

Notes on recursive functions

Eurípides Montagne

School of Electrical Engineering and Computer
Science

University of Central Florida

COP 4020 Programming Languages I

Primitive recursive functions

- A Turing machine is a symbol manipulating device proposed by Alan Turing in 1936 as a model of computation.
- The Von Neumann architecture is a concrete representation of the Turing model of computation.
- Another approach to carry out computation is by means of recursive function theory.

The Church Thesis states that, as computation models, Turing machines and recursive functions are equivalent.

Initial functions

- Recursive function theory is the study of a small initial class of primitive functions which can be used to build a large class of computable functions.
- We can consider that any computable function f can be expressed as a function from \mathcal{N} to \mathcal{N} , where \mathcal{N} stands for non-negative integers.

$$f : \mathcal{N}^m \rightarrow \mathcal{N}^n$$

where

$$n, m \in \mathcal{N}$$

- The initial functions are a set of primitive recursive functions which are accepted as self-evidently computable functions. These functions are: The zero function, The successor function, and the projection function.

Zero Function

The Zero function is a function that always return zero and is defined as:

$$Z(x) = 0 \quad \forall x \in \mathcal{N}$$

Successor function

The Successor function when applied to x returns $x + 1$ and is defined as:

$$S(x) = x + 1 \quad \forall x \in \mathcal{N}$$

Projection function

The projection function selects one of the arguments from the argument list and is defined as:

$$\Pi_k^n(x_1, x_2, x_3, \dots, x_k, \dots, x_n) = x_k \quad \text{with } 1 \leq k \leq n$$

where n stands for the number of arguments and k represents the selected argument.

Computing with functions: Using the initial functions one can build other more complex primitive recursive functions by applying the following rules:

Combination: let f and g be primitive recursive functions defined as:

$$f : \mathcal{N}^k \rightarrow \mathcal{N}^m \quad \text{and} \quad g : \mathcal{N}^k \rightarrow \mathcal{N}^n$$

with $k, m, n \in \mathcal{N}$

The combination of these two functions is expressed as:

$$f \times g : \mathcal{N}^k \rightarrow \mathcal{N}^{m+n}$$

and is defined by:

$$f \times g(\bar{x}) = (f(\bar{x}), g(\bar{x}))$$

where $\bar{x} = (x_1, x_2, x_3, \dots, x_k)$

Example:

$$\Pi_2^3 \times \Pi_3^3 (5, 4, 2) = (\Pi_2^3 (5, 4, 2), \Pi_3^3 (5, 4, 2)) = (4, 2)$$

Composition: let f and g be primitive recursive functions defined as:

$$f : \mathcal{N}^k \rightarrow \mathcal{N}^m \quad \text{and} \quad g : \mathcal{N}^m \rightarrow \mathcal{N}^n$$

with $k, m, n \in \mathcal{N}$

The composition of these two functions is expressed as:

$$g \circ f : \mathcal{N}^k \rightarrow \mathcal{N}^n$$

and is defined by:

$$f \circ g(\bar{x}) = g(f(\bar{x}))$$

where $\bar{x} = (x_1, x_2, x_3, \dots, x_k)$

Example:

$$S(Z(x)) = S(0) = 1$$

Primitive recursion: let g be a primitive recursive function with arity (number of arguments) k , defined as:

$$g : \mathcal{N}^k \rightarrow \mathcal{N}$$

and let h be a primitive recursive function with arity $k + 2$, defined as

$$h : \mathcal{N}^{k+2} \rightarrow \mathcal{N}$$

then the function f with arity $k + 1$ is said to be defined by primitive recursion from g and h if:

$$\begin{aligned} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, y + 1) &= h(\bar{x}, y, f(\bar{x}, y)). \end{aligned}$$

where $\bar{x} = (x_1, x_2, x_3, \dots, x_k)$

The first equation defines the boundary condition and is applied when last argument equals 0; the second one is the recursive equation and is applied when the last argument is different from 0.

Examples of primitive recursion:

Example: The ADD function can be defined using primitive recursion as:

$$\begin{aligned} \text{ADD}(x, 0) &= \Pi_1^1(x) = x \\ \text{ADD}(x, y + 1) &= S(\Pi_3^3(x, y, \text{ADD}(x, y))). \end{aligned}$$

Now we can compute $\text{ADD}(3,2)$ as follows:

$$\begin{aligned} \text{ADD}(3, 2) &= S(\Pi_3^3(3, 1, \text{ADD}(3, 1))) \\ &= S(\Pi_3^3(3, 1, S(\Pi_3^3(3, 0, \text{ADD}(3, 0)))))) \\ &= S(\Pi_3^3(3, 1, S(\Pi_3^3(3, 0, \Pi_1^1(3)))))) \\ &= S(\Pi_3^3(3, 1, S(\Pi_3^3(3, 0, 3)))) \\ &= S(\Pi_3^3(3, 1, S(3))) \\ &= S(\Pi_3^3(3, 1, 4)) \\ &= S(4) \\ &= 5 \end{aligned}$$

The initial functions are primitive recursive and functions built up from the initial functions and a finite application of composition, combination and primitive recursion are also **primitive recursive functions**.

Constructing more primitive recursive functions:

Example: The MULT function can be defined using primitive recursion as:

$$\begin{aligned}MULT(x, 0) &= Z(x) = 0 \\MULT(x, y+1) &= ADD(\Pi_1^3 \times \Pi_3^3(x, y, MULT(x, y))).\end{aligned}$$

MULT can be defined as well in a concise form as:

$$\begin{aligned}MULT(x, 0) &= 0 \\MULT(x, y + 1) &= ADD(x, MULT(x, y)).\end{aligned}$$

Using this short notation we will introduce more recursive functions:

Factorial: can be defined as:

$$\begin{aligned}FACT(0) &= 1 \\FACT(y + 1) &= MULT(y + 1, FACT(y)).\end{aligned}$$

Predecessor: can be defined as:

$$\begin{aligned}PRED(0) &= Z(0) \\PRED(y + 1) &= \Pi_1^2(y, PRED(y)).\end{aligned}$$

We can consider predecessor as the inverse of successor (i.e. $PRED(5)=4$, $Pred(0)=0$); using $PRED$ we can define $MONUS$ (subtraction over the natural numbers).

(Monus) can be defined as:

$$MONUS(x, 0) = \Pi_1^1(x, 0) = x$$

$$MONUS(x, y + 1) = PRED(MONUS(x, y)).$$

If $x \geq y$ $MONUS(x, y)$ is $x - y$,

otherwise $MONUS(x, y) = 0$.

The short notation for the function $MONUS(x, y)$ is $x \dot{-} y$.

Thus the function equality ($EQ(x, y)$) can be defined as:

$$EQ(x, y) = 1 \dot{-} (y \dot{-} x) = (x \dot{-} y)$$

If $EQ(x, y) = 1$ then $x = y$ otherwise $EQ(x, y) = 0$