1.7 Nested Simple Recursion

1.7.1 Introduction. In this section we will investigate recursive definitions for which parameters may be arbitrarily substituted for, even with nested recursive applications. For instance:

$$f(0,y) = g(y)$$
$$f(x+1,y) = h\Big(x, f\big(x, \sigma[x,y,f(x,y)]\big), y\Big).$$

The function f is an example of a 1-*recursive* function in the hierarchy of *multiply recursive* functions studied. Péter has proved in [1] that primitive recursive functions are closed under 1-recursion. The schema of 1-recursion is usually called nested simple recursion and we will also adopt this convention.

1.7.2 Notation. We will use the *special lambda notation* $\tau[\dot\lambda\vec y.\rho[\vec y];\vec x]$ where $\vec y$ are n variables for the term obtained from τ by the replacement of all applications $f(\vec\tau)$ by terms $\rho[\vec\tau]$. Note that we have $\tau[g;\vec x] \equiv \tau[\dot\lambda\vec y.g(\vec y);\vec x]$.

1.7.3 Nested simple recursion. Let $\rho[\vec y]$ and $\tau[f;x,\vec y]$ are terms in which no other variables than the indicated ones are free. Suppose that ρ does not apply f. Consider the $(n{+}1)$-ary function f defined by

$$f(0,\vec y) = \rho[\vec y] \tag{1}$$
$$f(x+1,\vec y) = \tau[\dot\lambda x_1\vec y.f(x,\vec y);x,\vec y]. \tag{2}$$

We say that f is defined by *nested simple recursion.*

We will show at the end of this section in Thm. 1.7.15 that p.r. functions are closed under nested simple recursion. The proof proceeds in stages. First, we prove the claim for the schema with with two recursive applications ($k = 2$) and one parameter ($n = 1$). This is proved in Thm. 1.7.11 by reducing the schema to course of values recursion with parameter substitution. Next, we extend this result to the schema with arbitrary number of recursive applications (Thm. 1.7.13). Finally, we prove the claim for the schema with arbitrary number of parameters (Thm. 1.7.15).

We may assume that the function f is applied in τ at least once because otherwise there would be nothing to prove. In order to simplify our discussion, in particular for the cases when $k \geq 2$, we transform the equation (2) into equivalent one by 'unnesting' all recursive applications of f in the term τ:

$$\bigwedge_{i=1}^{k} f(x,\vec\sigma_i[x,\vec y,\vec z_{i-1}]) = z_i \to f(x+1,\vec y) = \theta[x,\vec z,\vec y]. \tag{3}$$

Here $\vec z_i$ abbreviates $z_1,\dots,z_i$ and the terms $\sigma_1,\dots,\sigma_k,\theta$ contain at most the indicated variables and do not apply f.

Nested Simple Recursion: Case $k = 2$ and $n = 1$

1.7.4 Introduction. In this subsection we will investigate the schema of nested simple recursion with two different recursive applications ($k = 2$) and one parameter ($n = 1$):

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

The closure of p.r. functions under the schema will be shown in Thm. 1.7.11. Below we will assume that $\rho[y], \theta\big[x,z_1,z_2,y\big], \sigma_1[x,y], \sigma_2[x,y,z_1]$ are all primitive recursive. We want to show that f is primitive recursive as well.

1.7.5 The outline of the proof. We will introduce the function f as primitive recursive by arithmetization of its computation trees in which we use as computational rules the defining axioms 1.7.4(1)(2). The evaluation of the application $f(x,y)$ can be visualized as a full binary tree of depth $x+1$ 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 subtree, 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 $f(x,y)$:

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

We will show that the course of values function is primitive recursive. Hence it suffices to define f by explicit definition $f(x,y) = \pi_1\overline{f}(x,y)$ as a p.r. function.

Note that the natural evaluation strategy for calculating $f(x,y)$ corresponds to postorder traversal of the computation tree $\overline{f}(x,y)$ for $f(x,y)$. Consider, for instance, the computation tree from Fig. 1.3. The sequence

$$f(x_0,y_0), f(x_1,y_1), f(x_2,y_2), \ldots, f(x_i,y_i), \ldots \tag{1}$$

of its labels consists of all applications which are needed to compute its root value. Note that the parameter y_i of each application $f(x_i,y_i)$ depends only on those values $f(x_j,y_j)$ which are directly before it ($j < i$). This means that the order in which the sequence (1) is sorted corresponds to postorder traversl of the computation tree. We will extend this indexing schema also for binary trees (already shown in Fig. 1.3).

Let us now consider a finite sequence of full binary trees of depth $x+1$

$$t_0, t_1, t_2, \ldots, t_i, \ldots, t_{2^{x+1} \dot{-} 1}$$

where each tree t_{i+1} satisfies the following condition: *the subtree of* t_{i+1} *at position* i *is a computation tree for* $f(x_i, y_i)$. We will call such trees *partial* computation trees for $f(x,y)$. Note that the last tree $t_{2^{x+1} \dot{-} 1}$ is in fact a (full) computation tree for $f(x,y)$.

