1.7 Recursion with Parameter Substitution

1.7.1 Introduction. In this section we will investigate recursive definitions for which parameters change in recursive applications. This new scheme is called recursion with parameter substitution.

1.7.2 Example of primitive recursion with parameter substitution. Our first example of recursion with parameter substitution is the efficient implementation of the sequence of Fibonacci (see Par. 1.5.2). Recall that the fast program for the Fibonacci function

$$\begin{aligned} \mathrm{fib}(0) &= 0 \\ \mathrm{fib}(n+1) &= g(n,1,0) \end{aligned}$$

is obtained with the help of the auxiliary ternary function g defined by

$$\begin{aligned} g(0,a,b) &= a \\ g(n+1,a,b) &= g(n,a+b,a). \end{aligned}$$

Note that the value $g(n+1,a,b)$ depends on the value $g(n,a+b,a)$ from the previous stage, where the terms $a+b$ and a has been substituted for the parameters a and b, respectively. We say that the function $g(n,a,b)$ is defined by primitive recursion on n with substitution in the parameters a and b.

1.7.3 Example of course of values recursion with parameter substitution. The second example is the algorithm of Euclid for computing of the greatest common divisor of two numbers:

$$\begin{aligned} \gcd(0,y) &= y \\ \gcd(x+1,y) &= \gcd\big(y \bmod (x+1), x+1\big), \end{aligned}$$

which relies on the following property of divisibility:

$$y \neq 0 \to z \mid x \wedge z \mid y \leftrightarrow z \mid y \wedge z \mid x \bmod y.$$

These two equations have a form of a course of values recursive definition because the first argument decreases in the recursive application:

$$y \bmod (x+1) \leq x < x+1.$$

The only difference between the above definition and the one discussed in Sect. 1.5 is that in this case the parameter changes in recursion. Namely, the term $x+1$ is substituted for the parameter y in the second equation. We say that the function $\gcd(x,y)$ is defined by course of values recursion on x with substitution in the parameter y.

1.7.4 Recursion with 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 which do not apply f with all their free variables indicated s.t.

$$T \vdash \xi_1[x, \vec{y}] \leq x \quad \ldots \quad T \vdash \xi_k[x, \vec{y}] \leq x.$$

Consider the $(n{+}1)$-ary function f satisfying

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

We say that f is defined by *by (course of values) recursion with parameter substitution*. The definition can be viewed as a function operator which takes all functions applied in the terms $\rho, \tau, \xi_1, \vec{\sigma}_1, \ldots, \xi_k, \vec{\sigma}_k$ and yields the function f as a result.

The assertion that the class of primitive recursive functions is closed under the operator of course of values recursion with parameter substitution will be proved at the end of this section. The proof proceeds in stages. First, we prove the result for the scheme of primitive recursion with parameter substitution with two recursive applications ($k = 2$) and one parameter ($n = 1$). This is shown in Thm. 1.7.21 by reducing the scheme to backward recursion. Next, we extend the result to primitive recursion with arbitrary number of recursive applications (Thm. 1.7.24) and parameters (Thm. 1.7.27). Finally, we show how to reduce course of values recursion with parameter substitution to primitive recursion (Thm. 1.7.31).

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

1.7.5 Introduction. In this subsection we will investigate the scheme of primitive recursion with substitution in one parameter ($n = 1$), where only one recursive applications are allowed ($k = 1$). The fact that p.r. functions are closed under such recursion will be proved in Thm. 1.7.11.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following recursion with parameter substitution

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

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

from p.r. functions. We claim that f is also primitive recursive. We will prove this fact by reducing the scheme to backward recursion.

1.7.6 The outline of of the proof. We will introduce the function f as primitive recursive by the arithmetization of computation sequences for f in which we use as computational rules its defining axioms 1.7.5(1)(2). The computation of the application $f(x,y)$ can be visualized as a finite sequence with labels consisting of all applications $f(x_i,y_i)$ which are needed to compute the value $f(x,y)$.

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 sequence for the application $f(x,y)$, i.e. we would like to have

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

The function f can then be defined explicitly by

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

as primitive recursive.

1.7.7 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 i in the computation sequence for $f(x,y)$. The function is defined explicitly as a p.r. function by

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

1.7.8 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 i in the computation sequence for $f(x,y)$. The function $\mathbf{y}_i(x,y)$ is defined by primitive recursion on i as primitive recursive by

$$\begin{aligned}\mathbf{y}_0(x,y) &= y\\ \mathbf{y}_{i+1}(x,y) &= \sigma[\mathbf{x}_{i+1}(x),\mathbf{y}_i(x,y)].\end{aligned}$$

1.7.9 Partial course of values sequence. The ternary function $\overline{f}(x,y).i$ returns the subsequence of the computation sequence for $f(x,y)$ starting from the position indexed i. The function is defined by backward recursion on the difference $x \dot{-} i$ as a p.r. function by

$$\overline{f}(x,y).i = \begin{cases}\big\langle \rho[\mathbf{y}_i(x,y)],0\big\rangle & \text{if } i \geq x,\\ \Big\langle \tau\big[\mathbf{x}_i(x) \dot{-} 1,\overline{f}(x,y).(i+1)[0],\mathbf{y}_i(x,y)\big],\overline{f}(x,y).(i+1)\Big\rangle & \text{if } i < x.\end{cases}$$

The auxiliary course of values function satisfies

$$i \leq x \to \overline{f}(x,y).i = \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).0. \tag{1}$$

Proof. By backward induction on the difference $x \dot{-} i$ as $\forall y(1)$. So take any i, x, y such that $i \leq x$ and consider two cases. If $i = x$ then we have

$$\begin{aligned}\overline{f}(x,y).x &= \Big\langle g\big(\mathbf{y}_x(x,y)\big), 0\Big\rangle = \Big\langle g\big(\mathbf{y}_0(0,\mathbf{y}_x(x,y))\big), 0\Big\rangle = \\ &= \overline{f}\big(0,\mathbf{y}_x(x,y)\big).0 = \overline{f}\big(x \dot{-} x,\mathbf{y}_x(x,y)\big).0 = \overline{f}\big(\mathbf{x}_x(x),\mathbf{y}_x(x,y)\big).0.\end{aligned}$$

