1.1 Basic Development

1.1.1 Initial primitive recursive functions. The *zero* function Z is such that $Z(x) = 0$; the *successor* function S satisfies the equation $S(x) = x + 1$. For every $n \geq 1$ and $1 \leq i \leq n$, the n-ary *identity* function I^n_i yields its i-th argument, i.e. we have

$$I^n_i(x_1, \dots, x_n) = x_i.$$

We usually write I instead of I^1_1 and we have $I(x) = x$.

1.1.2 Composition. For every $m \geq 1$ and $n \geq 1$, the operator of *composition* takes an m-ary function h and m n-ary functions $g_1, \dots, g_m$ and yields an n-ary function f satisfying:

$$f(\vec{x}) = h(g_1(\vec{x}), \dots, g_m(\vec{x})).$$

1.1.3 Primitive recursion. For every $n \geq 1$, the operator of *primitive recursion* takes an n-ary function g and an $(n{+}2)$-ary function h and yields an $(n{+}1)$-ary function f such that

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

The first argument is called the *recursive argument* whereas the remaining arguments are *parameters*. Note that the definition has at least one parameter.

1.1.4 Primitive recursive functions. A sequence of functions $f_1, \dots, f_k$ is called a *primitive recursive derivation* of a function f if we have

(i) $f = f_k$,
(ii) for each f_i, the function f_i is either one of the basic primitive recursive functions S, Z, and I^n_i, or it is obtained from some of the previous functions f_j, $j < i$, by composition or primitive recursion.

A function is primitive recursive if it has a primitive recursive derivation. A predicate is primitive recursive if its characteristic function is.

1.1.5 Theorem *The class of primitive recursive functions is the smallest class of functions containing the zero, successor and identity functions and which is closed under composition and primitive recursion.*

Proof. This is standard. □

1.1.6 Constant functions are primitive recursive. We first show, by induction on m, that every unary constant function $C_m(x) = m$ is a primitive recursive function. In the base case we have $C_0 = Z$. In the induction step we assume that the function C_m is primitive recursive by IH and define the function C_{m+1} as primitive recursive by unary composition:

$$C_{m+1}(x) = S\,C_m(x).$$

The n-ary constant function $C^n_m(\vec{x}) = m$ is obtained as primitive recursive by the following composition:

$$C^n_m(x_1,\dots,x_n) = C_m\,I^n_1(x_1,\dots,x_n).$$

1.1.7 Explicit definitions of functions. Every explicit definition of a form

$$f(x_1,\dots,x_n) = \tau[x_1,\dots,x_n]$$

can be viewed as a function operator which takes all functions applied in the term τ and which returns the function f as a result. We suppose that the term τ does not apply f and that all its free variables are among the indicated ones.

1.1.8 Theorem *Primitive recursive functions are closed under explicit definitions of functions.*

Proof. By induction on the structure of terms τ we prove that primitive recursive functions are closed under explicit definitions of n-ary functions:

$$f(\vec{x}) = \tau[\vec{x}].$$

If $\tau \equiv x_i$ then the function f is the n-ary identity function I^n_i which is one of the initial primitive recursive functions.

If $\tau \equiv m$ then the function f is the n-ary constant function C^n_m which is primitive recursive by Par. 1.1.6.

If $\tau \equiv h(\tau_1,\dots,\tau_m)$, where h is an m-ary primitive recursive function, then the n-ary functions $g_1,\dots,g_m$ defined explicitly by

$$g_1(\vec{x}) = \tau_1[\vec{x}] \qquad \dots \qquad g_m(\vec{x}) = \tau_m[\vec{x}]$$

are primitive recursive by IH. The function f is obtained as primitive recursive by the following composition

$$f(\vec{x}) = h(g_1(\vec{x}),\dots,g_m(\vec{x})).$$ □

1.1.9 Primitive recursive definitions. Let $\tau_1[\vec{y},\vec{z}]$ and $\tau_2[\vec{y},x,a,\vec{z}]$ be terms containing at most the indicated variables free and neither of them applies the function symbol f. Then the following system of functional equations

$$\begin{aligned} f(\vec{y},0,\vec{z}) &= \tau_1[\vec{y},\vec{z}] \\ f(\vec{y},x+1,\vec{z}) &= \tau_2[\vec{y},x,f(\vec{y},x,\vec{z}),\vec{z}] \end{aligned}$$

has a unique solution f. The definition is called a *primitive recursive definition* of the function f. The definition can be viewed as a function operator which takes all functions applied in the terms τ_1 and τ_2 and yields f as a result.