This suggests the following method for building the computation tree for $f(x,y)$. We start by creating a 'dummy' full binary tree t_0 of depth $x+1$. Suppose now that after $i < 2^{x+1}$-steps we have a partial computation tree t_i for $f(x,y)$, The tree is updated at position i by the value $f(x_i, y_i)$ whereby we obtain a new partial computation tree t_{i+1} for $f(x,y)$. After 2^{x+1} steps we obtain a full computation tree for $f(x,y)$.

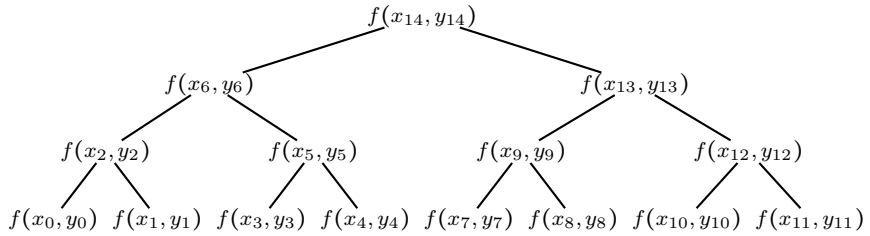

Fig. 1.3 Postorder traversal of a computation tree of depth 4.

1.7.6 Full binary trees. The function $\mathit{Full}(n)$ creates a full binary tree of the depth n. The function is defined by primitive recursion as a p.r. function:

$$\begin{aligned} \mathit{Full}(0) &= 0 \\ \mathit{Full}(n+1) &= \langle 0, \mathit{Full}(n), \mathit{Full}(n) \rangle. \end{aligned}$$

1.7.7 Local node condition. The application $V(x,y,l,r)$ determines the correct value $f(x,y)$ from the subtrees l and r of a partial computation tree $\langle z,l,r \rangle$ for $f(x,y)$. The function is primitive recursive by the following explicit definition (with monadic discrimination):

$$\begin{aligned} V(0,y,l,r) &= \rho[y] \\ V(x+1,y,l,r) &= \theta[x, \pi_1(l), \pi_1(r), y]. \end{aligned}$$

1.7.8 Local update. The 4-ary function $U(t,i,x,y)$ updates the partial computation tree t for $f(x,y)$ at position i by the correct value $f(x_i, y_i)$. The function has the following basic properties

$$i < 2^x \dot{-} 1 \to U\big(\langle z,l,r\rangle, i, x, y\big) = \Big\langle z, U\big(l, i, x \dot{-} 1, \sigma_1[x \dot{-} 1, y]\big), r\Big\rangle \tag{1}$$

$$\begin{aligned} j < 2^x \dot{-} 1 \to U\big(\langle z,l,r\rangle, 2^x \dot{-} 1 + j, x, y\big) = \\ \Big\langle z, l, U\big(r, j, x \dot{-} 1, \sigma_2[x \dot{-} 1, y, \pi_1(l)]\big)\Big\rangle \end{aligned} \tag{2}$$

$$U\big(\langle z,l,r\rangle, 2^{x+1} \dot{-} 2, x, y\big) = \big\langle V(x,y,l,r), l, r\big\rangle. \tag{3}$$

Note that both 'recursive' applications of the function U on the right-hand side of the conditional equations (1) and (2) are applied to lesser arguments $l < \langle z,l,r\rangle$ and $r < \langle z,l,r\rangle$ than the one on the left. We will use this observation to find a course of values recursive definition of U as follows. The transformation of the specification properties into course of values derivation of U is based on the following simple properties of the projection functions:

$$\begin{gathered} \exists z \exists l \exists r\, t = \langle z,l,r\rangle \leftrightarrow \pi_2(t) \neq 0 \\ t = \langle z,l,r\rangle \to z = \pi_1(t) \wedge l = \pi_1\pi_2(t) \wedge r = \pi_2^2(t) \\ \pi_1(t+1) < t+1 \wedge \pi_1\pi_2(t+1) < t+1 \wedge \pi_2^2(t+1) < t+1. \end{gathered}$$

Now let ξ be the term

$$\begin{aligned} \xi[t,l_1,r_1,i,x,y] \equiv\ & D\Big(\pi_2(t), \\ & D\Big(i{+}1 <_* 2^x, \\ & \quad \big\langle \pi_1(t), l_1, \pi_2^2(t)\big\rangle, \\ & \quad D\Big(i{+}2 <_* 2^{x+1}, \\ & \quad\quad \big\langle \pi_1(t), \pi_1\pi_2(t), r_1\big\rangle, \\ & \quad\quad \Big\langle V\big(x,y,\pi_1\pi_2(t),\pi_2^2(t)\big), \pi_1\pi_2(t), \pi_2^2(t)\Big\rangle\Big)\Big), 0\Big). \end{aligned}$$