If $i < x$ then first note that we have

$$\mathbf{x}_1\mathbf{x}_i(x) = \mathbf{x}_{i+1}(x) \tag{$\dagger_1$}$$
$$\mathbf{y}_1(\mathbf{x}_i(x),\mathbf{y}_i(x,y)) = \mathbf{y}_{i+1}(x,y). \tag{$\dagger_2$}$$

Indeed, we have

$$\begin{aligned}\mathbf{x}_1\mathbf{x}_i(x) &= x \dot{-} i \dot{-} 1 = x \dot{-} (i+1) = \mathbf{x}_{i+1}(x)\\ \mathbf{y}_1(\mathbf{x}_i(x),\mathbf{y}_i(x,y)) &= \sigma[\mathbf{x}_1\mathbf{x}_i(x),\mathbf{y}_0(\mathbf{x}_i(x),\mathbf{y}_i(x,y))] \overset{(\dagger_1)}{=}\\ &= \sigma[\mathbf{x}_{i+1}(x),\mathbf{y}_i(x,y)] = \mathbf{y}_{i+1}(x,y).\end{aligned}$$

We have $i < i+1 \leq x$ and thus

$$\mathbf{x}_i(x) \dot{-} 1 = x \dot{-} i \dot{-} 1 = x \dot{-} (i+1) < x \dot{-} i.$$

Therefore

$$\begin{aligned}\overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).1 &\overset{\text{IH}}{=} \overline{f}\big(\mathbf{x}_1\mathbf{x}_i(x),\mathbf{y}_1(\mathbf{x}_i(x),\mathbf{y}_i(x,y))\big).0 \overset{(\dagger_1),(\dagger_2)}{=}\\ &= \overline{f}\big(\mathbf{x}_{i+1}(x),\mathbf{y}_{i+1}(x,y)\big).0.\end{aligned}$$

Note that the induction hypothesis is applied with $\mathbf{y}_i(x,y)$ in place of y. This means that we have

$$\overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).1 = \overline{f}\big(\mathbf{x}_{i+1}(x),\mathbf{y}_{i+1}(x,y)\big).0\ . \tag{$\dagger_3$}$$

The induction step follows from

$$\begin{aligned}&\overline{f}(x,y).i = \Big\langle \tau\big[\mathbf{x}_i(x) \dot{-} 1, \overline{f}(x,y).(i+1)[0], \mathbf{y}_i(x,y)\big], \overline{f}(x,y).(i+1)\Big\rangle \overset{\text{IH}}{=}\\ &\Big\langle \tau\big[\mathbf{x}_i(x) \dot{-} 1, \overline{f}\big(\mathbf{x}_{i+1}(x),\mathbf{y}_{i+1}(x,y)\big).0, \mathbf{y}_i(x,y)\big], \overline{f}\big(\mathbf{x}_{i+1}(x),\mathbf{y}_{i+1}(x,y)\big).0\Big\rangle \overset{(\dagger_3)}{=}\\ &\Big\langle \tau\big[\mathbf{x}_i(x) \dot{-} 1, \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).1, \mathbf{y}_i(x,y)\big], \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).1\Big\rangle =\\ &\Big\langle \tau\big[\mathbf{x}_0\mathbf{x}_i(x) \dot{-} 1, \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).1, \mathbf{y}_0(\mathbf{x}_i(x),\mathbf{y}_i(x,y)), \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).1\big]\Big\rangle =\\ &\overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).0.\end{aligned}$$

□

1.7.10 Course of values function. The binary function $\overline{f}(x,y)$ returns the course of values sequence for $f(x,y)$. The function satisfies

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

$$\overline{f}(x+1,y) = \Big\langle \tau\big[x, \overline{f}(x, \sigma[x,y])[0], y\big], \overline{f}(x, \sigma[x,y]) \Big\rangle \tag{2}$$

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

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

Proof. (1): We have $0 \geq 0$ and thus

$$\overline{f}(0,y) = \overline{f}(0,y).0 = \langle \mathbf{y}_0(0, \rho[y]), 0 \rangle = \langle \rho[y], 0 \rangle.$$

(2): First note that we have

$$\mathbf{x}_1(x+1) = x + 1 \dot{-} 1 = x \tag{$\dagger_1$}$$

$$\mathbf{y}_1(x+1,y) = \sigma\big[\mathbf{x}_1(x+1), \mathbf{y}_0(x+1,y)\big] \overset{(\dagger_1)}{=} \sigma[x,y]. \tag{$\dagger_2$}$$

Clearly $0 < x+1$ and therefore

$$\begin{aligned}
&\overline{f}(x+1,y) = \overline{f}(x+1,y).0 = \\
&= \Big\langle \tau\big[\mathbf{x}_0(x+1) \dot{-} 1, \overline{f}(x+1,y).1[0], \mathbf{y}_0(x+1,y)\big], \overline{f}(x+1,y).1 \Big\rangle = \\
&= \Big\langle \tau\big[x, \overline{f}(x+1,y).1[0], y\big], \overline{f}(x+1,y).1 \Big\rangle \overset{1.7.9(1)}{=} \\
&= \Big\langle \tau\big[x, \overline{f}\big(\mathbf{x}_1(x+1), \mathbf{y}_1(x+1,y)\big)[0], y\big], \overline{f}\big(\mathbf{x}_1(x+1), \mathbf{y}_1(x+1,y)\big) \Big\rangle \overset{(\dagger_1),(\dagger_2)}{=} \\
&= \Big\langle \tau\big[x, \overline{f}(x, \sigma[x,y])[0], y\big], \overline{f}(x, \sigma[x,y]) \Big\rangle. \qquad \square
\end{aligned}$$

