

CSI 35 LECTURE NOTES (Ojakian)

Topic 4: Recursive Definitions and Programs

OUTLINE

(References: Wells sections 105-107, 124, 125; Rosen: 5.3, 5.4)

1. Recursive Definitions
 2. Recursive Programs
-

1. Recursive Functions

- (a) To define: Need a basis value. Need Recursive definition
- (b) Example: Define the function $F(n) = 2^n$ without using exponentiation, instead using recursion.
- (c) Example (we'll make up!). Calculate some values, given the following definition.
 - i. $g(\text{[class chooses value]}) = \text{[class chooses number]}$
 - ii. $g(n + 1) = \text{[class chooses expression with arithmetic and } g(n)\text{]}$

2. Programming Recursive Functions

- (a) Program the above using their recursive definitions
(think: start at input value, and move to the basis value)
- (b) Program them iteratively
(think: start at the basis value, and move to the input value)
- (c) For both: Can put in print statements to see operation.

3. Well-defined?

- (a) Missing basis value (run related program).
- (b) Circular recursion (ex: $F(n) = \dots F(n) \dots$; run related program).
- (c) Fibonacci: First do with one basis value. Then fix it.
- (d) Collatz-like functions. Well-defined recursion not always clear! (2 toy examples, then serious one ...)
 - i. Half an even and double an odd.
 - ii. Half an even and minus 1 from an odd.
 - iii. The serious example based on Collatz sequence: Wells 106.1.2 (page 160). Depends on an open question ...

4. Kinds of Problems

- (a) Given a recursive definition, evaluate at various values (as we did above with g).
- (b) Given a known function, find a recursive definition for it (as we did above with F).
 - i. Another Example: Give a recursive definition for factorial, then program it.
 - ii. Another example: Find a recurrence for $C(n, k)$... tricky! (probably skip... but can see Wells Theorem 125.5 on page 192).
- (c) Given a recursive definition, find a non-recursive definition for it (“solving the recurrence”).
 - i. This is key point, since can often express an idea with recursion quickly.
 - ii. However it can be very hard to solve ... (we’ll only do the easy ones!). For example: state solution to Fibonacci.
- (d) Examples of solving a recurrence.
 - i. Solve $f(0) = 0$ and $f(n + 1) = 5 + f(n)$
 - ii. Solve $g(2) = 1$, $g(n + 1) = \begin{cases} g(n) & \text{if } n \text{ even} \\ g(n) + 1 & \text{otherwise} \end{cases}$

5. Recursively defined sets

- (a) Example 5 (contains 3 and closed under $+$)
- (b) Strings over an alphabet (Definition 1, page 370)
 - i. Consider alphabet $\{0, 1\}$
 - ii. Consider alphabet $\{A, B, \dots, Z\}$
- (c) Example: Give a recursive definition of the multiples of 5 (including zero, and negatives)
- (d) Example: Give a recursive definition of the set of positive integers **not** divisible by 5.

6. Exercises

- (a) Recursive calculations. Section 5.3: 1 - 4
- (b) Recursive algorithms. Section 5.4: 1, 2, 8, 9, 10, 11
- (c) Check if valid recursive definition. Section 5.3: 5, 6
- (d) Find recursive definition. Section 5.3: 7 - 11, 20
- (e) Recursively defining a set. Section 5.3: 23, 24ab, 30