2.2 Beyond Primitive Recursion: Universal Function for Primitive Recursive Functions

2.2.1 Introduction. In this section we give yet another example of an intuitively computable function which is not primitive recursive. The function in question is called universal function for primitive recursive functions. Its construction requires arithmetization of primitive recursive function symbols.

2.2.2 Primitive recursive function symbols. For every $n \geq 1$, the class PR^n of n-ary *primitive recursive function symbols* is defined inductively as follows:

- $Z \in \mathrm{PR}^1$, $S \in \mathrm{PR}^1$ and $I^n_i \in \mathrm{PR}^n$ for $1 \leq i \leq n$,
- if $h \in \mathrm{PR}^m$ and $g_1, \ldots, g_m \in \mathrm{PR}^n$ then $Comp^n_m(h, g_1, \ldots, g_m) \in \mathrm{PR}^n$,
- if $g \in \mathrm{PR}^n$ and $h \in \mathrm{PR}^{n+2}$ then $Rec_{n+1}(g, h) \in \mathrm{PR}^{n+1}$.

We set $\mathrm{PR} = \bigcup_{n \geq 1} \mathrm{PR}^n$.

We interpret n-ary p.r. function symbols by n-ary functions. The interpretation $f^{\mathcal{N}}$ of a p.r. function symbol f is defined by induction on the structure of p.r. function symbols as follows:

- $Z^{\mathcal{N}}$ is the zero function $Z(x) = 0$,
- $S^{\mathcal{N}}$ is the successor function $S(x) = x + 1$,
- $(I^n_i)^{\mathcal{N}}$ is the identity function $I^n_i(\vec{x}) = x_i$,
- $\big(Comp^n_m(h, g_1, \ldots, g_m)\big)^{\mathcal{N}}$ is the n-ary function defined by composition:

$$\big(Comp^n_m(h, g_1, \ldots, g_m)\big)^{\mathcal{N}}(\vec{x}) = h^{\mathcal{N}}\big(g_1^{\mathcal{N}}(\vec{x}), \ldots, g_m^{\mathcal{N}}(\vec{x})\big),$$

- $\big(Rec_n(g, h)\big)^{\mathcal{N}}$ is the n-ary function defined by primitive recursion:

$$\begin{aligned} \big(Rec_n(g, h)\big)^{\mathcal{N}}(0, \vec{y}) &= g^{\mathcal{N}}(\vec{y}) \\ \big(Rec_n(g, h)\big)^{\mathcal{N}}(x + 1, \vec{y}) &= h^{\mathcal{N}}\big(x, \big(Rec_n(g, h)\big)^{\mathcal{N}}(x, \vec{y}), \vec{y}\big). \end{aligned}$$

In the sequel we will often drop the superscript in $f^{\mathcal{N}}$ and write shortly f instead of $f^{\mathcal{N}}$.

It is easy to see that primitive recursive functions are exactly those functions which are denoted by p.r. function symbols. In other words, the class of primitive recursive functions is just the set $\bigcup_{n \geq 1} \{f^{\mathcal{N}} \mid f \in \mathrm{PR}^n\}$.

2.2.3 Arithmetization of primitive recursive function symbols. Now we consider the problem of coding of p.r. function symbols into N. The symbols are arithmetized with the help of the following pair constructors:

$$\begin{aligned}
\boldsymbol{Z} &= \langle 0, 0\rangle && \text{(zero)}\\
\boldsymbol{S} &= \langle 1, 0\rangle && \text{(successor)}\\
\boldsymbol{I}_i^n &= \langle 2, n, i\rangle && \text{(identities)}\\
\langle\!\langle g, gs\rangle\!\rangle &= \langle 3, g, gs\rangle && \text{(contraction)}\\
\boldsymbol{Comp}_m^n(h, gs) &= \langle 4, n, m, h, gs\rangle && \text{(composition)}\\
\boldsymbol{Rec}_n(g, h) &= \langle 5, n, g, h\rangle. && \text{(primitive recursion)}
\end{aligned}$$

The arities of the constructors are as shown in their definitions. We postulate that the binary constructor $\langle\!\langle g, gs\rangle\!\rangle$ groups to the right and has the same precedence as the pairing function $\langle x, y\rangle$.