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

Proof. Let f be defined by recursion with parameter substitution as in Par. 1.7.5 from p.r. functions. Let further $\overline{f}$ be its course of values function as in Par. 1.7.10. We claim that we have

$$f(x,y) = \overline{f}(x,y)[0]. \tag{1}$$

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

The property is proved by induction on x as $\forall y(1)$. In the base case take any y and we obtain

$$f(0,y) = \rho[y] = \langle \rho[y], 0 \rangle[0] \overset{1.7.10(1)}{=} \overline{f}(0,y)[0].$$

In the induction step take any y and we have

$$f(x+1,y) = \tau\big[x, f(x,\sigma[x,y]), y\big] \overset{\text{IH}}{=} \tau\big[x, \overline{f}(x,\sigma[x,y])[0], y\big] =$$
$$= \Big\langle \tau\big[x, \overline{f}(x,\sigma[x,y])[0], y\big], \overline{f}\big(x,\sigma[x,y]\big)\Big\rangle[0] \overset{1.7.10(2)}{=} \overline{f}(x+1,y)[0]. \qquad \square$$

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

1.7.12 Introduction. In this subsection we will investigate the scheme of primitive recursion with substitution in one parameter ($n = 1$), where only two different recursive applications are allowed ($k = 2$). The fact that p.r. functions are closed under such recursion will be proved in Thm. 1.7.21.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following recursion with parameter substitution

$$f(0,y) = \rho[y] \tag{1}$$
$$f(x+1,y) = \tau\big[x, f\big(x,\sigma_1[x,y]\big), f\big(x,\sigma_2[x,y]\big), y\big] \tag{2}$$

from p.r. functions. We claim that f is also primitive recursive. We will prove this fact by reducing the scheme to backward recursion.

1.7.13 The outline of of the proof. We will 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 1.7.12(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

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

The function f can then be defined explicitly by

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

as primitive recursive.

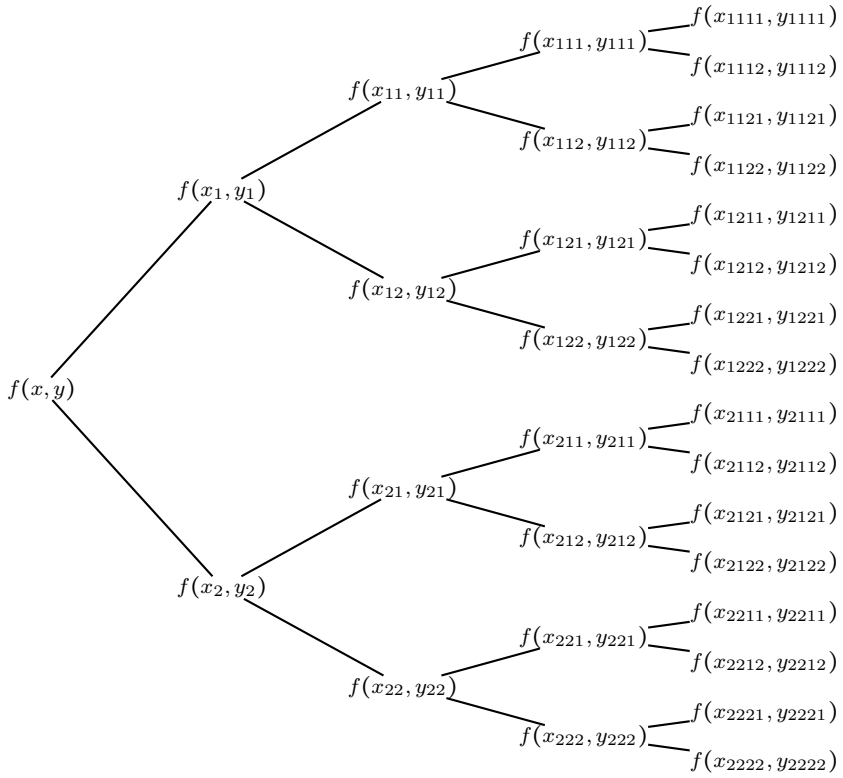

Fig. 1.4 Computation tree of depth 5

Figure 1.4 shows an example of computation tree $\overline{f}(x,y)$ for the application $f(x,y)$. Note that each node of the tree has the form $f(x_i,y_i)$ for some word i over two-symbol alphabet $\Sigma = \{1,2\}$. The dyadic word i represents the path from the root to that node in obvious manner. Note that we can easily recover the arguments of the application $f(x_i,y_i)$ from the arguments of the application $f(x,y)$ and the path i. Clearly, the recursive argument x_i is the difference between x and the and the length of the dyadic word i. Moreover

$$y_\varnothing = y \qquad y_{i1} = \sigma_1[x_i \dot{-} 1, y_i] \qquad y_{i2} = \sigma_2[x_i \dot{-} 1, y_i].$$

This gives us simple recurrences for computing the parameter y_i.

There is simple correspondence between dyadic words and the so-called dyadic representation of natural numbers. It is easy 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 the dyadic successors

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

This is called the *dyadic representation* of natural numbers. Dyadic representation allows simple coding of dyadic words into natural numbers. For instance, the code number of the dyadic word 221 is the number

$$0\mathbf{221} = 2 \times (2 \times (2 \times 0 + 2) + 2) + 1 = 2 \times 2^2 + 2 \times 2^1 + 1 \times 2^0 = 13.$$

Arithmetization is so straightforward that, from now on, we will usually identify dyadic words with their code numbers.

We intend to compute $\overline{f}(x, y)$ from bottom-up using backward recursion. This is done with the help of the course of values subtree function $\overline{f}(x, y).i$ which returns the subtree of the computation tree for $f(x, y)$ at position indexed by the dyadic path i. That is, we would like to have

$$\pi_1 \overline{f}(x, y).i = \overline{f}(x_i, y_i)$$

for every dyadic path i in the computation tree $\overline{f}(x, y)$. Hence

$$\overline{f}(x, y) = \pi_1 \overline{f}(x, y).0$$

and we can take the identity as an explicit definition of the course of values function. Note that 0 is the code number of the empty dyadic word $\varnothing$.

1.7.14 Dyadic case analysis. Note that we have

$$x = 0 \vee \exists y\, x = x\mathbf{1} \vee \exists y\, x = x\mathbf{2}. \tag{1}$$

This is called the principle of *dyadic case analysis on the number* x.

