

## Exercise 5: Arithmetization of Reductions

**5.1 Arithmetization of recursive terms.** We arithmetize R-terms and R-functions symbols with the following pair constructors:

$x_i = \langle 0, i \rangle$	(variables)
$\mathbf{0} = \langle 1, 0 \rangle$	(zero)
$\mathbf{S}(t) = \langle 2, t \rangle$	(successor)
$\mathbf{P}(t) = \langle 3, t \rangle$	(predecessor)
$\mathbf{D}(t_1, t_2, t_3) = \langle 4, t_1, t_2, t_3 \rangle$	(conditional)
$t_1 \bullet t_2 = \langle 5, t_1, t_2 \rangle$	(curried application)
$e[ts] = \langle 6, e, ts \rangle$	(partial application)
$\mathbf{f}_n = \langle 0, n \rangle$	(recursors)
$\lambda_n \cdot t = \langle 1, n, t \rangle$	(defined functions)

We postulate that the binary constructor  $\bullet$  groups to the left, i.e. that  $t_1 \bullet t_2 \bullet t_3$  abbreviates  $(t_1 \bullet t_2) \bullet t_3$ .

We assign to every R-term  $\tau$  and to every R-function symbol  $f$  their codes  $\ulcorner \tau \urcorner$  and  $\ulcorner f \urcorner$  inductively as follows:

$$\ulcorner x_i \urcorner = x_i \quad (1)$$

$$\ulcorner 0 \urcorner = \mathbf{0} \quad (2)$$

$$\ulcorner \tau + 1 \urcorner = \mathbf{S}(\ulcorner \tau \urcorner) \quad (3)$$

$$\ulcorner \tau \div 1 \urcorner = \mathbf{P}(\ulcorner \tau \urcorner) \quad (4)$$

$$\ulcorner \mathbf{D}(\tau_1, \tau_2, \tau_3) \urcorner = \mathbf{D}(\ulcorner \tau_1 \urcorner, \ulcorner \tau_2 \urcorner, \ulcorner \tau_3 \urcorner) \quad (5)$$

$$\ulcorner f(\tau_1, \dots, \tau_n) \urcorner = \ulcorner f \urcorner[\langle \ulcorner \tau_1 \urcorner, \dots, \ulcorner \tau_k \urcorner, 0 \rangle] \bullet \ulcorner \tau_{k+1} \urcorner \bullet \dots \bullet \ulcorner \tau_n \urcorner \quad (6)$$

where  $k$  is the maximal number such that  
the terms  $\tau_1, \dots, \tau_k$  are numerals

$$\ulcorner \mathbf{f}_n \urcorner = \mathbf{f}_n \quad (7)$$

$$\ulcorner \lambda_n \cdot \tau \urcorner = \lambda_n \cdot \ulcorner \tau \urcorner. \quad (8)$$

**5.2 Codes of numerals.** Applications of functions are reduced when their arguments are numerals. In order to recognize when the codes of arguments are already reduced we will need a unary predicate  $Nm$  holding of the codes of numerals, i.e.  $Nm(t) \leftrightarrow \exists x t = \ulcorner x \urcorner$ . The predicate is primitive recursive by parameterless course of values recursive definition:

$$\begin{aligned} Nm(\mathbf{0}) \\ Nm(\mathbf{S}(t)) \leftarrow Nm(t). \end{aligned}$$

We will need a unary *coding* function  $\ulcorner \underline{x} \urcorner$  which takes a number  $x$  and yields the code of the numeral  $\underline{x}$ . The function is primitive recursive by primitive recursive definition:

$$\begin{aligned} \ulcorner \underline{0} \urcorner &= \mathbf{0} \\ \ulcorner \underline{x+1} \urcorner &= \mathbf{S}(\ulcorner \underline{x} \urcorner). \end{aligned}$$

Its inverse  $Dc(t)$ , called the *decoding* function, satisfies

$$Dc(\ulcorner \underline{x} \urcorner) = x. \quad (1)$$

The function is primitive recursive by parameterless course of values recursive definition:

$$\begin{aligned} Dc(\mathbf{0}) &= 0 \\ Dc \mathbf{S}(t) &= Dc(t) + 1. \end{aligned}$$

**5.3 Contraction function.** The binary *contraction* function  $t_1 \bullet t_2$  associating to the left satisfies the identity

$$\ulcorner f(\tau_1, \dots, \tau_n) \urcorner = \ulcorner f \urcorner [\ulcorner \tau_1 \urcorner, \dots, \ulcorner \tau_k \urcorner, 0 \urcorner] \bullet \ulcorner \tau_{k+1} \urcorner \bullet \dots \bullet \ulcorner \tau_n \urcorner, \quad (1)$$

where the terms  $\tau_1, \dots, \tau_k$  are numerals, and it is defined by explicit definition as a primitive recursive function:

$$\begin{aligned} e[ts] \bullet t_2 &= e[ts \oplus \langle t_2, 0 \rangle] \leftarrow Nm(t_2) \\ t_1 \bullet t_2 &= t_1 \bullet t_2 \leftarrow \neg(\exists e \exists ts \, t_1 = e[ts] \wedge Nm(t_2)). \end{aligned}$$