The assignment of the code $\ulcorner f \urcorner$ to the p.r. function symbol f is defined inductively on the structure of p.r. function symbols:

$$\begin{aligned}
\ulcorner Z \urcorner &= \boldsymbol{Z}\\
\ulcorner S \urcorner &= \boldsymbol{S}\\
\ulcorner I_i^n \urcorner &= \boldsymbol{I}_i^n\\
\ulcorner Comp_m^n(h, g_1, \ldots, g_m) \urcorner &= \boldsymbol{Comp}_m^n\big(\ulcorner h \urcorner, \langle\!\langle \ulcorner g_1 \urcorner, \ldots, \ulcorner g_m \urcorner \rangle\!\rangle\big)\\
\ulcorner Rec_n(g, h) \urcorner &= \boldsymbol{Rec}_n(\ulcorner g \urcorner, \ulcorner h \urcorner).
\end{aligned}$$

Note that the binary operator $\langle\!\langle g, gs\rangle\!\rangle$ plays a similar role as the pairing function $\langle x, y\rangle$ does for n-tuples of natural numbers. Its sole purpose is to represent the m-tuple $\ulcorner g_1 \urcorner, \ldots, \ulcorner g_m \urcorner$ of the codes of p.r. function symbols by its *contraction* which is the number of the form $\langle\!\langle \ulcorner g_1 \urcorner, \ldots, \ulcorner g_m \urcorner \rangle\!\rangle$.

2.2.4 Interpreter of primitive recursive functions. In this paragraph we give a definition of a binary function $e \bullet x$ which effectively realizes the interpretation of p.r. function symbols. The application $\ulcorner f \urcorner \bullet \langle x_1, \ldots, x_n\rangle$ takes the code of an n-ary p.r. function symbol f and the contraction of an n-tuple $x_1, \ldots, x_n$ of numbers, and yields the number $f(x_1, \ldots, x_n)$ as the result, i.e.

$$\ulcorner f \urcorner \bullet \langle x_1, \ldots, x_n\rangle = f(x_1, \ldots, x_n).$$

To improve readability we will write $e_1 \bullet e_2 \bullet x$ instead of $e_1 \bullet (e_2 \bullet x)$, that is we let the operator associates right.

The interpreter $e \bullet x$ of primitive recursive functions is defined by

$$\begin{aligned}
&\boldsymbol{Z} \bullet x = 0\\
&\boldsymbol{S} \bullet x = x + 1\\
&\boldsymbol{I}_i^n \bullet x = [x]_i^n\\
&\langle\!\langle g, gs\rangle\!\rangle \bullet x = \langle g \bullet x, gs \bullet x\rangle\\
&\boldsymbol{Comp}_m^n(h, gs) \bullet x = h \bullet gs \bullet x\\
&\boldsymbol{Rec}_n(g, h) \bullet \langle 0, y\rangle = g \bullet y\\
&\boldsymbol{Rec}_n(g, h) \bullet \langle\!\langle x + 1, y\rangle\!\rangle = h \bullet \langle\!\langle x, \boldsymbol{Rec}_n(g, h) \bullet \langle x, y\rangle, y\rangle\!\rangle.
\end{aligned}$$

This is an example of regular recursive definition which is into the lexicographical order $(x_1, y_1) <_{\text{lex}} (x_2, y_2)$ of natural numbers. This is because the first argument of each recursive application except the one in the last recursive clause goes down. In the recursive application of the last recursive clause the first argument $\boldsymbol{Rec}_n(g, h)$ stays the same and the second argument goes down since $\langle x, y\rangle < \langle x + 1, y\rangle$. We have therefore

$$\big(\boldsymbol{Rec}_n(g, h), \langle x, y\rangle\big) <_{\text{lex}} \big(\boldsymbol{Rec}_n(g, h), \langle x + 1, y\rangle\big).$$

From the results of this chapter, we obtain that the interpreter is effectively computable; the closed form of the above definition constitutes a program for the reduction model discussed later. As we will see below the function is not primitive recursive.

2.2.5 Enumeration functions. The $(n{+}1)$-ary function θ is said to be an *enumeration function* for the class of n-ary functions $\mathcal{F}$ if we have the following for every n-ary function f:

$f \in \mathcal{F}$ iff there is a number e such that

$$f(x_1, \ldots, x_n) = \theta(e, x_1, \ldots, x_n)$$

holds for all numbers $x_1, \ldots, x_n$.

The function θ is often called the *universal function* for the class $\mathcal{F}$.