1.7.15 Dyadic size. The unary function $|x|_{\mathrm{d}}$ yields the number of dyadic successors in the dyadic representation of x. The dyadic size function satisfies

$$|0|_{\mathrm{d}} = 0 \tag{1}$$
$$|x\mathbf{1}|_{\mathrm{d}} = |x|_{\mathrm{d}} + 1 \tag{2}$$
$$|x\mathbf{2}|_{\mathrm{d}} = |x|_{\mathrm{d}} + 1 \tag{3}$$

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

$$|0|_{\mathrm{d}} = 0$$
$$|x + 1|_{\mathrm{d}} = |x \div 2|_{\mathrm{d}} + 1.$$

The following property will be needed later:

$$|x|_{\mathrm{d}} < n \leftrightarrow x + 1 < 2^n. \tag{4}$$

In the sequel we will tacitly use the properties (1)-(3) of dyadic size.

1.7.16 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

$$x \star 0 = x \tag{1}$$
$$x \star y\mathbf{1} = (x \star y)\mathbf{1} \tag{2}$$
$$x \star y\mathbf{2} = (x \star y)\mathbf{2} \tag{3}$$

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

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

We will need the following distributive property:

$$|x \star y|_{\mathrm{d}} = |x|_{\mathrm{d}} + |y|_{\mathrm{d}}\,. \tag{4}$$

In the sequel we will tacitly use the properties (1)-(3) of dyadic concatenation.

1.7.17 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

$$\mathbf{x}_0(x) = x \tag{1}$$
$$\mathbf{x}_{i\mathbf{1}}(x) = \mathbf{x}_i(x) \dot{-} 1 \tag{2}$$
$$\mathbf{x}_{i\mathbf{2}}(x) = \mathbf{x}_i(x) \dot{-} 1 \tag{3}$$

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

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

We will need the following *composition* property of the selector function:

$$\vdash_{\mathrm{PA}} \mathbf{x}_{i\star j}(x) = \mathbf{x}_j\mathbf{x}_i(x). \tag{4}$$

In the sequel we will tacitly use the properties (1)-(3).

1.7.18 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

$$\mathbf{y}_0(x, y) = y \tag{1}$$
$$\mathbf{y}_{i\mathbf{1}}(x, y) = \sigma_1[\mathbf{x}_i(x) \dot{-} 1, \mathbf{y}_i(x, y)] \tag{2}$$
$$\mathbf{y}_{i\mathbf{2}}(x, y) = \sigma_2[\mathbf{x}_i(x) \dot{-} 1, \mathbf{y}_i(x, y)] \tag{3}$$

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

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

The following is the *composition* property of the parameter selector function:

$$\mathbf{y}_{i\star j}(x,y) = \mathbf{y}_j\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big). \tag{4}$$

In the sequel we will use the properties (1)-(3) without explicit reference.

1.7.19 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:

$$|i|_{\mathrm{d}} = x \to \overline{f}(x,y).i = \big\langle \rho[\mathbf{y}_i(x,y)], 0, 0\big\rangle \tag{1}$$

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

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

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

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

$$|i \star j|_{\mathrm{d}} \leq x \to \overline{f}(x,y).(i \star j) = \overline{f}\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big).j. \tag{3}$$

Proof. (1),(2): Directly from definition by noting that

$$i < 2^x \dot{-} 1 \Leftrightarrow i + 1 < 2^x \overset{1.7.15(4)}{\Leftrightarrow} |i|_{\mathrm{d}} < x.$$

(3): By backward induction on the difference $x \dot{-} |j|_{\mathrm{d}}$. So take any i, j, x such that $|i \star j|_{\mathrm{d}} \leq x$ and consider two cases. If $|i \star j|_{\mathrm{d}} = x$ then $|i|_{\mathrm{d}} + |j|_{\mathrm{d}} = x$ by 1.7.16(4) and therefore $|j|_{\mathrm{d}} = x \dot{-} |i|_{\mathrm{d}} = \mathbf{x}_i(x)$. We obtain

$$\begin{aligned}\overline{f}(x,y).(i \star j) &\overset{(1)}{=} \big\langle \rho[\mathbf{y}_{i\star j}(x,y)], 0, 0\big\rangle \overset{1.7.18(4)}{=} \big\langle \rho\big[\mathbf{y}_j\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big)\big], 0, 0\big\rangle \overset{(1)}{=}\\ &= \overline{f}\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big).j\end{aligned}$$

So suppose that $|i \star j|_{\mathrm{d}} < x$. We then have $|i|_{\mathrm{d}} + |j|_{\mathrm{d}} < x$ by 1.7.16(4) and therefore $|j|_{\mathrm{d}} < x \dot{-} |i|_{\mathrm{d}} = \mathbf{x}_i(x)$ and

$$|i \star j\mathbf{1}|_{\mathrm{d}} \overset{1.7.16(4)}{=} |i|_{\mathrm{d}} + |j\mathbf{1}|_{\mathrm{d}} = |i|_{\mathrm{d}} + |j|_{\mathrm{d}} + 1 \leq x.$$

From $|j|_{\mathrm{d}} < x$ we obtain $|j|_{\mathrm{d}} < |j|_{\mathrm{d}} + 1 = |j\mathbf{1}|_{\mathrm{d}} \leq x$ and thus $x \dot{-} |j\mathbf{1}|_{\mathrm{d}} < x \dot{-} |j|_{\mathrm{d}}$. Therefore

