1.6 Recursion with Parameter Substitution

1.6.1 Substitution in parameters. Suppose that

$$\rho[\vec{y}], \tau[x, \vec{z}, \vec{y}], \xi_1[x, \vec{y}], \vec{\sigma}_1[x, \vec{y}], \ldots, \xi_k[x, \vec{y}], \vec{\sigma}_k[x, \vec{y}]$$

are terms with all their free variables indicated not applying the symbol f. Suppose further that $\xi_1[x, \vec{y}] \leq x$, ..., $\xi_k[x, \vec{y}] \leq x$. Consider the $(n{+}1)$-ary function f defined by

$$\begin{aligned} f(0, \vec{y}) &= \rho[\vec{y}] \\ f(x+1, \vec{y}) &= \tau\big[x, f\big(\xi_1[x, \vec{y}], \vec{\sigma}_1[x, \vec{y}]\big), \ldots, f\big(\xi_k[x, \vec{y}], \vec{\sigma}_k[x, \vec{y}]\big), \vec{y}\big]. \end{aligned}$$

We say that f is defined by *course of values recursion recursion with parameter substitution*. The special case when $\xi_i[x, \vec{y}] \equiv x$ for each $i = 1, \ldots, k$ is called *extension by primitive recursion with parameter substitution*.

We will show at the end of this section in Thm. 1.6.20 that p.r. functions are closed under course of values recursion with parameter substitution. The proof proceeds in stages. First, we prove the claim for the schema of primitive recursion with parameter substitution with two recursive applications ($k = 2$) and one parameter ($n = 1$). This is proved in Thm. 1.6.11 by reducing the schema to backward recursion. Next, we extend this result to primitive recursion with arbitrary number of recursive applications (Thm. 1.6.13) and parameters (Thm. 1.6.16). Finally, we show how to reduce course of values recursion with parameter substitution to primitive recursion (Thm. 1.6.20).

Primitive Recursion with Parameter Substitution: Case $k = 2$ and $n = 1$

1.6.2 Introduction. In this subsection we will investigate the schema of primitive recursion with substitution in one parameter ($n = 1$), where only two different recursive applications are allowed ($k = 2$):

$$f(0, y) = \rho[y] \tag{1}$$

$$f(x+1, y) = \tau\big[x, f\big(x, \sigma_1[x, y]\big), f\big(x, \sigma_2[x, y]\big), y\big]. \tag{2}$$

Its admissibility is proved by reducing it to backward recursion.

1.6.3 The outline of the proof. We introduce the function f as primitive recursive by the arithmetization of computation trees for f in which we use as computational rules its defining axioms 1.6.2(1)(2). The computation of the application $f(x, y)$ can be visualized as a binary tree with labels consisting of all applications $f(x_i, y_i)$ which are needed to compute the value $f(x, y)$.

Binary trees are coded as follows. The empty tree is coded by the number 0. A non-empty tree is coded by the number $\langle z, l, r\rangle$, where z is the label of its root node and l and r are the codes of its left and right subtrees, respectively. Note that if t is the code of a non-empty tree then the label of its root node is the first projection of t, i.e. the number $\pi_1(t)$.

We intend to introduce f with the help of its course of values function $\overline{f}$. The function $\overline{f}(x, y)$ yields the computation tree for the application $f(x, y)$, i.e. we would like to have

$$\begin{aligned}\overline{f}(0, y) &= \big\langle f(0, y), 0, 0\big\rangle\\ \overline{f}(x+1, y) &= \Big\langle f(x+1, y), \overline{f}\big(x, \sigma_1[x, y]\big), \overline{f}\big(x, \sigma_2[x, y]\big)\Big\rangle.\end{aligned}$$

The function f can be defined explicitly by $f(x, y) = \pi_1\overline{f}(x, y)$.

1.6.4 Dyadic representation of natural numbers. The *dyadic successors* are unary p.r. functions $x\mathbf{1}$ and $x\mathbf{2}$ explicitly defined by

$$x\mathbf{1} = 2x + 1 \qquad x\mathbf{2} = 2x + 2.$$

It is not difficult to see that every natural number has a unique representation as a *dyadic numeral* which are terms built up from the constant 0 by applications of dyadic successors. Some examples:

$$\begin{aligned} 0 &= 0 & 0\mathbf{12} &= 2(2\times 0+1)+2 = 4\\ 0\mathbf{1} &= 2\times 0+1 = 1 & 0\mathbf{21} &= 2(2\times 0+2)+1 = 5\\ 0\mathbf{2} &= 2\times 0+2 = 2 & 0\mathbf{22} &= 2(2\times 0+2)+2 = 6\\ 0\mathbf{11} &= 2(2\times 0+1)+1 = 3 & 0\mathbf{111} &= 2(2(2\times 0+1)+1)+1 = 7.\end{aligned}$$

