Arithmetization of Reductions

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

$\boldsymbol{x}_i = \langle 0, i\rangle$	*(variables)*
$\boldsymbol{0} = \langle 1, 0\rangle$	*(zero)*
$\boldsymbol{S}(t) = \langle 2, t\rangle$	*(successor)*
$\boldsymbol{P}(t) = \langle 3, t\rangle$	*(predecessor)*
$\boldsymbol{D}(t_1, t_2, t_3) = \langle 4, t_1, t_2, t_3\rangle$	*(conditional)*
$t_1 \bullet t_2 = \langle 5, t_1, t_2\rangle$	*(curried application)*
$e[ts] = \langle 6, e, ts\rangle$	*(partial application)*
$\boldsymbol{f}_n = \langle 7, n\rangle$	*(recursors)*
$\boldsymbol{\lambda}_n\boldsymbol{.}\, t = \langle 8, n, t\rangle.$	*(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 τ and to every R-function symbol f their codes $\ulcorner\tau\urcorner$ and $\ulcorner f\urcorner$ inductively as follows:

$$
\begin{aligned}
\ulcorner x_i\urcorner &= \boldsymbol{x}_i\\
\ulcorner 0\urcorner &= \boldsymbol{0}\\
\ulcorner \mathrm{S}(\tau)\urcorner &= \boldsymbol{S}(\ulcorner\tau\urcorner)\\
\ulcorner \mathrm{P}(\tau)\urcorner &= \boldsymbol{P}(\ulcorner\tau\urcorner)\\
\ulcorner \mathrm{D}(\tau_1,\tau_2,\tau_3)\urcorner &= \boldsymbol{D}(\ulcorner\tau_1\urcorner,\ulcorner\tau_2\urcorner,\ulcorner\tau_3\urcorner)\\
\ulcorner f(\tau_1,\dots,\tau_n)\urcorner &= \ulcorner f\urcorner[\langle\ulcorner\tau_1\urcorner,\dots,\ulcorner\tau_k\urcorner,0\rangle] \bullet \ulcorner\tau_{k+1}\urcorner \bullet \cdots \bullet \ulcorner\tau_n\urcorner\\
&\quad \text{where } k \text{ is the maximal number such that}\\
&\quad \text{the terms } \tau_1,\dots,\tau_k \text{ are numerals}\\
\ulcorner \mathfrak{f}_n\urcorner &= \boldsymbol{f}_n\\
\ulcorner \lambda_n.\tau\urcorner &= \boldsymbol{\lambda}_n\boldsymbol{.}\ulcorner\tau\urcorner.
\end{aligned}
$$

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:

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

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:

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

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

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

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

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

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

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

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

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

4 Arithmetization of substitution function. The substitution function $\tau[\lambda_n.\sigma; \underline{\vec{x}}]$ is over recursive terms. Its arithmetization $t[e; rs]$ is a ternary function which takes the code t of the R-term $\tau[\mathfrak{f}_n; x_1, \dots, x_n]$ 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.

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

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

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

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

$$Pn(\ulcorner \underline{x} \urcorner) = \ulcorner \underline{x \dot{-} 1} \urcorner$$
$$Dn(\ulcorner \underline{x} \urcorner, t_2, t_3) = \mathrm{D}(x, t_2, t_3).$$

The functions are defined explicitly as a primitive recursive functions:

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

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

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

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

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

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

7 Arithmetization of reductions. The iteration of Rd defined by

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

is a primitive recursive function. Properties of Rd generalizes to

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

8 Codes of defined recursive function symbols. We claim that there is a binary primitive recursive predicate $Rf(n, e)$ satisfying

$$Rf(n, e) \text{ iff } e = \ulcorner \boldsymbol{\lambda}_n\boldsymbol{.}\, \tau \urcorner \text{ for some defined R-function symbol } \boldsymbol{\lambda}_n\boldsymbol{.}\, \tau.$$

For that we need some auxiliary functions and predicates.