$$\begin{aligned}
&\overline{f}(x,y).(i \star j) \overset{(2)}{=} \\
&= \Big\langle \tau\big[\mathbf{x}_{i\star j}(x) \dot{-} 1, \pi_1\big(\overline{f}(x,y).(i \star j)\mathbf{1}\big), \pi_1\big(\overline{f}(x,y).(i \star j)\mathbf{2}\big), \mathbf{y}_{i\star j}(x,y)\big], \\
&\quad \overline{f}(x,y).(i \star j)\mathbf{1}, \overline{f}(x,y).(i \star j)\mathbf{2}\Big\rangle \overset{1.7.17(4),1.7.18(4)}{=} \\
&= \Big\langle \tau\big[\mathbf{x}_j\mathbf{x}_i(x) \dot{-} 1, \pi_1\big(\overline{f}(x,y).(i \star j\mathbf{1})\big), \pi_1\big(\overline{f}(x,y).(i \star j\mathbf{2})\big), \\
&\quad \mathbf{y}_j\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big)\big], \overline{f}(x,y).(i \star j\mathbf{1}), \overline{f}(x,y).(i \star j\mathbf{2})\Big\rangle \overset{2\times\mathrm{IH}}{=} \\
&= \Big\langle \tau\big[\mathbf{x}_j\mathbf{x}_i(x) \dot{-} 1, \pi_1\big(\overline{f}(\mathbf{x}_i(x), \mathbf{y}_i(x,y)).j\mathbf{1}\big), \pi_1\big(\overline{f}(\mathbf{x}_i(x), \mathbf{y}_i(x,y)).j\mathbf{2}\big), \\
&\quad \mathbf{y}_j\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big)\big], \overline{f}(\mathbf{x}_i(x), \mathbf{y}_i(x,y)).j\mathbf{1}, \overline{f}(\mathbf{x}_i(x), \mathbf{y}_i(x,y)).j\mathbf{2}\Big\rangle \overset{(2)}{=} \\
&= \overline{f}\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big).j. \qquad \square
\end{aligned}$$

1.7.20 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

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

Proof. (1): We have $|0|_{\mathrm{d}} = 0$ and thus

$$\overline{f}(0,y) = \overline{f}(0,y).0 \overset{1.7.19(1)}{=} \big\langle \rho[\mathbf{y}_0(0,y)], 0, 0\big\rangle = \big\langle \rho[y], 0, 0\big\rangle.$$

(2): First note that the following holds

$$T \vdash |i|_{\mathrm{d}} \leq x \to \overline{f}(x,y).i = \overline{f}\big(\mathbf{x}_i(x), \mathbf{y}_i(x,y)\big). \tag{$\dagger_1$}$$

Indeed, if $|i|_{\mathrm{d}} \leq x$ then $|i \star 0|_{\mathrm{d}} = |i|_{\mathrm{d}} \leq x$ and therefore

$$\overline{f}(x,y).i = \overline{f}(x,y).(i \star 0) \overset{1.7.19(3)}{=} \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).0 = \overline{f}\big(\mathbf{x}_i(x),\mathbf{y}_i(x,y)\big).$$

Further note, for instance, that

$$\mathbf{x}_{0_1}(x+1) = \mathbf{x}_0(x+1) \dot{-} 1 = x+1 \dot{-} 1 = x$$
$$\mathbf{y}_{0_1}(x+1,y) = \sigma_1\big[\mathbf{x}_0(x+1) \dot{-} 1, \mathbf{y}_0(x+1,y)\big] = \sigma_1\big[x+1 \dot{-} 1, y\big] = \sigma_1[x,y],$$

and thus we have

$$\mathbf{x}_{0_1}(x+1) = x \wedge \mathbf{x}_{0_2}(x+1) = x \tag{$\dagger_2$}$$
$$\mathbf{y}_{0_1}(x+1,y) = \sigma_1[x,y] \wedge \mathbf{y}_{0_2}(x+1,y) = \sigma_2[x,y]. \tag{$\dagger_3$}$$

We have $|0|_{\mathrm{d}} = 0 < x+1$ and therefore

$$\begin{aligned}
&\overline{f}(x+1,y) = \overline{f}(x+1,y).0 \overset{1.7.19(2)}{=} \\
&= \Big\langle \tau\big[\mathbf{x}_0(x+1) \dot{-} 1, \pi_1\big(\overline{f}(x+1,y).0\mathbf{1}\big), \pi_1\big(\overline{f}(x+1,y).0\mathbf{2}\big), \mathbf{y}_0(x+1,y)\big], \\
&\quad \overline{f}(x+1,y).0\mathbf{1}, \overline{f}(x+1,y).0\mathbf{2}\Big\rangle = \\
&= \Big\langle \tau\big[x, \pi_1\big(\overline{f}(x+1,y).0\mathbf{1}\big), \pi_1\big(\overline{f}(x+1,y).0\mathbf{2}\big), y\big], \\
&\quad \overline{f}(x+1,y).0\mathbf{1}, \overline{f}(x+1,y).0\mathbf{2}\Big\rangle \overset{(\dagger_1)}{=} \\
&= \Big\langle \tau\big[x, \pi_1\overline{f}\big(\mathbf{x}_{0_1}(x+1),\mathbf{y}_{0_1}(x+1,y)\big), \pi_1\overline{f}\big(\mathbf{x}_{0_2}(x+1),\mathbf{y}_{0_2}(x+1,y)\big), y\big], \\
&\quad \overline{f}\big(\mathbf{x}_{0_1}(x+1),\mathbf{y}_{0_1}(x+1,y)\big), \overline{f}\big(\mathbf{x}_{0_2}(x+1),\mathbf{y}_{0_2}(x+1,y)\big)\Big\rangle \overset{(\dagger_2),(\dagger_3)}{=} \\
&= \Big\langle \tau\big[x, \pi_1\overline{f}\big(x,\sigma_1[x,y]\big), \pi_1\overline{f}\big(x,\sigma_2[x,y]\big), y\big], \overline{f}\big(x,\sigma_1[x,y]\big), \overline{f}\big(x,\sigma_2[x,y]\big)\Big\rangle.
\end{aligned}$$

This proves the property (2). □

1.7.21 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 recursion with parameter substitution as in Par. 1.7.12 from p.r. functions. Let further $\overline{f}$ be its course of values function as in Par. 1.7.20. We claim that we have

$$f(x,y) = \pi_1\overline{f}(x,y), \tag{1}$$

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

This is proved by induction on x as $\forall y(1)$. The base case follows from

$$f(0,y) = \rho[y] = \pi_1\langle g(y),0,0\rangle \overset{1.7.20(1)}{=} \pi_1\overline{f}(0,y).$$