In the next theorem we will prove for every $n \geq 1$ that the $(n{+}1)$-ary function U_n explicitly defined by

$$U_n(e, x_1, \ldots, x_n) = e \bullet \langle x_1, \ldots, x_n\rangle$$

is the enumeration function for the class of n-ary primitive recursive functions. Since the function $e \bullet x$ is effectively computable, so is U_n. Note also that U_2 and $e \bullet x$ are the same functions. In the sequel we will often abbreviate $U_1(e, x)$ to $U(e, x)$.

2.2.6 Enumeration theorem. *For every $n \geq 1$, the U_n is an effectively computable function enumerating the class of n-ary primitive recursive functions.*

Proof. We wish to prove that the following holds for every n-ary function f:

the function f is primitive recursive iff there is a number e such that

$$f(x_1, \ldots, x_n) = U_n(e, x_1, \ldots, x_n)$$

for every $x_1, \ldots, x_n$.

For the proof of the $(\Rightarrow)$-part of the claim it suffices to show that for every n-ary p.r. function symbol f and every $x_1, \ldots, x_n$ we have

$$f(x_1, \ldots, x_n) = \ulcorner f \urcorner \bullet \langle x_1, \ldots, x_n \rangle.$$

This is proved by induction on the structure of p.r. function symbols. So take any n-ary p.r. function symbol f, any n-tuple $\vec{x}$ of numbers, and continue by the case analysis of f. The cases when $f \equiv Z$ or $f \equiv S$ are straightforward. Now, if $f \equiv Comp^n_m(h, g_1, \ldots, g_m)$ then we have

$$\begin{aligned}
&\ulcorner Comp^n_m(h, g_1, \ldots, g_m) \urcorner \bullet \langle \vec{x} \rangle = \boldsymbol{Comp}^n_m(\ulcorner h \urcorner, \langle \ulcorner g_1 \urcorner, \ldots, \ulcorner g_m \urcorner \rangle) \bullet \langle \vec{x} \rangle = \\
&\quad = \ulcorner h \urcorner \bullet \langle \ulcorner g_1 \urcorner, \ldots, \ulcorner g_m \urcorner \rangle \bullet \langle \vec{x} \rangle = \ulcorner h \urcorner \bullet \big\langle \ulcorner g_1 \urcorner \bullet \langle \vec{x} \rangle, \ldots, \ulcorner g_m \urcorner \bullet \langle \vec{x} \rangle \big\rangle \overset{\text{IH}}{=} \\
&\quad = h\big(g_1(\vec{x}), \ldots, g_m(\vec{x})\big) = Comp^n_m(h, g_1, \ldots, g_m)(\vec{x}).
\end{aligned}$$

Finally, if $f \equiv Rec_n(g, h)$ then $\vec{x} \equiv z, \vec{y}$ for some z and a non-empty $\vec{y}$. The desired property

$$\ulcorner Rec_n(g, h) \urcorner \bullet \langle z, \vec{y} \rangle = Rec_n(g, h)(z, \vec{y})$$

is proved by (inner) induction on z. In the base case we have

$$\begin{aligned}
\ulcorner Rec_n(g, h) \urcorner \bullet \langle 0, \vec{y} \rangle = \boldsymbol{Rec}_n(\ulcorner g \urcorner, \ulcorner h \urcorner) \bullet \langle 0, \vec{y} \rangle = \ulcorner g \urcorner \bullet \langle \vec{y} \rangle &\overset{\text{outer IH}}{=} \\
= g(\vec{y}) = Rec_n(g, h)(0, \vec{y}).&
\end{aligned}$$

In the induction step we have

$$\begin{aligned}
&\ulcorner Rec_n(g, h) \urcorner \bullet \langle z + 1, \vec{y} \rangle = \boldsymbol{Rec}_n(\ulcorner g \urcorner, \ulcorner h \urcorner) \bullet \langle z + 1, \vec{y} \rangle = \\
&\quad = \ulcorner h \urcorner \bullet \big\langle z, \boldsymbol{Rec}_n(\ulcorner g \urcorner, \ulcorner h \urcorner) \bullet \langle z, \vec{y} \rangle, \vec{y} \big\rangle \overset{\text{outer IH}}{=} \\
&\quad = h\big(z, \boldsymbol{Rec}_n(\ulcorner g \urcorner, \ulcorner h \urcorner) \bullet \langle z, \vec{y} \rangle, \vec{y}\big) = h\big(z, \ulcorner Rec_n(g, h) \urcorner \bullet \langle z, \vec{y} \rangle, \vec{y}\big) \overset{\text{inner IH}}{=} \\
&\quad = h\big(z, Rec_n(g, h)(z, \vec{y}), \vec{y}\big) = Rec_n(g, h)(z + 1, \vec{y}).
\end{aligned}$$

This finishes the proof for the case when $f \equiv Rec_n(g, h)$.

