Exercise 11: Arithmetization of Reductions

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

$$\begin{aligned}
&\boldsymbol{x}_i = \langle 0, i\rangle && \textit{(variables)}\\
&\mathbf{0} = \langle 1, 0\rangle && \textit{(zero)}\\
&\boldsymbol{S}(t) = \langle 2, t\rangle && \textit{(successor)}\\
&\boldsymbol{P}(t) = \langle 3, t\rangle && \textit{(predecessor)}\\
&\boldsymbol{D}(t_1, t_2, t_3) = \langle 4, t_1, t_2, t_3\rangle && \textit{(conditional)}\\
&t_1 \bullet t_2 = \langle 5, t_1, t_2\rangle && \textit{(curried application)}\\
&e\big[ts\big] = \langle 6, e, ts\rangle && \textit{(partial application)}\\
&\boldsymbol{f}_n = \langle 0, n\rangle && \textit{(recursors)}\\
&\boldsymbol{\lambda}_n\mathbf{.}\, t = \langle 1, n, t\rangle. && \textit{(defined functions)}
\end{aligned}$$

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

$$\ulcorner x_i \urcorner = \boldsymbol{x}_i \tag{1}$$
$$\ulcorner 0 \urcorner = \mathbf{0} \tag{2}$$
$$\ulcorner \tau + 1 \urcorner = \boldsymbol{S}(\ulcorner \tau \urcorner) \tag{3}$$
$$\ulcorner \tau \dot{-} 1 \urcorner = \boldsymbol{P}(\ulcorner \tau \urcorner) \tag{4}$$
$$\ulcorner \mathcal{D}(\tau_1, \tau_2, \tau_3) \urcorner = \boldsymbol{D}(\ulcorner \tau_1 \urcorner, \ulcorner \tau_2 \urcorner, \ulcorner \tau_3 \urcorner) \tag{5}$$
$$\ulcorner f(\tau_1, \ldots, \tau_n) \urcorner = \ulcorner f \urcorner \big[\langle \ulcorner \tau_1 \urcorner, \ldots, \ulcorner \tau_k \urcorner, 0\rangle\big] \bullet \ulcorner \tau_{k+1} \urcorner \bullet \cdots \bullet \ulcorner \tau_n \urcorner \tag{6}$$
where k is the maximal number such that the terms $\tau_1, \ldots, \tau_k$ are numerals
$$\ulcorner \mathfrak{f}_n \urcorner = \boldsymbol{f}_n \tag{7}$$
$$\ulcorner \lambda_n . \tau \urcorner = \boldsymbol{\lambda}_n \mathbf{.}\, \ulcorner \tau \urcorner. \tag{8}$$

11.2 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. $Nm(t) \leftrightarrow \exists x\, t = \ulcorner \underline{x} \urcorner$. The predicate is primitive recursive by parameterless course of values recursive definition:

$Nm(\mathbf{0})$
$Nm\, \boldsymbol{S}(t) \leftarrow Nm(t)$.

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:

$\ulcorner \underline{0} \urcorner = \mathbf{0}$
$\ulcorner \underline{x+1} \urcorner = \boldsymbol{S}(\ulcorner \underline{x} \urcorner)$.

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

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

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

$Dc(\mathbf{0}) = 0$
$Dc\, \boldsymbol{S}(t) = Dc(t) + 1.$

11.3 Contraction function. The binary *contraction* function $t_1 \mathbin{\underline{\bullet}} t_2$ associating to the left satisfies the identity

$$\ulcorner f(\tau_1, \ldots, \tau_n) \urcorner = \ulcorner f \urcorner \big[\langle \ulcorner \tau_1 \urcorner, \ldots, \ulcorner \tau_k \urcorner, 0 \rangle\big] \mathbin{\underline{\bullet}} \ulcorner \tau_{k+1} \urcorner \mathbin{\underline{\bullet}} \cdots \mathbin{\underline{\bullet}} \ulcorner \tau_n \urcorner, \tag{1}$$