In the induction step take any y and we obtain

$$f(x+1,y) = \tau\big[x, f\big(x,\sigma_1[x,y]\big), f\big(x,\sigma_2[x,y]\big), y\big] \overset{2\times\mathrm{IH}}{=}$$
$$= \tau\big[x, \pi_1\overline{f}\big(x,\sigma_1[x,y]\big), \pi_1\overline{f}\big(x,\sigma_2[x,y]\big), y\big] \overset{1.7.20(2)}{=} \pi_1\overline{f}(x+1,y). \qquad \square$$

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

1.7.22 Introduction. In this subsection we will show that p.r. functions are closed under the scheme of primitive recursion with substitution in one parameter. This will be proved in Thm. 1.7.24 by reducing the number of recursive applications. This leads eventually to primitive recursion with substitution in one parameter and two recursive applications.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following recursion with parameter substitution

$$f(0,y) = \rho[y] \tag{1}$$
$$f(x+1,y) = \tau\big[x, f\big(x,\sigma_1[x,y]\big), \ldots, f\big(x,\sigma_{k+1}[x,y]\big), y\big] \tag{2}$$

from p.r. functions. We claim that f is also primitive recursive.

1.7.23 Reduction of the number of recursive applications. We will reduce the above definition for $k \geq 2$ to a new one, where only k recursive applications are allowed. This new definition is for a binary function $\hat{f}(u,v)$ such that $\hat{f}(2x,y) = f(x,y)$ and it is of the form

$$\hat{f}(0,v) = \rho[v]$$
$$\hat{f}(u+1,v) = \hat{\tau}\big[u, \hat{f}\big(u,\hat{\sigma}_1[u,v]\big), \ldots, \hat{f}\big(u,\hat{\sigma}_k[u,v]\big), v\big]$$

for suitable terms $\hat{\tau}[u,\vec{w},v], \hat{\sigma}_1[u,v], \ldots, \hat{\sigma}_k[u,v]$. Note that this is primitive recursion on u with substitution in the parameter v with k recursive applications. We will take then the identity $f(x,y) = \hat{f}(2x,y)$ as an alternative, explicit definition of f.

The idea behind reduction of recursive applications is as follows. We would like to have $\hat{f}(2x,y) = f(x,y)$ and so it must be

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

For the values of the form $\hat{f}(2x+1,v)$ we require

$$\hat{f}\big(2x+1,\langle 1,y\rangle\big) = \Big\langle f\big(x,\sigma_1[x,y]\big), \ldots, f\big(x,\sigma_k[x,y]\big)\Big\rangle$$
$$\hat{f}\big(2x+1,\langle 2,y\rangle\big) = f\big(x,\sigma_{k+1}[x,y]\big).$$

Note that the application $\hat{f}\big(2x+1,\langle 1,y\rangle\big)$ returns a number which codes k values $f\big(x,\sigma_1[x,y]\big),\ldots,f\big(x,\sigma_k[x,y]\big)$ of the function f.

These informal arguments can be rewritten without mentioning the function f as follows:

$$\hat{f}\big(2x+1,\langle 1,y\rangle\big) = \Big\langle \hat{f}\big(2x,\sigma_1[x,y]\big),\ldots,\hat{f}\big(2x,\sigma_k[x,y]\big)\Big\rangle \tag{2}$$

$$\hat{f}\big(2x+1,\langle 2,y\rangle\big) = \hat{f}\big(2x,\sigma_{k+1}[x,y]\big). \tag{3}$$

$$\hat{f}\big(2x+2,y\big) = \tau\Big[x,\big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_1^k,\ldots,\big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_k^k, \hat{f}\big(2x+1,\langle 2,y\rangle\big),y\Big] \tag{4}$$

This means that the terms $\hat{\tau},\hat{\sigma}_1,\ldots,\hat{\sigma}_k$. satisfy the properties

$$\hat{\tau}[2x+1,z,z_{k+1},\ldots,y] = \tau\Big[x,[z]_1^k,\ldots,[z]_k^k,z_{k+1},y\Big] \tag{5}$$

$$\hat{\tau}\big[2x,z_1,\ldots,z_k,\langle 1,y\rangle\big] = \langle z_1,\ldots,z_k\rangle \tag{6}$$

$$\hat{\tau}\big[2x,z_{k+1},\ldots,\langle 2,y\rangle\big] = z_{k+1} \tag{7}$$

and

$$\hat{\sigma}_1[2x+1,y] = \langle 1,y\rangle \tag{8}$$

$$\hat{\sigma}_2[2x+1,y] = \langle 2,y\rangle \tag{9}$$

$$\bigwedge_{i=1}^{k}\hat{\sigma}_i\big[2x,\langle 1,y\rangle\big] = \sigma_i[x,y] \tag{10}$$

$$\hat{\sigma}_1\big[2x,\langle 2,y\rangle\big] = \sigma_{k+1}[x,y]. \tag{11}$$

For that it is sufficient to set

$$\hat{\tau}[u,w_1,\ldots,w_k,v] \equiv D\Big(u \bmod 2,\tau\big[u\div 2,[w_1]_1^k,\ldots,[w_1]_k^k,w_2,v\big], D\big(\pi_1(v) =_* 1,\langle w_1,\ldots,w_k\rangle,w_1\big)\Big)$$

and

$$\hat{\sigma}_1[u,v] \equiv D\Big(u \bmod 2,\langle 1,v\rangle, D\Big(\pi_1(v) =_* 1,\sigma_1\big[u\div 2,\pi_2(v)\big],\sigma_{k+1}\big[u\div 2,\pi_2(v)\big]\Big)\Big)$$

$$\hat{\sigma}_2[u,v] \equiv D\Big(u \bmod 2,\langle 2,v\rangle,\sigma_2\big[u\div 2,\pi_2(v)\big]\Big)$$

$$\hat{\sigma}_i[u,v] \equiv \sigma_i\big[u\div 2,\pi_2(v)\big] \quad \text{for } i=3,\ldots,k.$$