The function $U(t,i,x,y)$ is defined by course of values recursion on t with substitution in the parameters as a p.r. function by

$$\begin{aligned} U(0,i,x,y) &= 0 \\ U(t+1,i,x,y) &= \xi\Big[t+1, U\big(\pi_1\pi_2(t+1), i, x \dot{-} 1, \sigma_1[x \dot{-} 1, y]\big), \\ &\qquad U\big(\pi_2^2(t+1), i \dot{-} (2^x \dot{-} 1), x \dot{-} 1, \sigma_2[x \dot{-} 1, y, \pi_1^2\pi_2(t+1)]\big), \\ &\qquad i, x, y\Big]. \end{aligned}$$

1.7.9 Global update. The 4-ary function $M_i(x,y,t)$ updates the partial computation tree t for $f(x,y)$ at each position $j < i$ by the correct value

$f(x_i, y_i)$. The function is defined by primitive recursion on i as a p.r. function:

$$\begin{aligned} M_0(x,y,t) &= t \\ M_{i+1}(x,y,t) &= U(M_i(x,y,t), i, x, y). \end{aligned}$$

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

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

$$\overline{f}\big(x, \sigma_1[x,y]\big) = l \wedge \overline{f}\big(x, \sigma_2[x,y,\pi_1(l)]\big) = r \to \overline{f}(x+1,y) = \langle \theta[x,\pi_1(l),\pi_1(r),y], l, r \rangle \tag{2}$$

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

$$\overline{f}(x,y) = M_{2^{x+1}\dot{-}1}\big(x, y, \mathit{Full}(x+1)\big).$$

1.7.11 Theorem *Primitive recursive functions are closed under nested simple recursion for the case $k = 2$ and $n = 1$.*

Proof. Let f be defined by nested simple recursion from p.r. functions as in Par. 1.7.4. Let further $\overline{f}$ be its course of values function as in Par. 1.7.10. We claim that we have

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

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

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

$$f(0,y) = \rho[y] = \pi_1 \langle g(y), 0, 0 \rangle \overset{1.7.10(1)}{=} \pi_1 \overline{f}(0,y).$$

In the induction step take any y and we obtain

$$\begin{aligned} f(x+1,y) &= \theta\Big[x, f\big(x, \sigma_1[x,y]\big), f\Big(x, \sigma_2\big[x,y,f(x,\sigma_1[x,y])\big]\Big), y\Big] \overset{\text{IH}}{=} \\ &= \theta\Big[x, f\big(x, \sigma_1[x,y]\big), \pi_1\overline{f}\Big(x, \sigma_2\big[x,y,f(x,\sigma_1[x,y])\big]\Big), y\Big] \overset{\text{IH}}{=} \\ &= \theta\Big[x, \pi_1\overline{f}\big(x, \sigma_1[x,y]\big), \pi_1\overline{f}\Big(x, \sigma_2\big[x,y,\pi_1\overline{f}(x,\sigma_1[x,y])\big]\Big), y\Big] \overset{1.7.10(2)}{=} \\ &= \pi_1 \overline{f}(x+1,y). \end{aligned}$$

□

Nested Simple Recursion: Case $n = 1$

1.7.12 Introduction. In this subsection we will investigate the schema of nested simple recursion with one parameter ($n = 1$) with arbitrary number

recursive applications:

$$f(0,y) = \rho[y]$$
$$\bigwedge_{i=1}^{k+1} f\big(x,\sigma_i[x,y,\vec{z}_{i-1}]\big) = z_i \to f(x+1,y) = \theta[x,z_1,\ldots,z_{k+1},y].$$

The closure of p.r. functions under the schema will be shown in in Thm. 1.7.13.

1.7.13 Theorem *Primitive recursive functions are closed under nested simple recursion for the case $n = 1$.*

Proof. Similar to the proof of Thm. 1.6.13. □

Nested Simple Recursion: General Case

1.7.14 Introduction. In this subsection we will investigate the schema of nested simple recursion with arbitrary number of parameters and recursive applications:

$$f(0,\vec{y}) = \rho[\vec{y}]$$
$$\bigwedge_{i=1}^{k} f\big(x,\vec{\sigma}_i[x,\vec{y},\vec{z}_{i-1}]\big) = z_i \to f(x+1,\vec{y}) = \theta[x,\vec{z},\vec{y}].$$

The closure of p.r. functions under the schema will be shown in Thm. 1.7.15.

1.7.15 Theorem *Primitive recursive functions are closed under nested simple recursion.*

Proof. Similar to the proof of Thm. 1.6.16. □