, 9, 10

CS 241 - Foundations of Sequential Programs	**Spring 2017**

Lecture 8, 9, 10: May 29th - June 6th, 2017

Lecturer: Kevin Lanctot *Notes By: Harsh Mistry*

8.1 The Compiler

8.1.1 Basic Compilation Steps

- Scanning : Create tokens
- Syntax analysis : create a parse tree
- Semantic analysis : create a symbol tree
- Code generation : generate actual code

8.1.2 Finite Automata

Deterministic Finite Automata (DFA)

- Also known as a deterministic finite state machine (FSM)
- Goal: to be able to clearly and unambiguously recognize all the tokens in a computer language
- The components of a DFA are - A finite set of states (represented by circles) including
  - one start state and
  - (possibly many) accepting states
- A finite set of input symbols known as the alphabet
- A finite set of transitions (represented by edges) from one state to another determined by the input
- The DFA determines if the input is accepted (is a word in the language) or rejected (is not a word in the language)
- In our case: is the input a valid token (and if so, which one)

Features of a DFA

- Easy to trace where you are in the computation
- it is deterministic, i.e. for each state, the transitions out of that state are uniquely labelled (no pair of transitions with the same label)

- there are no explicit error states
- If you are in a state, and the DFA gets an input, say x, such that there is no edge out of that state with that label on it, it is an error.
- The language accepted by the DFA M is called L(M)
- for the previous slide $L(M) = \{bne, beq\}$.

Extensions

- Transducers : for each transition, provide the ability to output a single character

8.2 ϵ-Non-deterministic Finite Automata (ϵ-NFA)

- An ϵ-NFA allows the use of ϵ-transitions (i.e. a transition that happens without consuming any input).
- ϵ-Transitions
  - allow transitions from one state to another without consuming (or requiring) any input
  - makes it easy to join different FAs together
  - can convert an ϵ-NFA to an NFA