Proof. (5)-(11): Directly from definition. (2): It follows from

$$\begin{aligned}
&\hat{f}\big(2x+1,\langle 1,y\rangle\big) = \\
&= \hat{\tau}\Big[2x, \hat{f}\Big(2x, \hat{\sigma}_1\big[2x,\langle 1,y\rangle\big]\Big), \ldots, \hat{f}\Big(2x, \hat{\sigma}_k\big[2x,\langle 1,y\rangle\big]\Big), \langle 1,y\rangle\Big] \overset{(10)}{=} \\
&= \hat{\tau}\big[2x, \hat{f}\big(2x,\sigma_1[x,y]\big), \ldots, \hat{f}\big(2x,\sigma_k[x,y]\big), \langle 1,y\rangle\big] \overset{(6)}{=} \\
&= \Big\langle \hat{f}\big(2x,\sigma_1[x,y]\big), \ldots, \hat{f}\big(2x,\sigma_k[x,y]\big)\Big\rangle.
\end{aligned}$$

(3): It follows from

$$\begin{aligned}
\hat{f}\big(2x+1,\langle 2,y\rangle\big) &= \hat{\tau}\Big[2x, \hat{f}\Big(2x, \hat{\sigma}_1\big[2x,\langle 2,y\rangle\big]\Big), \ldots, \langle 2,y\rangle\Big] \overset{(11)}{=} \\
&= \hat{\tau}\big[2x, \hat{f}\big(2x,\sigma_{k+1}[x,y]\big), \ldots, \langle 2,y\rangle\big] \overset{(7)}{=} \hat{f}\big(2x,\sigma_{k+1}[x,y]\big).
\end{aligned}$$

(4): It follows from

$$\begin{aligned}
&\hat{f}(2x+2,y) = \hat{f}(2x+1+1,y) = \\
&= \hat{\tau}\Big[2x+1, \hat{f}\big(2x+1,\hat{\sigma}_1[2x+1,y]\big), \hat{f}\big(2x+1,\hat{\sigma}_2[2x+1,y]\big), \ldots, y\Big] \overset{(8),(9)}{=} \\
&= \hat{\tau}\Big[2x+1, \hat{f}\big(2x+1,\langle 1,y\rangle\big), \hat{f}\big(2x+1,\langle 2,y\rangle\big), \ldots, y\Big] \overset{(5)}{=} \\
&= \tau\Big[x, \big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_1^k, \ldots, \big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_k^k, \hat{f}\big(2x+1,\langle 2,y\rangle\big), y\Big].
\end{aligned}$$

We are now in position to prove (1):

$$\begin{aligned}
&\hat{f}\big(2(x+1),y\big) = \hat{f}(2x+2,y) \overset{(4)}{=} \\
&= \tau\Big[x, \big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_1^k, \ldots, \big[\hat{f}\big(2x+1,\langle 1,y\rangle\big)\big]_k^k, \hat{f}\big(2x+1,\langle 2,y\rangle\big), y\Big] \overset{(2),(3)}{=} \\
&= \tau\big[x, \hat{f}\big(2x,\sigma_1[x,y]\big), \ldots, \hat{f}\big(2x,\sigma_k[x,y]\big), \hat{f}\big(2x,\sigma_{k+1}[x,y]\big), y\big]. \qquad \square
\end{aligned}$$

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

Proof. The claim is proved by (meta-)induction on the number k of recursive applications in the defining axiom 1.7.22(2). The case $k = 0$ is in fact explicit definition with monadic discrimination on the first argument and it follows from Thm. 1.2.5. The cases $k = 1$ or $k = 2$ follow from Thm. 1.7.21. So suppose that the claim holds for the case $k \geq 2$. We will prove that the claim holds also for the case $k + 1$.

Let f be defined by primitive recursion with parameter substitution as in Par. 1.7.22 from p.r. functions. Let further $\hat{f}$ be the function from Par. 1.7.23. We claim that we have

$$f(x,y) = \hat{f}(2x,y), \tag{1}$$

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

The property is proved by induction on x as $\forall y(1)$. In the base case take any y and we have

$$f(0,y) = \rho[y] = \hat{f}(0,y) = \hat{f}(2\times 0, y).$$

In the induction step take any y and we obtain

$$\begin{aligned} f(x+1,y) &= \tau\big[x, f\big(x,\sigma_1[x,y]\big),\dots,f\big(x,\sigma_{k+1}[x,y]\big),y\big] \overset{(k+1)\times\mathrm{IH}}{=} \\ &= \tau\big[x, \hat{f}\big(2x,\sigma_1[x,y]\big),\dots,\hat{f}\big(2x,\sigma_{k+1}[x,y]\big),y\big] \overset{1.7.23(1)}{=} \\ &= \hat{f}\big(2(x+1),y\big). \end{aligned}$$

□

Primitive Recursion with Parameter Substitution

1.7.25 Introduction. In this subsection we will show that p.r. functions are closed under the scheme of primitive recursion with substitution in arbitrary number of parameters. This will be proved in Thm. 1.7.27 by reducing it to primitive recursion with substitution in one parameter.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following recursion with parameter substitution

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

$$f(x+1,\vec{y}) = \tau\left[x, f\big(x,\vec{\sigma}_1[x,\vec{y}]\big),\dots,f\big(x,\vec{\sigma}_k[x,\vec{y}]\big),\vec{y}\right] \tag{2}$$

from p.r. functions. We claim that f is also primitive recursive.

Below we will consider the case when the definition has at least two parameters, i.e. $n \geq 2$. The case $n = 0$ is in fact parameterless primitive recursion for which the claim has been already proved in Thm. 1.2.5. The case with one parameter ($n = 1$) follows from Thm. 1.7.24.

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

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

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

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

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

Proof. Let f be defined by primitive recursion with parameter substitution as in Par. 1.7.25 from p.r. functions, where the number of parameters is at least two ($n \geq 2$).[1] Let further $\langle f\rangle$ be the contraction function of f from Par. 1.7.26. We claim that

$$f(x,\vec y) = \langle f\rangle\big(x,\langle\vec y\rangle\big). \tag{1}$$

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

The property is proved by induction on x as $\forall\vec y(1)$. In the base case take any $\vec y$ and we have

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

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

1.7.28 Introduction. In this subsection we will show that p.r. functions are closed under the general scheme of course of values recursion with parameter substitution. This will be proved in Thm. 1.7.31 by it to primitive recursion with parameter substitution.