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

$e\big[ts\big] \mathbin{\underline{\bullet}} t_2 = e\big[ts \oplus \langle t_2, 0 \rangle\big] \leftarrow Nm(t_2)$
$t_1 \mathbin{\underline{\bullet}} t_2 = t_1 \bullet t_2 \leftarrow \neg\big(\exists e \exists ts\, t_1 = e\big[ts\big] \wedge Nm(t_2)\big).$

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

$$\ulcorner \tau \urcorner \big[\ulcorner \lambda_n.\sigma \urcorner; \langle \ulcorner \underline{x_1} \urcorner, \ldots, \ulcorner \underline{x_n} \urcorner, 0 \rangle\big] = \ulcorner \tau[\lambda_n.\sigma; \underline{x_1}, \ldots, \underline{x_n}] \urcorner. \tag{1}$$

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

$\boldsymbol{x}_i\big[e; rs\big] = (rs)_{i \dot{-} 1}$
$\mathbf{0}\big[e; rs\big] = \mathbf{0}$
$\boldsymbol{S}(t)\big[e; rs\big] = \boldsymbol{S}(t\big[e; rs\big])$
$\boldsymbol{P}(t)\big[e; rs\big] = \boldsymbol{P}(t\big[e; rs\big])$
$\boldsymbol{D}(t_1, t_2, t_3)\big[e; rs\big] = \boldsymbol{D}(t_1\big[e; rs\big], t_2\big[e; rs\big], t_3\big[e; rs\big])$
$(t_1 \bullet t_2)\big[e; rs\big] = t_1\big[e; rs\big] \mathbin{\underline{\bullet}} t_2\big[e; rs\big]$
$\boldsymbol{f}_n\big[ts\big]\big[e; rs\big] = e\big[ts\big]$
$(\boldsymbol{\lambda}_n.\, t)\big[ts\big]\big[e; rs\big] = (\boldsymbol{\lambda}_n.\, t)\big[ts\big].$

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

$$Pn(\ulcorner \underline{x} \urcorner) = \ulcorner \underline{x \dot{-} 1} \urcorner \tag{1}$$

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

The functions are defined explicitly as a primitive recursive functions:

$Pn(\mathbf{0}) = \mathbf{0}$
$Pn\ \boldsymbol{S}(t) = t$

$Dn(\mathbf{0}, t_2, t_3) = t_3$
$Dn(\boldsymbol{S}(t_1), t_2, t_3) = t_2$.

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

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

$$\text{for every } \rho, \text{ if } \tau \rhd_1 \rho \text{ then } Rd(\ulcorner \tau \urcorner) = \ulcorner \rho \urcorner. \tag{2}$$

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

$Rd(\mathbf{0}) = \mathbf{0}$
$Rd\ \boldsymbol{S}(t) = \boldsymbol{S}\ Rd(t)$
$Rd\ \boldsymbol{P}(t) = Pn(t) \leftarrow Nm(t)$
$Rd\ \boldsymbol{P}(t) = \boldsymbol{P}\ Rd(t) \leftarrow \neg Nm(t)$
$Rd\ \boldsymbol{D}(t_1, t_2, t_3) = Dn(t_1, t_2, t_3) \leftarrow Nm(t_1)$
$Rd\ \boldsymbol{D}(t_1, t_2, t_3) = \boldsymbol{D}(Rd(t_1), t_2, t_3) \leftarrow \neg Nm(t_1)$
$Rd(t_1 \bullet t_2) = t_1 \mathbin{\underline{\bullet}} Rd(t_2) \leftarrow \exists e \exists ts\, t_1 = e[ts]$
$Rd(t_1 \bullet t_2) = Rd(t_1) \mathbin{\underline{\bullet}} t_2 \leftarrow \neg\exists e \exists ts\, t_1 = e[ts]$
$Rd\ (\boldsymbol{\lambda}_n.\, t)[ts] = t[\boldsymbol{\lambda}_n.\, t; ts]$.

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

$Rd^0(t) = t$
$Rd^{k+1}(t) = Rd\ Rd^k(t)$

is a primitive recursive function. Properties 11.6(1)(2) generalizes to

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

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