In the proof of ($\Leftarrow$)-part of the claim it suffices to show that the unary functions ϕ_e explicitly defined by

$$\phi_e(x) = e \bullet x$$

are primitive recursive functions. This is proved by complete induction on e. So take any e and continue by case analysis on e. If $e = \boldsymbol{Z}$, $e = \boldsymbol{S}$ or $e = \boldsymbol{I}^n_i$ then the following explicit definitions listed in that order

$$\begin{aligned}
\phi_e(x) &= 0 \\
\phi_e(x) &= x + 1 \\
\phi_e(x) &= [x]^n_i
\end{aligned}$$

are derivations of ϕ_e as a primitive recursive function. If $e = \boldsymbol{Comp}^n_m(e_1, e_2)$ for some e_1 and e_2 then the functions ϕ_{e_1} and ϕ_{e_2} are primitive recursive by

IH and we derive ϕ_e as a primitive recursive function by composition:

$$\phi_e(x) = \phi_{e_1}\,\phi_{e_2}(x).$$

If $e = \boldsymbol{Rec}_n(e_1, e_2)$ for some e_1 and e_2 then the functions ϕ_{e_1} and ϕ_{e_2} are primitive recursive by IH and we derive ϕ_e as a primitive recursive function by the following course of values recursive definition:

$$\begin{aligned}\phi_e(0) &= 0\\ \phi_e\langle 0, y\rangle &= \phi_{e_1}(y)\\ \phi_e\langle x+1, y\rangle &= \phi_{e_2}\big\langle x, \phi_e\langle x, y\rangle, y\big\rangle.\end{aligned}$$

If neither of the above cases applies then we derive ϕ_e as a primitive recursive function by explicit definition:

$$\phi_e(x) = 0. \qquad \square$$

2.2.7 Enumeration functions are not primitive recursive. We already know that the enumeration functions U_n are effectively computable. In this paragraph none of the enumeration functions is primitive recursive. We prove this fact for the case when $n = 1$ and left the proof the general result to the reader.

The standard proof uses a diagonal argument. Suppose by contradiction that the enumeration function $U(e, x)$ is primitive recursive. Then also the explicitly defined function f:

$$f(x) = U(x, x) + 1 \tag{1}$$

is a primitive recursive function. By the Enumeration Theorem there is a number e such that for every number x we have

$$f(x) = U(e, x). \tag{2}$$

We obtain contradiction from

$$f(e) \overset{(1)}{=} U(e, e) + 1 \overset{(2)}{=} f(e) + 1.$$

2.2.8 Primitive recursive indices. We say that a number e is a *primitive recursive index* of the n-ary function f if

$$f(x_1, \ldots, x_n) = U_n(e, x_1, \ldots, x_n)$$

for all numbers $x_1, \ldots, x_n$. This can be expressed equivalently by

$$f(x_1, \ldots, x_n) = e \bullet \langle x_1, \ldots, x_n\rangle.$$

Primitive index is said to be *well-formed* if it is a code of some p.r. function symbol. As a simple corollary of the Enumeration Theorem we can see that a function is primitive recursive iff it has a primitive recursive index.

For every $n \geq 1$ by $\phi_e^{(n)}$ we denote the n-ary primitive recursive function with the primitive recursive index e. Note that we then have

$$\phi_e^{(n)}(x_1, \ldots, x_n) = U_n(e, x_1, \ldots, x_n).$$

and

$$\phi_e^{(n)}(x_1, \ldots, x_n) = e \bullet \langle x_1, \ldots, x_n \rangle.$$

By the Enumeration Theorem, an n-ary function f is primitive recursive iff $f = \phi_e^{(n)}$ for some e. In the sequel we will often abbreviate $\phi_e^{(1)}(\vec{x})$ to $\phi_e(\vec{x})$.