1.6.5 Dyadic size. The unary function $|x|$ yields the number of dyadic successors in the dyadic representation of x. The dyadic size function satisfies

$$\begin{aligned}|0| &= 0\\ |x\mathbf{1}| &= |x| + 1\\ |x\mathbf{2}| &= |x| + 1\end{aligned}$$

and it is defined by course of values recursion as a p.r. function by

$$\begin{aligned}|0| &= 0\\ |x+1| &= |x \div 2| + 1.\end{aligned}$$

The following property will be needed later:

$$|x| < n \leftrightarrow x + 1 < 2^n.$$

1.6.6 Dyadic concatenation. The binary function $x \star y$ yields a number which dyadic representation is obtained from the dyadic representations of x and y by appending the digits of y after the digits of x. The dyadic concatenation function $x \star y$ satisfies the identities

$$\begin{aligned} x \star 0 &= x \\ x \star y\mathbf{1} &= (x \star y)\mathbf{1} \\ x \star y\mathbf{2} &= (x \star y)\mathbf{2} \end{aligned}$$

and it is defined explicitly as a p.r. function by

$$x \star y = x2^{|y|} + y.$$

1.6.7 Selector function for recursive arguments. By $\mathbf{x}_i(x)$ we denote the binary function which computes the recursive argument of the recursive application of f at position indexed by the dyadic path i in the computation tree for $f(x, y)$. The function satisfies

$$\begin{aligned} \mathbf{x}_0(x) &= x \\ \mathbf{x}_{i\mathbf{1}}(x) &= \mathbf{x}_i(x) \dot{-} 1 \\ \mathbf{x}_{i\mathbf{2}}(x) &= \mathbf{x}_i(x) \dot{-} 1 \end{aligned}$$

and it is defined explicitly as a p.r. function by

$$\mathbf{x}_i(x) = x \dot{-} |i|\,.$$

1.6.8 Selector function for parameters. The ternary function $\mathbf{y}_i(x, y)$ computes the parameter of the recursive application of f at position indexed by the dyadic path i in the computation tree for $f(x, y)$. The function satisfies

$$\begin{aligned} \mathbf{y}_0(x, y) &= y \\ \mathbf{y}_{i\mathbf{1}}(x, y) &= \sigma_1[\mathbf{x}_{i\mathbf{1}}(x), \mathbf{y}_i(x, y)] \\ \mathbf{y}_{i\mathbf{2}}(x, y) &= \sigma_2[\mathbf{x}_{i\mathbf{2}}(x), \mathbf{y}_i(x, y)] \end{aligned}$$

and it is defined by course of values recursion on i as a p.r. function:

$$\begin{aligned} \mathbf{y}_0(x, y) &= y \\ \mathbf{y}_{i+1}(x, y) &= D\big((i+1) \bmod 2, \sigma_1[\mathbf{x}_{i+1}(x), \mathbf{y}_{i\div 2}(x, y)], \sigma_2[\mathbf{x}_{i+1}(x), \mathbf{y}_{i\div 2}(x, y)]\big). \end{aligned}$$

1.6.9 Course of values subtree function. The ternary function $\overline{f}(x, y).i$ returns the subtree of the computation tree for $f(x, y)$ at position indexed by the dyadic path i. It has the following basic properties:

$$
\begin{aligned}
&|i| = x \to \overline{f}(x,y).i = \big\langle \rho[\mathbf{y}_i(x,y)], 0, 0\big\rangle\\
&|i| < x \to \overline{f}(x,y).i =\\
&\qquad \Big\langle \tau\big[\mathbf{x}_i(x) \dot{-} 1, \pi_1\big(\overline{f}(x,y).i\mathbf{1}\big), \pi_1\big(\overline{f}(x,y).i\mathbf{2}\big), \mathbf{y}_i(x,y)\big],\\
&\qquad\ \overline{f}(x,y).i\mathbf{1}, \overline{f}(x,y).i\mathbf{2}\Big\rangle.
\end{aligned}
$$

