COMPUTER SCIENCE 20, SPRING 2014

Module #15 (Recursive Data Types and Structural Induction) - in class

- Meyer, introductory section in Chapter 6, and section 6.1.
  6.2-6.4 (optional) provide concrete examples of recursive data types and structural induction (from real definitions in programming languages).

Executive Summary

1. Recursive definitions. Remember how to recursively define a set S.
   - Base case(s). These define the base cases of S.
   - Constructor case(s). These define new elements of S from previously constructed elements of S or from the base cases of S.
   - Examples:
     - Strings: Let A be a nonempty set called an *alphabet*, whose elements are *characters*. Let the set of strings A^* be defined as follows:
       * Base case: the empty string ϵ is in A^*.
       * Constructor case: If $a \in A$ and $s \in A^*$, then $(a, s) \in A^*$.
       * If $A = \{a, b, \ldots, y, z\}$, the string $(x, (y, (z, \epsilon)))$ would be a member of A^*.
     - String Concatenation: Let the concatenation $s \cdot t$ of strings $s, t \in A^*$ be defined as follows:
       * Base case: $\epsilon \cdot t ::= t$.
       * Constructor case: $(a, s) \cdot t ::= (a, s \cdot t)$.
2. Structural induction. This proof technique is used for proving properties of recursively defined data types.
   - Let S be a inductively defined set. Let $P(x)$ be a property we're trying to prove about S for all $x \in S$.
   - Base case(s): For each base case x in the definition of S, prove $P(x)$.
   - Constructor case(s): For each construction case that uses $x_1, \ldots, x_n \in S$ to construct $x \in S$, show that

     $$P(x_1), \ldots, P(x_n) \implies P(x)$$

     In structural induction proofs, the inductive hypothesis is that the property P holds true for all "sub-cases" of x: $P(x_1), \ldots, P(x_n)$.