2.2.9 Indices of initial primitive recursive functions. We clearly have

$$\begin{aligned} \boldsymbol{Z} \bullet x &= 0 \\ \boldsymbol{S} \bullet x &= x + 1 \\ \boldsymbol{I}_i^n \bullet \langle x_1, \ldots, x_n \rangle &= x_i \end{aligned}$$

and therefore the numbers $\boldsymbol{Z}$, $\boldsymbol{S}$ and $\boldsymbol{I}_i^n$ are p.r. indices of the initial p.r. functions $Z(x) = 0$, $S(x) = x + 1$ and $I_i^n(\vec{x}) = x_i$, respectively.

2.2.10 Indices of constant functions. There is a unary p.r. function $\boldsymbol{C}_m$ which yields p.r. indices of unary constant functions $C_m(x) = m$, i.e. we have

$$\phi_{\boldsymbol{C}_m}(x) = m$$

for every x. The property can be easily expressed in the elementary logic by

$$\boldsymbol{C}_m \bullet x = m \tag{1}$$

Note that we have

$$\begin{aligned} C_0(x) &= 0 \\ C_{m+1}(x) &= S\, C_m(x) \end{aligned}$$

and thus the function $\boldsymbol{C}_m$ has the following primitive recursive definition:

$$\begin{aligned} \boldsymbol{C}_0 &= \boldsymbol{Z} \\ \boldsymbol{C}_{m+1} &= \boldsymbol{Comp}_1^1(\boldsymbol{S}, \boldsymbol{C}_m). \end{aligned}$$

There is a binary primitive recursive function $\boldsymbol{C}_m^n$ which yields p.r. indices of n-ary constant functions $C_m^n(\vec{x}) = m$, i.e. we have

$$\phi_{\boldsymbol{C}_m^n}(x_1, \ldots, x_n) = m.$$

This is expressed in the elementary logic by

$$\boldsymbol{C}_m^n \bullet \langle x_1, \ldots, x_n \rangle = m \tag{2}$$

Note that we have

$$C_m^n(x_1, \ldots, x_n) = C_m\, I_1^n(x_1, \ldots, x_n)$$

and thus the function $\boldsymbol{C}_m^n$ has the following primitive recursive definition:

$$\boldsymbol{C}_m^n = \boldsymbol{Comp}_1^n(\boldsymbol{C}_m, \boldsymbol{I}_1^n).$$

Verification. (1): By induction on m. The base case is trivial and the induction step follows from

$$\boldsymbol{C}_{m+1} \bullet x = \boldsymbol{Comp}_1^1(\boldsymbol{S}, \boldsymbol{C}_m) \bullet x = \boldsymbol{S} \bullet \boldsymbol{C}_m \bullet x \overset{\text{IH}}{=} \boldsymbol{S} \bullet m = m + 1.$$

(2): It follows from

$$\begin{aligned} \boldsymbol{C}_m^n \bullet \langle x_1, \ldots, x_n \rangle &= \boldsymbol{Comp}_1^n(\boldsymbol{C}_m, \boldsymbol{I}_1^n) \bullet \langle x_1, \ldots, x_n \rangle = \\ &= \boldsymbol{C}_m \bullet \boldsymbol{I}_1^n \bullet \langle x_1, \ldots, x_n \rangle = \boldsymbol{C}_m \bullet x_1 \overset{(1)}{=} m. \end{aligned}$$

□

2.2.11 Explicit definitions. Now we consider the problem of finding p.r. indices of functions defined by explicit definitions:

$$f(x_1, \ldots, x_n) = \tau[x_1, \ldots, x_n] \tag{1}$$

We suppose here that the term τ is built up from from variables and constants by applications of p.r. function symbols.

The primitive recursive index $\ulcorner \lambda \vec{x}.\tau \urcorner$ of the function f defined by (1) is constructed inductively on the structure of the term τ as follows:

$$\begin{aligned} \ulcorner \lambda \vec{x}.x_i \urcorner &= \boldsymbol{I}_i^n \\ \ulcorner \lambda \vec{x}.m \urcorner &= \boldsymbol{C}_m^n \\ \ulcorner \lambda \vec{x}.g(\tau_1, \ldots, \tau_m) \urcorner &= \boldsymbol{Comp}_m^n\big(\ulcorner g \urcorner, \langle \ulcorner \lambda \vec{x}.\tau_1 \urcorner, \ldots, \ulcorner \lambda \vec{x}.\tau_m \urcorner \rangle\big). \end{aligned}$$