We will fix the notation used in this subsection as follows. Let f be the function defined by the following course of values recursion with parameter

[1] The cases $n = 0$ or $n = 1$ follow from Thm. 1.2.5 or Thm. 1.7.22, respectively.

substitution

$$f(0,\vec{y}) = \rho[\vec{y}] \tag{1}$$
$$f(x+1,\vec{y}) = \tau\big[x, f\big(\xi_1[x,\vec{y}],\vec{\sigma}_1[x,\vec{y}]\big),\ldots,f\big(\xi_k[x,\vec{y}],\vec{\sigma}_k[x,\vec{y}]\big),\vec{y}\big] \tag{2}$$

from p.r. functions, where

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

We claim that f is also primitive recursive.

1.7.29 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 introduce f as primitive recursive by the following explicit definition $f(x,\vec{y}) = f^+(x+1,x,\vec{y})$.

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

$$f^+(0,x,\vec{y}) = 0 \tag{1}$$
$$f^+(z+1,0,\vec{y}) = \rho[\vec{y}] \tag{2}$$
$$f^+(z+1,x+1,\vec{y}) = \tau\Big[x, f^+\big(z,\xi_1[x,\vec{y}],\vec{\sigma}_1[x,\vec{y}]\big),\ldots, f^+\big(z,\xi_k[x,\vec{y}],\vec{\sigma}_k[x,\vec{y}]\big),\vec{y}\Big], \tag{3}$$

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

$$f^+(0,x,\vec{y}) = 0$$
$$f^+(z+1,x,\vec{y}) = D\Big(x, \tau\Big[x\dot{-}1, f^+\big(z,\xi_1[x\dot{-}1,\vec{y}],\vec{\sigma}_1[x\dot{-}1,\vec{y}]\big),\ldots, f^+\big(z,\xi_k[x\dot{-}1,\vec{y}],\vec{\sigma}_k[x\dot{-}1,\vec{y}]\big),\vec{y}\Big],\rho[\vec{y}]\Big).$$

1.7.30 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 $\forall x\forall\vec{y}\forall z_2(1)$. In the base case there is nothing to prove. In the induction step take any numbers $x,\vec{y},z_2$

such that $x < z_1 + 1$ and $x < z_2$. Then $z_2 = z_2' + 1$ for some z_2'. We consider two cases. If $x = 0$ then we have

$$f^+(z_1 + 1, 0, \vec{y}) \overset{1.7.29(2)}{=} \rho[\vec{y}] \overset{1.7.29(2)}{=} f^+(z_2' + 1, 0, \vec{y}).$$

If $x = x' + 1$ for some x' then $\xi_i[x', \vec{y}] \leq x' < z_1$ and $\xi_i[x', \vec{y}] \leq x' < z_2'$ for every $i = 1, \ldots, k$ by 1.7.28(3), We obtain

$$\begin{aligned}
&f^+(z_1 + 1, x' + 1, \vec{y}) \overset{1.7.29(3)}{=} \\
&\quad = \tau\Big[x', f^+\big(z_1, \xi_1[x', \vec{y}], \vec{\sigma}_1[x', \vec{y}]\big), \ldots, f^+\big(z_1, \xi_k[x', \vec{y}], \vec{\sigma}_k[x', \vec{y}]\big), \vec{y}\Big] \overset{k\times\mathrm{IH}}{=} \\
&\quad = \tau\Big[x', f^+\big(z_2', \xi_1[x', \vec{y}], \vec{\sigma}_1[x', \vec{y}]\big), \ldots, f^+\big(z_2', \xi_k[x', \vec{y}], \vec{\sigma}_k[x', \vec{y}]\big), \vec{y}\Big] \overset{1.7.29(3)}{=} \\
&\quad = f^+(z_2' + 1, x' + 1, \vec{y}).
\end{aligned}$$

□

1.7.31 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 as in Par. 1.7.28 from p.r. functions. Let further f^+ be the approximation function of f from Par. 1.7.29. We claim that

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

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

The property is proved by complete induction on x as $\forall \vec{y}(1)$. So take any $\vec{y}$ and consider two cases. If $x = 0$ then we have

$$f(0, \vec{y}) = \rho[\vec{y}] \overset{1.7.29(2)}{=} f^+(0 + 1, 0, \vec{y}).$$

If $x = x' + 1$ for some x' then $\xi_i[x', \vec{y}] < x' + 1$ for every $i = 1, \ldots, k$ by 1.7.28(3), We then obtain

$$\begin{aligned}
f(x' + 1, \vec{y}) &= \tau\Big[x', f\big(\xi_1[x', \vec{y}], \vec{\sigma}_1[x', \vec{y}]\big), \ldots, f\big(\xi_k[x', \vec{y}], \vec{\sigma}_k[x', \vec{y}]\big), \vec{y}\Big] \overset{k\times\mathrm{IH}}{=} \\
&= \tau\Big[x', f^+\big(\xi_1[x', \vec{y}] + 1, \xi_1[x', \vec{y}], \vec{\sigma}_1[x', \vec{y}]\big), \ldots, \\
&\qquad f^+\big(\xi_k[x', \vec{y}] + 1, \xi_k[x', \vec{y}], \vec{\sigma}_k[x', \vec{y}]\big), \vec{y}\Big] \overset{1.7.30(1)}{=} \\
&= \tau\Big[x', f^+\big(x' + 1, \xi_1[x', \vec{y}], \vec{\sigma}_1[x', \vec{y}]\big), \ldots, \\
&\qquad f^+\big(x' + 1, \xi_k[x', \vec{y}], \vec{\sigma}_k[x', \vec{y}]\big), \vec{y}\Big] \overset{1.7.29(3)}{=} \\
&= f^+(x' + 1 + 1, x' + 1, \vec{y}).
\end{aligned}$$

□