1.2 Basic Development

1.2.1 Constant functions are primitive recursive. We first show, by induction on m, that every unary constant function $C_m(x) = m$ is primitive recursive. In the base case we have $C_0 = Z$ is one of the basic p.r. functions. In the induction step we assume that C_m is primitive recursive by IH and define 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:

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

1.2.2 Explicit definitions of functions. Every explicit definition

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

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

1.2.3 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 basic 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. 1.2.1.

If $\tau \equiv h(\rho_1, \ldots, \rho_m)$, where h is an m-ary primitive recursive function, then the n-ary functions $g_1, \ldots, g_m$ defined explicitly by

$$g_1(\vec{x}) = \rho_1[\vec{x}] \qquad \ldots \qquad g_m(\vec{x}) = \rho_m[\vec{x}]$$

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

$$f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x})). \qquad \square$$

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

$$f(\vec{y},0,\vec{z}) = \rho[\vec{y},\vec{z}]$$
$$f(\vec{y},x+1,\vec{z}) = \tau[\vec{y},x,f(\vec{y},x,\vec{z}),\vec{z}]$$

has a unique solution f. The definition is called *primitive recursive definition* of f. The definition can be viewed as a function operator which takes all functions applied in the terms ρ and τ and yields the function f as a result. Note that we do not exclude the case when the parameters $\vec{y}$ or $\vec{z}$ or both are empty. Also the variable a does not have to occur freely in the term τ.

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

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

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

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

Proof. Let f be defined by the primitive recursive definition as in Par. 1.2.4 from p.r. functions. First we define explicitly two auxiliary functions

$$g(w,\vec{y},\vec{z}) = \rho[\vec{y},\vec{z}]$$
$$h(x,a,w,\vec{y},\vec{z}) = \tau[\vec{y},x,a,\vec{z}],$$

which are primitive recursive by Thm. 1.2.3. Next we define a p.r. function f_1 by primitive recursion (note that we have at least one parameter!):

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

We derive f as primitive recursive by the following explicit definition

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

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

$$0 + y = y$$
$$(x + 1) + y = S(x + y).$$

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

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

$$0 \times y = 0$$
$$(x + 1) \times y = x \times y + y.$$

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

$$x^0 = 1$$
$$x^{y+1} = xx^y.$$

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

$$\sum_{i=0}^{0} i = 0$$
$$\sum_{i=0}^{n+1} i = \sum_{i=0}^{n} i + n + 1.$$

This is an example of *parameterless* primitive recursive definition.

1.2.10 Predecessor function is primitive recursive. The unary predecessor function $x \dot{-} 1$ is defined by the following *explicit definition with monadic discrimination on* x:

$$0 \dot{-} 1 = 0$$
$$(x + 1) \dot{-} 1 = x.$$

The definition has a form of *parameterless* primitive recursive definition, where the term on the right hand side of the second identity is without any recursive application. Hence the predecessor function is primitive recursive.

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

$$x \dot{-} 0 = x$$
$$x \dot{-} (y + 1) = (x \dot{-} y) \dot{-} 1.$$

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