We have

$$\ulcorner \lambda \vec{x}.\tau \urcorner \bullet \langle x_1, \ldots, x_n \rangle = \tau[x_1, \ldots, x_n]. \tag{2}$$

Proof. Property (2) is proved by (meta-)induction on the structure of the term τ. If $\tau \equiv x_i$ then we have

$$\ulcorner \lambda \vec{x}.x_i \urcorner \bullet \langle x_1, \ldots, x_n \rangle = \boldsymbol{I}_i^n \bullet \langle x_1, \ldots, x_n \rangle = x_i.$$

Similarly, if $\tau \equiv m$ then we have

$$\ulcorner\lambda\vec{x}.m\urcorner \bullet \langle x_1, \ldots, x_n\rangle = \boldsymbol{C}_m^n \bullet \langle x_1, \ldots, x_n\rangle \overset{2.2.10(2)}{=} m.$$

Finally, if $\tau \equiv g(\tau_1, \ldots, \tau_m)$ then we have

$$\begin{aligned}
&\ulcorner\lambda\vec{x}.g(\tau_1, \ldots, \tau_m)\urcorner \bullet \langle x_1, \ldots, x_n\rangle = \\
&\quad = \boldsymbol{Comp}_m^n\big(\ulcorner g\urcorner, \big\langle \ulcorner\lambda\vec{x}.\tau_1\urcorner, \ldots, \ulcorner\lambda\vec{x}.\tau_m\urcorner\big\rangle\big) \bullet \langle x_1, \ldots, x_n\rangle = \\
&\quad = \ulcorner g\urcorner \bullet \big\langle \ulcorner\lambda\vec{x}.\tau_1\urcorner, \ldots, \ulcorner\lambda\vec{x}.\tau_m\urcorner\big\rangle \bullet \langle x_1, \ldots, x_n\rangle = \\
&\quad = \ulcorner g\urcorner \bullet \big\langle \ulcorner\lambda\vec{x}.\tau_1\urcorner \bullet \langle x_1, \ldots, x_n\rangle, \ldots, \ulcorner\lambda\vec{x}.\tau_m\urcorner \bullet \langle x_1, \ldots, x_n\rangle\big\rangle \overset{\text{IH}}{=} \\
&\quad = \ulcorner g\urcorner \bullet \langle \tau_1[x_1, \ldots, x_n], \ldots, \tau_m[x_1, \ldots, x_n]\rangle = \\
&\quad = g(\tau_1, \ldots, \tau_m)[x_1, \ldots, x_n].
\end{aligned}$$

□

2.2.12 Example. Addition is defined by primitive recursion (cf. Par. 1.2.6)

$$\begin{aligned}
0 + y &= I(y) \\
(x + 1) + y &= h(x, x + y, y)
\end{aligned}$$

from the identity function $I(y) = y$ and the ternary function $h(x, a, y) = S(a)$. By Par. 2.2.11, $\ulcorner\lambda x_1 x_2 x_3.S(x_2)\urcorner$ is a p.r. index of h and therefore the number

$$\boldsymbol{Rec}_2(\boldsymbol{I}_1^1, \ulcorner\lambda x_1 x_2 x_3.S(x_2)\urcorner)$$

is a p.r. index of the addition. Primitive recursive indices of some other p.r. functions (e.g. multiplication) are obtained similarly (cf. Sect. 1.2).

2.2.13 Parametric function. The binary *parametric* function e/x takes a p.r. index e of a binary p.r. f and a number x and yields a p.r. index of the unary p.r. function g such that $g(y) = f(x, y)$, i.e. we have

$$\phi_{e/x}(y) = \phi_e^{(2)}(x, y).$$

This is expressed in the elementary logic by

$$(e/x) \bullet y = e \bullet \langle x, y\rangle. \tag{1}$$

The parametric function is defined explicitly as a p.r. function by

$$e/x = \boldsymbol{Comp}_2^1\big(e, \big\langle \boldsymbol{C}_x, \boldsymbol{I}_1^1\big\rangle\big).$$

Verification. Property (1) follows from

$$\begin{aligned}
(e/x) \bullet y = \boldsymbol{Comp}_2^1\big(e, \big\langle \boldsymbol{C}_x, \boldsymbol{I}_1^1\big\rangle\big) \bullet y = e \bullet \big\langle \boldsymbol{C}_x, \boldsymbol{I}_1^1\big\rangle \bullet y = \\
= e \bullet \langle \boldsymbol{C}_x \bullet y, \boldsymbol{I}_1^1 \bullet y\rangle \overset{2.2.10(1)}{=} e \bullet \langle x, y\rangle.
\end{aligned}$$