The course of values subtree function $\overline{f}(x,y).i$ for f is defined by backward recursion on the difference $2^x \dot{-} 1 \dot{-} i$ as a p.r. function by

$$
\begin{aligned}
&i \geq 2^x \dot{-} 1 \to \overline{f}(x,y).i = \big\langle \rho[\mathbf{y}_i(x,y)], 0, 0\big\rangle\\
&i < 2^x \dot{-} 1 \to \overline{f}(x,y).i = \Big\langle \tau\big[\mathbf{x}_i(x) \dot{-} 1, \pi_1\big(\overline{f}(x,y).i\mathbf{1}\big), \pi_1\big(\overline{f}(x,y).i\mathbf{2}\big), \mathbf{y}_i(x,y)\big],\\
&\qquad\qquad \overline{f}(x,y).i\mathbf{1}, \overline{f}(x,y).i\mathbf{2}\Big\rangle.
\end{aligned}
$$

The *composition* property of the course of values subtree function is

$$|i \star j| \leq x \to \overline{f}(x,y).(i \star j) = \overline{f}\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big).j.$$

1.6.10 Course of values function. The binary function $\overline{f}(x,y)$ returns the computation tree for $f(x,y)$. The function satisfies

$$\overline{f}(0,y) = \big\langle \rho[y], 0, 0\big\rangle \tag{1}$$

$$
\begin{aligned}
\overline{f}(x+1,y) = \Big\langle &\tau\big[x, \pi_1\overline{f}\big(x, \sigma_1[x,y]\big), \pi_1\overline{f}\big(x, \sigma_2[x,y]\big), y\big],\\
&\overline{f}\big(x, \sigma_1[x,y]\big), \overline{f}\big(x, \sigma_2[x,y]\big)\Big\rangle
\end{aligned} \tag{2}
$$

and it is defined explicitly with the help of the course of values subtree function $\overline{f}(x,y).i$ for f as follows

$$\overline{f}(x,y) = \overline{f}(x,y).0.$$

1.6.11 Theorem *Primitive recursive functions are closed under primitive recursion with parameter substitution for the case $k = 2$ and $n = 1$.*

Proof. Let f be defined by primitive recursion with parameter substitution from p.r. functions as in Par. 1.6.2. Let further $\overline{f}$ be its course of values function as in Par. 1.6.10. We claim that we have

$$f(x,y) = \pi_1\overline{f}(x,y). \tag{$\dagger_1$}$$

The function $\overline{f}$ is primitive recursive and so is f.

This is proved by induction on x as $\forall y(\dagger_1)$. The base case follows from

$$f(0,y) = \rho[y] = \pi_1\langle g(y), 0, 0\rangle \overset{1.6.10(1)}{=} \pi_1\overline{f}(0,y).$$

In the induction step take any y and we obtain

$$\begin{aligned} f(x+1,y) &= \tau\big[x, f\big(x,\sigma_1[x,y]\big), f\big(x,\sigma_2[x,y]\big), y\big] \overset{2\times\mathrm{IH}}{=} \\ &= \tau\big[x, \pi_1\overline{f}\big(x,\sigma_1[x,y]\big), \pi_1\overline{f}\big(x,\sigma_2[x,y]\big), y\big] \overset{1.6.10(2)}{=} \pi_1\overline{f}(x+1,y). \end{aligned}$$ □

Primitive Recursion with Parameter Substitution: Case $n = 1$

1.6.12 Introduction. In this subsection we will investigate the schema of primitive recursion with substitution in one parameter ($n = 1$), where arbitrary number of recursive applications are allowed:

$$\begin{aligned} f(0,y) &= \rho[y] \\ f(x+1,y) &= \tau\left[x, f\big(x,\sigma_1[x,y]\big), \ldots, f\big(x,\sigma_{k+1}[x,y]\big), y\right]. \end{aligned}$$

1.6.13 Theorem *Primitive recursive functions are closed under primitive recursion with parameter substitution for the case $n = 1$.*

Proof. It will be supplied later. □

Primitive Recursion with Parameter Substitution

1.6.14 Introduction. In this subsection we will investigate the schema of primitive recursion with arbitrary number of parameters:

$$\begin{aligned} f(0,\vec{y}) &= \rho[\vec{y}] \\ f(x+1,\vec{y}) &= \tau\left[x, f\big(x,\vec{\sigma}_1[x,\vec{y}]\big), \ldots, f\big(x,\vec{\sigma}_k[x,\vec{y}]\big), \vec{y}\right]. \end{aligned}$$

We will prove the admissibility of the schema by reducing it to primitive recursion with substitution in one parameter (see Thm. 1.6.16).

