=$
  - $>$
  - integer_const
- the corresponding DFA will be,



Example of a Lexical Analyzer

- When the current state of automaton is a final state, a match is found in DFA and no transition is enable on the next input character

Example of a Lexical Analyzer

- When the current state of automaton is a final state, a match is found in DFA and no transition is enable on the next input character
- Actions on finding a match,
  1. If the lexeme is valid, then copy it in appropriate place where the parser (Syntax analysis) can access it
  2. Save any necessary scanner state so that scanning can subsequently resume at the right place
  3. Return a value indicating the token found

Assignment # 02: Building a lexical analyzer

1. To build your own programming language, define the following,

Assignment # 02: Building a lexical analyzer

1. To build your own programming language, define the following,
    1. Rules for defining the identifier name
    2. data types (use simple cases for instance int instead of long, short, signed or unsigned)
    3. reserve words
    4. operators
    5. parenthesis
    6. symbol used for ending a statement, use your own instead of semi-colon (;)

Assignment # 02: Building a lexical analyzer

1. To build your own programming language, define the following,
   1. Rules for defining the identifier name
   2. data types (use simple cases for instance int instead of long, short, signed or unsigned)
   3. reserve words
   4. operators
   5. parenthesis
   6. symbol used for ending a statement, use your own instead of semi-colon (;)
2. Construct or draw a single DFA for your own programming language. For fast working use JFLAP tool.

Assignment # 02: Building a lexical analyzer

1. To build your own programming language, define the following,
   1. Rules for defining the identifier name
   2. data types (use simple cases for instance int instead of long, short, signed or unsigned)
   3. reserve words
   4. operators
   5. parenthesis
   6. symbol used for ending a statement, use your own instead of semi-colon (;)
2. Construct or draw a single DFA for your own programming language. For fast working use JFLAP tool.
3. Write a corresponding Lexical code that (in a programming language of your own code) that should capable of,
   1. To identify it a corresponding string is a valid lexeme
   2. to assign a token group to the valid lexemes

Assignment # 02: Building a lexical analyzer

4. your Scanner should perform the following,
   - Your Scanner program should read a source code (on a single statement at least) from a text file
   - Return a lexical error for a invalid character or stream of characters if any
   - assign tokens to the valid lexemes
   - store the tokens into another file such as execl sheet or even simple txt file

NOTE: Submit your assignment in Soft form on the GCR before the deadline