The predicate $Nms(ts)$ holds if ts is a list of the codes of numerals. The predicate is defined by course of values recursion as a p.r. predicate:

$Nms(0)$
$Nms\,\langle t, ts \rangle \leftarrow Nm(t) \wedge Nms(ts).$

The ternary predicate $Tm(t, rs, n)$ satisfies for all $n \geq 1$ and for all R-terms $\rho_1, \ldots, \rho_k$ in the recursor $\mathfrak{f}_n$ and in the variables $x_1, \ldots, x_n$:

predicate $Tm(t, \langle \ulcorner\rho_1\urcorner, \ldots, \ulcorner\rho_k\urcorner, 0\rangle, n)$ holds iff there is a R-term τ in the recursor $\mathfrak{f}_n$ and variables $x_1, \ldots, x_n$ such that

$$\ulcorner\tau\urcorner = t \bullet \ulcorner\rho_1\urcorner \bullet \cdots \bullet \ulcorner\rho_k\urcorner.$$

The predicate is defined by course of values recursion on t with substitution in parameters as a primitive recursive predicate:

$Tm(\boldsymbol{x}_i, 0, n) \leftarrow 1 \le i \le n$
$Tm(\boldsymbol{0}, 0, n)$
$Tm(\boldsymbol{S}(t), 0, n) \leftarrow Tm(t, 0, n)$
$Tm(\boldsymbol{P}(t), 0, n) \leftarrow Tm(t, 0, n)$
$Tm(\boldsymbol{D}(t_1, t_2, t_3), 0, n) \leftarrow Tm(t_1, 0, n) \wedge Tm(t_2, 0, n) \wedge Tm(t_3, 0, n)$
$Tm(t_1 \bullet t_2, rs, n) \leftarrow$
$\quad Tm(t_1, \langle t_2, rs\rangle, n) \wedge Tm(t_2, 0, n) \wedge \exists e \exists ts\, t_1 = e[ts] \wedge \neg Nm(t_2)$
$Tm(t_1 \bullet t_2, rs, n) \leftarrow$
$\quad Tm(t_1, \langle t_2, rs\rangle, n) \wedge Tm(t_2, 0, n) \wedge \neg\exists e \exists ts\, t_1 = e[ts]$
$Tm(\boldsymbol{f}_m[ts], rs, n) \leftarrow m \ge 1 \wedge m = n \wedge Nms(ts) \wedge L(ts) + L(rs) = m$
$Tm((\boldsymbol{\lambda}_m\boldsymbol{.}\, t)[ts], rs, n) \leftarrow$
$\quad m \ge 1 \wedge Nms(ts) \wedge L(ts) + L(rs) = m \wedge Tm(t, 0, m)$

The predicate $Rf(n, e)$ holding of the codes of n-ary defined recursive function symbols is defined explicitly as a primitive recursive predicate:

$$Rf(n, e) \leftrightarrow n \ge 1 \wedge \exists t \le e(e = \boldsymbol{\lambda}_n\boldsymbol{.}\, t \wedge \wedge Tm(t, 0, n)).$$

9 Auxiliary functions and predicates. The function $Ar(e)$ takes the code e of a R-function symbol f and yields the arity of f, i.e. we have

$$Ar(\ulcorner\mathfrak{f}_n\urcorner) = Ar(\ulcorner\lambda_n.\tau\urcorner) = Ar(\ulcorner g_i^n\urcorner) = n.$$

The function is defined explicitly as a primitive recursive function:

$Ar(\boldsymbol{f}_n) = n$
$Ar(\boldsymbol{\lambda}_n\boldsymbol{.}\, t) = n.$

The ternary *iteration contraction* function $t \mathbin{\underline{\bullet}}_n rs$ satisfying

$$t \mathbin{\underline{\bullet}}_n \langle r_1, \ldots, r_n\rangle = t \mathbin{\underline{\bullet}} r_1 \mathbin{\underline{\bullet}} \cdots \mathbin{\underline{\bullet}} r_n$$

is defined by course of values recursion regular in rs with substitution in parameter as a primitive recursive function:

$t \mathbin{\underline{\bullet}}_1 r = t \mathbin{\underline{\bullet}} r$
$t \mathbin{\underline{\bullet}}_{n+2} \langle r, rs\rangle = t \mathbin{\underline{\bullet}} r \mathbin{\underline{\bullet}}_{n+1} rs$

The binary *application* function $e(ts)$ is such that the following holds

$$\ulcorner f(\tau_1, \ldots, \tau_n)\urcorner = \ulcorner f\urcorner(\langle \ulcorner\tau_1\urcorner, \ldots, \ulcorner\tau_n\urcorner\rangle)$$

for every R-term $f(\tau_1, \ldots, \tau_n)$. We define the application function explicitly as a primitive recursive function:

$$e(ts) = e[0] \mathbin{\underline{\bullet}}_{Ar(e)} ts.$$