1.6.15 Contraction of parameters. We will reduce the above schema for $n \geq 2$ to a new one for a binary function $\langle f\rangle(x,y)$ so that

$$\langle f\rangle(x,y) = f\left(x, [y]_1^n, \ldots, [y]_n^n\right).$$

The n parameters $\vec{y} \equiv y_1, \ldots, y_n$ are replaced by a single parameter y. We will call the number $y = \langle\vec{y}\rangle \equiv \langle y_1, \ldots, y_n\rangle$ the *contraction* of the numbers $\vec{y}$.

The *contraction* function $\langle f\rangle(x,y)$ is defined by primitive recursion on x with substitution in the (only) parameter y as a p.r. function by

$$\begin{aligned}\langle f\rangle(0,y) &= \rho\big[[y]_1^n,\dots,[y]_n^n\big]\\ \langle f\rangle(x+1,y) &= \tau\Big[x,\big\langle f\big\rangle\Big(x,\big\langle\vec\sigma_1\big[x,[y]_1^n,\dots,[y]_n^n\big]\big\rangle\Big),\dots,\\ &\qquad \big\langle f\big\rangle\Big(x,\big\langle\vec\sigma_k\big[x,[y]_1^n,\dots,[y]_n^n\big]\big\rangle\Big),[y]_1^n,\dots,[y]_n^n\Big].\end{aligned}$$

1.6.16 Theorem *Primitive recursive functions are closed under primitive recursion with parameter substitution.*

Proof. Let f be defined by primitive recursion with parameter substitution from p.r. functions as in Par. 1.6.14, where $n \geq 2$.[1] Let further $\langle f\rangle$ be its contraction function as in Par. 1.6.15. We claim that we have

$$f(x,\vec y) = \big\langle f\big\rangle\big(x,\langle\vec y\rangle\big). \tag{$\dagger_1$}$$

The function $\langle f\rangle$ is primitive recursive and so is f.

This is proved by induction on x as $\forall y(\dagger_1)$. The base case follows from

$$f(0,\vec y) = \rho[\vec y] = \rho\big[[\langle\vec y\rangle]_1^n,\dots,[\langle\vec y\rangle]_n^n\big] = \big\langle f\big\rangle\big(0,\langle\vec y\rangle\big).$$

In the induction step take any $\vec y$ and we obtain

$$\begin{aligned}f(x+1,\vec y) &= \tau\big[x,f\big(x,\vec\sigma_1[x,\vec y]\big),\dots,f\big(x,\vec\sigma_k[x,\vec y]\big),\vec y\big] \overset{k\times\mathrm{IH}}{=}\\ &= \tau\Big[x,\big\langle f\big\rangle\Big(x,\big\langle\vec\sigma_1[x,\vec y]\big\rangle\Big),\dots,\big\langle f\big\rangle\Big(x,\big\langle\vec\sigma_k[x,\vec y]\big\rangle\Big),\vec y\Big] =\\ &= \tau\Big[x,\big\langle f\big\rangle\Big(x,\big\langle\vec\sigma_1\big[x,[\langle\vec y\rangle]_1^n,\dots,[\langle\vec y\rangle]_n^n\big]\big\rangle\Big),\dots,\\ &\qquad \big\langle f\big\rangle\Big(x,\big\langle\vec\sigma_k\big[x,[\langle\vec y\rangle]_1^n,\dots,[\langle\vec y\rangle]_n^n\big]\big\rangle\Big),[\langle\vec y\rangle]_1^n,\dots,[\langle\vec y\rangle]_n^n\Big] =\\ &= \big\langle f\big\rangle\big(x+1,\langle\vec y\rangle\big).\end{aligned}$$

□

Course of Values Recursion with Parameter Substitution

1.6.17 Introduction. In this subsection we will investigate the general schema of course of values recursion with parameter substitution:

$$f(0,\vec y) = \rho[\vec y] \tag{1}$$

$$f(x+1,\vec y) = \tau\big[x,f\big(\xi_1[x,\vec y],\vec\sigma_1[x,\vec y]\big),\dots,f\big(\xi_k[x,\vec y],\vec\sigma_k[x,\vec y]\big),\vec y\big], \tag{2}$$

where

[1] The case $n = 0$ is obvious and the case $n = 1$ follows from Thm. 1.6.13.

$$\xi_1[x,\vec{y}] \le x \quad \dots \quad \xi_k[x,\vec{y}] \le x. \tag{3}$$

We will prove the admissibility of the schema by reducing it to primitive recursion with parameter substitution (see Thm. 1.6.20).

