CS 444 - Compiler Construction **Winter 2020**

Lecture 3: January 13th, 2020

Lecturer: Ondřej Lhoták *Notes By: Harsh Mistry*

3.1 Analysis Continued

3.1.1 DFA Recognition

- input: sequence of symbols $w_1, w_2, \ldots, w_n$, DFA $< \sum$, Q, q_0, A
  - q := q_0
  - for each i from 1 to S
    - q := $\delta(q, w_i)$
  - return $q \in A$

3.1.2 Maximal Munch Scanning Pseudo Code

- Input: sequence of symbols$w$ and a DFA which defines language L of valid tokens
- Output: A sequence of **tokens** than each exist in L and concatenate to form w
- Types of Tokens
  - Kind (Id, Num, If, While, etc)
  - Lexene (Substring of w)
- General Algorithm for any language L
  - While there is still input
    1. Find the longest prefix of rest of input that is in L
    2. If no non-empty prefix exists in L, then throw $ERROR$
- Algorithm with DFA for any language L
  - Loop while there is still input
    1. Run DFA on rest of input until DFA gets stuck
    2. Backtrack to last-seen accepting state
    3. If no states were accepting, then throw $ERROR$
    4. Output token
       - Lexene = prefix of rest of input and kind determined by DFA state
    5. Set DFA back to start state

3.1.2.1 Java/Joose Maximal Munch Quirks

- $L = \{-, --, \ldots\}$
- $w = a - -b$
  - Scanner would accept this, but the parser would reject it
  - Scanner would scan it as $a, --, b$, but if it were scanned as $a, -, -, b$ the parser would accept this.
  - $a, -, -, b$ is not valid in java though, so Joos still needs the $--$ token in order reject the decrement operator