□

2.2.14 S-m-n theorem. For every $m, n \geq 1$, there exists an $(m{+}1)$-ary primitive recursive function $\mathrm{s}_n^m(e, \vec{x})$ such that

$$\phi^{(n)}_{\mathrm{s}_n^m(e,\vec{x})}(\vec{y}) = \phi_e^{(m+n)}(\vec{x}, \vec{y}).$$

This is expressed in the elementary logic by

$$\mathrm{s}_n^m(e, x_1, \ldots, x_m) \bullet \langle y_1, \ldots, y_n \rangle = e \bullet \langle x_1, \ldots, x_m, y_1, \ldots, y_n \rangle. \tag{1}$$

The function $\mathrm{s}_n^m(e, \vec{x})$ is defined explicitly as p.r. function by

$$\mathrm{s}_n^m(e, x_1, \ldots, x_m) = \boldsymbol{Comp}^n_{m+n}\big(e, \langle \boldsymbol{C}^n_{x_1}, \ldots, \boldsymbol{C}^n_{x_m}, \boldsymbol{I}^n_1, \ldots, \boldsymbol{I}^n_n \rangle\big).$$

Verification. Property (1) is proved as follows ($\vec{y} \equiv y_1, \ldots, y_m$):

$$\begin{aligned}
&\mathrm{s}_n^m(e, x_1, \ldots, x_m) \bullet \langle y_1, \ldots, y_n \rangle = \\
&\quad = \boldsymbol{Comp}^n_{m+n}\big(e, \langle \boldsymbol{C}^n_{x_1}, \ldots, \boldsymbol{C}^n_{x_m}, \boldsymbol{I}^n_1, \ldots, \boldsymbol{I}^n_n \rangle\big) \bullet \langle y_1, \ldots, y_n \rangle = \\
&\quad = e \bullet \langle \boldsymbol{C}^n_{x_1}, \ldots, \boldsymbol{C}^n_{x_m}, \boldsymbol{I}^n_1, \ldots, \boldsymbol{I}^n_n \rangle \bullet \langle y_1, \ldots, y_n \rangle = \\
&\quad = e \bullet \big\langle \boldsymbol{C}^n_{x_1} \bullet \langle \vec{y} \rangle, \ldots, \boldsymbol{C}^n_{x_m} \bullet \langle \vec{y} \rangle, \boldsymbol{I}^n_1 \bullet \langle \vec{y} \rangle, \ldots, \boldsymbol{I}^n_n \bullet \langle \vec{y} \rangle \big\rangle \overset{2.2.10(2)}{=} \\
&\quad = e \bullet \langle x_1, \ldots, x_m, y_1, \ldots, y_n \rangle.
\end{aligned}$$

□

2.2.15 Self-reproducing machine. We conclude this section by solving the following question. Does exist a primitive recursive function which produces its own description? More precisely, we wish to find a unary recursive function $\phi_e(x)$ which yields its own well-formed index e for every input x, i.e. we would like to have

$$\phi_e(x) = e,$$

or equivalently

$$e \bullet x = e. \tag{1}$$

For that consider the following unary p.r. function defined explicitly by

$$g(y) = \boldsymbol{Comp}^1_1(y, \boldsymbol{C}_y). \tag{2}$$

Let $\ulcorner g \urcorner$ be one of its well-formed p.r. indices. Then the number

$$e = g(\ulcorner g \urcorner) \tag{3}$$

is a well-formed index satisfying (1):

$$e \bullet x \overset{(3)}{=} g(\ulcorner g \urcorner) \bullet x \overset{(2)}{=} \boldsymbol{Comp}^1_1(\ulcorner g \urcorner, \boldsymbol{C}_{\ulcorner g \urcorner}) \bullet x = \ulcorner g \urcorner \bullet \boldsymbol{C}_{\ulcorner g \urcorner} \bullet x \overset{2.2.10(1)}{=}$$
$$= \ulcorner g \urcorner \bullet \ulcorner g \urcorner \overset{\text{index}}{=} g(\ulcorner g \urcorner) \overset{(3)}{=} e.$$