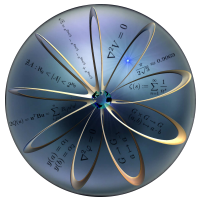

planetmath.org

Math for the people, by the people.

course-of-values recursion

Canonical name	CourseofvaluesRecursion
Date of creation	2013-03-22 19:06:13
Last modified on	2013-03-22 19:06:13
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	11
Author	CWoo (3771)
Entry type	Definition
Classification	msc 03D20
Synonym	course-of-value
Synonym	course-of-values

In defining a function by primitive recursion, the value of the next argument $f(n+1)$ depends only on the value of the current argument $f(n)$. Definition of functions by course-of-values recursion, $f(n+1)$ depends on value(s) of some or all of the preceding arguments $f(n), \ldots, f(0)$. Two very basic examples of definition by course-of-values recursion are

1. (Fibonacci numbers). $F(0) = F(1) = 1$, and $F(n+1) = F(n) + F(n-1)$.

2. $f(0) = 1$, and $f(n+1) = f(n) + f(n-1) + \cdots + f(0)$.

In the first example, we may write $F(n+1) = g(F(n), F(n-1))$, where g is the familiar addition function, a function of two arguments. In other words, value of the current argument of F depends on the values of a fixed number of preceding arguments (in this case, 2). In the second example, value of the current argument of f depends on the values of all of the values of the preceding arguments. Suppose we write $f(n+1) = h(f(n), \ldots, f(0))$, where h is the $(n+1)$-ary addition function. This h is different from g in the sense that g has a fixed arity, whereas the arity of h depends on the argument of f. In other words, as n varies, a "different" h is required! Is it possible to define f via a fixed function as in the first example? Moreover, what is the relation between course-of-values recursion and primitive recursion?

The answer to the first question is yes. To get around the difficulty of "varying arity", we employ the technique of encoding the entire sequence of values of the preceding arguments of f by a "code number", and come up with a function h that depends on the this code number. h then can be used to define f.

Pick an encoding of finite sequences of non-negative integers, preferably a primitive recursive encoding. As usual, we set

$$\langle a_1, \ldots, a_m \rangle$$

as the code, or sequence number of the sequence $a_1, \ldots, a_m$, and if $x = \langle a_1, \ldots, a_m \rangle$, we set $(x)_i := a_i$ if $i \leq m$, and 0 otherwise.

For this entry, we choose the multiplicative encoding of Gödel (see `http://planetmath.org/Exa` entry for more detail) because of its convenience. Then

$$\langle a_1, \ldots, a_m \rangle := p_1^{s(a_1)} \cdots p_m^{s(a_m)},$$

where p_i is the i-th prime number, and $\langle\rangle := 1$.

Definition. For any function $f : \mathbb{N}^{k+1} \to \mathbb{N}$, define the *course-of-values function* $\overline{f}$ of f as follows: $\overline{f}$ has the same arity as f, and for any $\boldsymbol{x} \in \mathbb{N}^k$,

1. $\overline{f}(\boldsymbol{x}, 0) := \langle\rangle$, and
2. $\overline{f}(\boldsymbol{x}, y+1) := \langle f(\boldsymbol{x}, 0), \ldots, f(\boldsymbol{x}, y)\rangle$.

For example, if $F(n)$ is the n-th Fibonacci number, then $\overline{F}(0) = 1$, $\overline{F}(1) = \langle 1\rangle = 4$, and $\overline{F}(2) = \langle 4, 1\rangle = 2^5 \cdot 3^2 = 288$, etc... It is evident that $\overline{F}$ grows very rapidly.

Definition. A function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ is said to be defined by *course-of-values recursion* via function $h : \mathbb{N}^{k+2} \to \mathbb{N}$ if, for any $\boldsymbol{x} \in \mathbb{N}^k$:

$$f(\boldsymbol{x}, y) = h(\boldsymbol{x}, y, \overline{f}(\boldsymbol{x}, y)).$$

The two examples above are defined by course-of-values recursion:

- $F(n) = h(n, \overline{F}(n))$, where $h(x, y) := (y)_x + (y)_{x \dot{-} 1}$.
- $f(n) = h(n, \overline{f}(n))$, where $h(x, y) = \sum_{i=0}^{x \dot{-} 1} (y)_i$.

The second question posed at the beginning can now be answered:

Proposition 1. *f is primitive recursive iff $\overline{f}$ is.*

Proof. By definition, and using multiplicative encoding, we have $\overline{f}(\boldsymbol{x}, 0) := \langle\rangle = 1$, and

$$\overline{f}(\boldsymbol{x}, y+1) := p_1^{f(\boldsymbol{x},0)} \cdots p_{y+1}^{f(\boldsymbol{x},y)} = \overline{f}(\boldsymbol{x}, y) p_{y+1}^{f(\boldsymbol{x},y)}$$

Thus, if f is primitive recursive, so is $\overline{f}$ by primitive recursion. Conversely, if $\overline{f}$ is primitive recursive, so is $f(\boldsymbol{x}, y) = (\overline{f}(\boldsymbol{x}, y+1))_{y+1}$. □

Proposition 2. *If $h : \mathbb{N}^{k+1} \to \mathbb{N}$ is primitive recursive, so is f defined by course-of-values recursion via h.*

Proof. From the proof of the proposition above, we see that

$$\overline{f}(\boldsymbol{x}, y+1) = \overline{f}(\boldsymbol{x}, y) p_{y+1}^{f(\boldsymbol{x},y)} = \overline{f}(\boldsymbol{x}, y) p_{y+1}^{h(\boldsymbol{x},y,\overline{f}(\boldsymbol{x},y))}.$$

Thus, as h is primitive recursive, so is $\overline{f}$ by primitive recursion, and hence f is primitive recursive by the previous proposition. □

Remark. If the value of the next argument of f depends on values of a fixed set of prior arguments, then the primitive recursiveness of f can be proved via mutual recursion. For example, suppose $f(k+1) = h(f(k-1), f(k-3), f(k-4))$. By setting $f_i(n) := f(n+i)$ for $i = 0, 1, 2, 3$, we see that

$$f_i(n+1) = f(n+i+1) = f_{i+1}(n) \quad \text{for } i = 0, 1, 2.$$

Furthermore,

$$\begin{aligned} f_3(n+1) &= f_0(n+4) = f(n+4) = h(f(n+2), f(n+1), f(n)) \\ &= h(f_0(n+2), f_0(n+1), f_0(n)) = h(f_2(n), f_1(n), f_0(n)). \end{aligned}$$

So the f_i are defined by mutual recursion, and if h is primitive recursive, so is each f_i.