CS313K: Logic, Sets and Functions
Fall 2010

Problem Set 3: Recursive Definitions

A *function* is a rule that applies to a number and produces a number. If a function f is applicable to a number x then we say that f is *defined* on x. The result of applying f to x is called the *value* of f at x and denoted by $f(x)$.

If the numbers that f is defined on are arbitrary positive integers then we can think of f as an infinite sequence of values $f(1), f(2), \ldots$.

A *recursive definition* of a function f shows how to compute $f(x)$ if we know the values of f at some points other than x. For example, a function f defined on all positive integers can be characterized by two equations: one for $f(1)$, and the other defining $f(n+1)$ in terms of $f(n)$.

Here is a recursive definition of the sequence B from Problem Set 1:

$$\begin{aligned} &B_1 = 1, \\ &B_{n+1} = B_n + n + 1. \end{aligned}$$

There are two ways to find B_5 using this definition. One is to find first B_2, B_3 and B_4:

$$\begin{aligned} B_2 &= B_1 + 2 = 1 + 2 = 3, \\ B_3 &= B_2 + 3 = 3 + 3 = 6, \\ B_4 &= B_3 + 4 = 6 + 4 = 10, \\ B_5 &= B_4 + 5 = 10 + 5 = 15. \end{aligned}$$

The other possibility is to form a chain of equalities that begins with B_5 and ends with a number:

$$\begin{aligned} B_5 &= B_4 + 5 \\ &= B_3 + 4 + 5 = B_3 + 9 \\ &= B_2 + 3 + 9 = B_2 + 12 \\ &= B_1 + 2 + 12 = B_1 + 14 \\ &= 1 + 14 = 15. \end{aligned}$$

3.1. Here is a recursive definition of the factorial function:

$$\begin{aligned} &0! = 1, \\ &(n+1)! = n! \cdot (n+1). \end{aligned}$$

Show how to calculate 5! using each of the two methods above.

3.2. Consider the function defined by the formulas

$$\begin{aligned} f(0) &= 0, \\ f(n+1) &= f(n) + \tfrac{1}{2n+3}. \end{aligned}$$

Find $f(1)$ and $f(2)$. Define f using sigma-notation, instead of recursion.

3.3. For any nonnegative integer n, let $f(n)$ be the product of all odd numbers from 1 to $2n+1$:

$$f(n) = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n+1).$$

Give a recursive definition of this function, and check that your definition gives the correct value for $f(3)$.

3.4. Express the function from the previous problem in terms of factorials. Check that your formula gives the correct value for $f(3)$.

3.5. Consider the function defined by the formulas

$$\begin{aligned} f(0) &= 0, \\ f(n+1) &= 2f(n) + 1. \end{aligned}$$

Find $f(n)$ for $n = 1, \ldots, 4$. Guess what a formula for $f(n)$ might be. Prove that this formula is correct.

3.6. Consider the function defined by the formulas

$$\begin{aligned} f(0) &= 1, \\ f(n+1) &= 2f(n) + n. \end{aligned}$$

Find $f(n)$ for $n = 1, \ldots, 4$. Guess which values of n satisfy the condition $f(n) > 2^n$. Prove your conjecture.

Recursive definitions can be written in a different format, as a set of cases instead of a set of equations. For instance, the recursive definition of factorial can be rewritten as

$$n! = \begin{cases} 1, & \text{if } n = 0, \\ (n-1)! \cdot n, & \text{otherwise.} \end{cases}$$

Recursive definitions in the case format can be easily translated into Java, for instance:

```
public static int fact(int n){
  if (n==0)
     {return 1;}
  else return fact(n-1)*n;
}
```

3.7. Rewrite the recursive definitions of sequence B and of the function from Problem 5 in the case format.

The *Fibonacci numbers* are defined by the recursive equations

$$\begin{aligned} &f(0) = 0, \\ &f(1) = 1, \\ &f(n+2) = f(n) + f(n+1). \end{aligned}$$

3.8. Find the first 10 Fibonacci numbers. Rewrite the definition of Fibonacci numbers in the case format.

3.9. A function f, defined on nonnegative integers, satisfies the Fibonacci equation

$$f(n+2) = f(n) + f(n+1).$$

If $f(1) = 1$ and $f(4) = 2$ then what is the value of $f(0)$?

3.10. The function f is defined by the condition

$$f(n) = \begin{cases} n - 10, & \text{if } n > 100, \\ f(f(n+11)), & \text{otherwise} \end{cases}$$

for any nonnegative integer n. Find $f(98)$.