**5.4 Arithmetization of substitution function.** The substitution function  $\tau[\lambda_n.\sigma; \underline{x}]$  is over recursive terms. Its arithmetization  $t[e; rs]$  is a ternary function which takes the code  $t$  of the R-term  $\tau[\{n; x_1, \dots, x_n\}$  with all free recursors and free variables indicated, the code  $e$  of the  $n$ -ary function symbol  $\lambda_n.\sigma$  and the list  $rs = \langle \ulcorner \underline{x}_1 \urcorner, \dots, \ulcorner \underline{x}_n \urcorner, 0 \rangle$  of the codes of the numerals  $\underline{x}_1, \dots, \underline{x}_n$ , and yields the code of the R-term  $\tau[\lambda_n.\sigma; \underline{x}_1, \dots, \underline{x}_n]$ , i.e.

$$\ulcorner \tau \urcorner [\ulcorner \lambda_n.\sigma \urcorner; \langle \ulcorner \underline{x}_1 \urcorner, \dots, \ulcorner \underline{x}_n \urcorner, 0 \urcorner \rangle] = \ulcorner \tau[\lambda_n.\sigma; \underline{x}_1, \dots, \underline{x}_n] \urcorner. \quad (1)$$

The arithmetized substitution function is primitive recursive by course of values definition regular in the first argument:

$$\begin{aligned} \mathbf{x}_i[e; rs] &= (rs)_{i \div 1} \\ \mathbf{0}[e; rs] &= \mathbf{0} \\ \mathbf{S}(t)[e; rs] &= \mathbf{S}(t[e; rs]) \\ \mathbf{P}(t)[e; rs] &= \mathbf{P}(t[e; rs]) \\ \mathbf{D}(t_1, t_2, t_3)[e; rs] &= \mathbf{D}(t_1[e; rs], t_2[e; rs], t_3[e; rs]) \\ (t_1 \bullet t_2)[e; rs] &= t_1[e; rs] \bullet t_2[e; rs] \\ \mathbf{f}_n[ts][e; rs] &= e[ts] \\ (\lambda_n.t)[ts][e; rs] &= (\lambda_n.t)[ts]. \end{aligned}$$

**5.5 Auxiliary functions.** We will also need two auxiliary functions  $Pn(t)$  and  $Dn(t_1, t_2, t_3)$  satisfying

$$Pn(\ulcorner \underline{x} \urcorner) = \ulcorner \underline{x} \div 1 \urcorner \quad (1)$$

$$Dn(\ulcorner \underline{x} \urcorner, t_2, t_3) = D(x, t_2, t_3). \quad (2)$$

The functions are defined explicitly as a primitive recursive functions:

$$\begin{aligned} Pn(\mathbf{0}) &= \mathbf{0} \\ Pn \mathbf{S}(t) &= t \end{aligned}$$

$$\begin{aligned} Dn(\mathbf{0}, t_2, t_3) &= t_3 \\ Dn(\mathbf{S}(t_1), t_2, t_3) &= t_2. \end{aligned}$$

**5.6 Arithmetization of one-step reduction.** We intend to define a unary function  $Rd$  satisfying:

$$Rd(\ulcorner \underline{x} \urcorner) = \ulcorner \underline{x} \urcorner \quad (1)$$

$$\text{for every } \rho, \text{ if } \tau \triangleright_1 \rho \text{ then } Rd(\ulcorner \tau \urcorner) = \ulcorner \rho \urcorner. \quad (2)$$

The function  $Rd$  is defined as primitive recursive by parameterless course of values definition:

$$\begin{aligned} Rd(\mathbf{0}) &= \mathbf{0} \\ Rd \mathbf{S}(t) &= \mathbf{S} Rd(t) \\ Rd \mathbf{P}(t) &= Pn(t) \leftarrow Nm(t) \\ Rd \mathbf{P}(t) &= \mathbf{P} Rd(t) \leftarrow \neg Nm(t) \\ Rd \mathbf{D}(t_1, t_2, t_3) &= Dn(t_1, t_2, t_3) \leftarrow Nm(t_1) \\ Rd \mathbf{D}(t_1, t_2, t_3) &= \mathbf{D}(Rd(t_1), t_2, t_3) \leftarrow \neg Nm(t_1) \\ Rd(t_1 \bullet t_2) &= t_1 \bullet Rd(t_2) \leftarrow \exists e \exists ts \ t_1 = e[ts] \\ Rd(t_1 \bullet t_2) &= Rd(t_1) \bullet t_2 \leftarrow \neg \exists e \exists ts \ t_1 = e[ts] \\ Rd(\lambda_n \cdot t)[ts] &= t[\lambda_n \cdot t; ts]. \end{aligned}$$

**5.7 Arithmetization of reductions.** The binary iteration of the reduction function  $Rd^k(t)$  defined by

$$\begin{aligned} Rd^0(t) &= t \\ Rd^{k+1}(t) &= Rd \ Rd^k(t) \end{aligned}$$

is a primitive recursive function. Properties 5.6(1)(2) generalizes to

$$Rd^k(\ulcorner \underline{x} \urcorner) = \ulcorner \underline{x} \urcorner \quad (1)$$

$$\text{for every } \tau, \text{ if } \tau \triangleright_k \rho \text{ then } Rd^k(\ulcorner \tau \urcorner) = \ulcorner \rho \urcorner. \quad (2)$$