1.6.18 Approximation function. We will introduce the function $f(x,\vec{y})$ as primitive recursive with the help of its *approximation* function $f^+(z,x,\vec{y})$. The additional argument z plays the role of the depth of recursion counter. It estimates the depth of recursion needed to compute the value $f(x,\vec{y})$. If z is sufficiently large then we have $f^+(z,x,\vec{y}) = f(x,\vec{y})$. As we will see below every number $z > x$ gives us sufficient estimation of the depth of recursion. This will allow us to define f explicitly by $f(x,\vec{y}) = f^+(x+1,x,\vec{y})$.

The $(n+2)$-ary function $f^+(z,x,\vec{y})$ satisfies

$$f^+(0,x,\vec{y}) = 0 \tag{1}$$

$$f^+(z+1,0,\vec{y}) = \rho[\vec{y}] \tag{2}$$

$$f^+(z+1,x+1,\vec{y}) = \tau\Big[x, f^+\big(z,\xi_1[x,\vec{y}],\vec{\sigma}_1[x,\vec{y}]\big),\dots, f^+\big(z,\xi_k[x,\vec{y}],\vec{\sigma}_k[x,\vec{y}]\big),\vec{y}\Big], \tag{3}$$

and it is defined by primitive recursion on z with substitution in the parameters $x,\vec{y}$ as a p.r. function by

$$f^+(0,x,\vec{y}) = 0$$

$$f^+(z+1,x,\vec{y}) = D\Big(x, \tau\Big[x\dot{-}1, f^+\big(z,\xi_1[x\dot{-}1,\vec{y}],\vec{\sigma}_1[x\dot{-}1,\vec{y}]\big),\dots, f^+\big(z,\xi_k[x\dot{-}1,\vec{y}],\vec{\sigma}_k[x\dot{-}1,\vec{y}]\big),\vec{y}\Big], \rho[\vec{y}]\Big).$$

1.6.19 Monotonicity of the approximation function. We have

$$x < z_1 \wedge x < z_2 \to f^+(z_1,x,\vec{y}) = f^+(z_2,x,\vec{y}). \tag{1}$$

The property asserts that the application $f^+(z,x,\vec{y})$ yields the same result for every number $z > x$.

Proof. The property is proved by induction on z_1 as $\forall x \forall \vec{y} \forall z_2(1)$. □

1.6.20 Theorem *Primitive recursive functions are closed under course of values recursion with parameter substitution.*

Proof. Let f be defined by course of values recursion with parameter substitution from p.r. functions as in Par. 1.6.17. Let further f^+ be its approximation function as in Par. 1.6.18. We claim that we have

$$f(x,\vec{y}) = f^+(x+1,x,\vec{y}). \qquad (\dagger_1)$$

The function f^+ is primitive recursive and so is f.

The property is proved by complete mathematical induction on x as $\forall y(\dagger_1)$. So take any $\vec{y}$ and consider two cases. If $x = 0$ then we have

$$f(0,\vec{y}) = \rho[\vec{y}] \overset{1.6.18(2)}{=} f^+(0+1,0,\vec{y}).$$

If $x = v+1$ for some v then $\xi_i[v,\vec{y}] < v+1$ for each $i = 1,\dots,k$ by 1.6.17(3). We then obtain

$$\begin{aligned}
f(v+1,\vec{y}) &= \tau\Big[v, f\big(\xi_1[v,\vec{y}],\vec{\sigma}_1[v,\vec{y}]\big),\dots,f\big(\xi_k[v,\vec{y}],\vec{\sigma}_k[v,\vec{y}]\big),\vec{y}\Big] \overset{k\times\mathrm{IH}}{=} \\
&= \tau\Big[v, f^+\big(\xi_1[v,\vec{y}]+1,\xi_1[v,\vec{y}],\vec{\sigma}_1[v,\vec{y}]\big),\dots, \\
&\qquad f^+\big(\xi_k[v,\vec{y}]+1,\xi_k[v,\vec{y}],\vec{\sigma}_k[v,\vec{y}]\big),\vec{y}\Big] \overset{1.6.19(1)}{=} \\
&= \tau\Big[v, f^+\big(v+1,\xi_1[v,\vec{y}],\vec{\sigma}_1[v,\vec{y}]\big),\dots, \\
&\qquad f^+\big(v+1,\xi_k[v,\vec{y}],\vec{\sigma}_k[v,\vec{y}]\big),\vec{y}\Big] \overset{1.6.18(3)}{=} \\
&= f^+(v+1+1,v+1,\vec{y}).
\end{aligned}$$

□