Note that we do not exclude the case when the parameters $\vec{y}$ or $\vec{z}$ or both are empty. Note also that we do not exclude the case when the term on the right hand side of the second equation does not contain any recursive application. This happens when the variable a does not occur in the term τ_2.

Example. Note that the operator of iteration of unary functions is a special case of primitive recursive definitions. The operator takes a unary function f and yields a binary function $f^n(x)$ satisfying:

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

The function $f^n(x)$ is called the *iteration of f*. As a simple corollary of the next theorem we have that primitive recursive functions are closed under iteration of unary functions.

1.1.10 Theorem *Primitive recursive functions are closed under primitive recursive definitions.*

Proof. Let f be defined by the above primitive recursive definition from primitive recursive functions. We first define explicitly two auxiliary functions

$$\begin{aligned} g(w,\vec{y},\vec{z}) &= \tau_1[\vec{y},\vec{z}] \\ h(x,a,w,\vec{y},\vec{z}) &= \tau_2[\vec{y},x,a,\vec{z}], \end{aligned}$$

which are primitive recursive by Thm. 1.1.8. We then define a p.r. function f_1 by primitive recursion (note that we have at least one parameter!):

$$\begin{aligned} f_1(0,w,\vec{y},\vec{z}) &= g(w,\vec{y},\vec{z}) \\ f_1(x+1,w,\vec{y},\vec{z}) &= h(x,f_1(x,w,\vec{y},\vec{z}),w,\vec{y},\vec{z}). \end{aligned}$$

We derive f as primitive recursive by the following explicit definition

$$f(\vec{y},x,\vec{z}) = f_1(x,0,\vec{y},\vec{z}). \qquad \square$$

1.1.11 Addition is primitive recursive. The addition function $x+y$ is a p.r. function by the following primitive recursive definition:

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

Note that we have $x+y = S^x(y) = S^y(x)$.

1.1.12 Multiplication is primitive recursive. The multiplication function $x \times y$ is a p.r. function by the following primitive recursive definition:

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

1.1.13 Exponentiation is primitive recursive. The exponentiation function x^y is a p.r. function by the following primitive recursive definition:

$$\begin{aligned} x^0 &= 1 \\ x^{y+1} &= xx^y. \end{aligned}$$

1.1.14 Summation function. The summation function $\sum_{i=0}^n i$ is a p.r. function by the following primitive recursive definition:

$$\begin{aligned} \sum_{i=0}^{0} i &= 0 \\ \sum_{i=0}^{n+1} i &= \sum_{i=0}^{n} i + n + 1. \end{aligned}$$

Note that this is an example of *parameterless* primitive recursive definition.

1.1.15 Predecessor function is primitive recursive. The unary predecessor function $x \dot{-} 1$ is primitive recursive by

$$\begin{aligned} 0 \dot{-} 1 &= 0 \\ (x+1) \dot{-} 1 &= x. \end{aligned}$$

Note that this is not explicit definition but rather parameterless primitive recursive definition which ignores recursive applications.

1.1.16 Modified subtraction is primitive recursive. The modified subtraction function $x \dot{-} y$ is a p.r. function by primitive recursive definition:

$$\begin{aligned} x \dot{-} 0 &= x \\ x \dot{-} (y+1) &= (x \dot{-} y) \dot{-} 1. \end{aligned}$$

Note that the last occurrence of the symbol $\dot{-}$ in the second equation belongs to the application of the predecessor function. Note alse that we have $x \dot{-} y = P^y(x)$, where $P(y) = y \dot{-} 1$.