The second variant of $\lambda_{\to}^{\mathrm{Cu}}$ is the *de Bruijn* version of $\lambda_{\to}^{\mathbb{A}}$, denoted by $\lambda_{\to\mathrm{dB}}^{\mathbb{A}}$ or $\lambda_{\to}^{\mathrm{dB}}$. Now only bound variables get ornamented with types, but only at the binding stage. The examples (1), (2) now become

$$\begin{array}{rll} & \vdash_{\lambda_{\to}}^{\mathrm{dB}} \ (\lambda x : A.x) : A{\to}A & (1_{\mathrm{dB}}) \\ y{:}A & \vdash_{\lambda_{\to}}^{\mathrm{dB}} \ (\lambda x : (A{\to}B).xy) : (A{\to}B){\to}A{\to}B & (2_{\mathrm{dB}}) \end{array}$$

The reasons to have these variants will be explained later in Section 1.4. In the meantime we will work intuitively.

1.1.20. Notation. Terms like $(\lambda fx.f(fx)) \in \Lambda^{\varnothing}(1{\to}o{\to}o)$ will often be written

$$\lambda f^1 x^0.f(fx)$$

to indicate the types of the bound variables. We will come back to this notational issue in section 1.4.

1.2. Normal inhabitants

In this section we will give an algorithm that enumerates the set of closed terms in normal form of a given type $A \in \mathbb{T}$. Since we will prove in the next chapter that all typable terms do have a nf and that reduction preserves typing, we thus have an enumeration of essentially all closed terms of that given type. We do need to distinguish various kinds of nf's.

1.2.1. Definition. Let $A = A_1{\to}\dots A_n{\to}\alpha$ and suppose $\Gamma \vdash M : A$.

(i) Then M is in long-nf, notation lnf, if $M \equiv \lambda x_1^{A_1} \dots x_n^{A_n}.xM_1 \dots M_n$ and each M_i is in lnf. By induction on the depth of the type of the closure of M one sees that this definition is well-founded.

(ii) M has a lnf if $M =_{\beta\eta} N$ and N is a lnf.

In Exercise 1.5.16 it is proved that if M has a $\boldsymbol{\beta}$-nf, which according to Theorem 2.2.4 is always the case, then it also has a unique lnf and will be its unique $\boldsymbol{\beta\eta}^{-1}$ nf. Here η^{-1} is the notion of reduction that is the converse of η.

1.2.2. Examples. (i) Note that $\lambda f^1.f =_{\beta\eta} \lambda f^1 \lambda x^o.fx$ and that $\lambda f^1.f$ is a $\boldsymbol{\beta\eta}$-nf but not a lnf.

(ii) $\lambda f^1 \lambda x^o.fx$ is a lnf, but not a $\boldsymbol{\beta\eta}$-nf.

(iii) $\lambda x{:}o.x$ is both in $\boldsymbol{\beta\eta}$-nf and lnf.

(iv) The $\boldsymbol{\beta}$-nf $\lambda F{:}2_2 \lambda f{:}1.Ff(\lambda x{:}o.fx)$ is neither in $\boldsymbol{\beta\eta}$-nf nor lnf.

(v) A variable of atomic type α is a lnf, but of type $A{\to}B$ not.

(vi) A variable $f : 1{\to}1$ has as lnf $\lambda g^1 \lambda x^o.f(\lambda y^o.gy)x$.

1.2.3. Proposition. *Every $\boldsymbol{\beta}$-nf M has a lnf M^ℓ such that $M^\ell \twoheadrightarrow_{\boldsymbol{\eta}} M$.*

Proof. Define M^ℓ by induction on the depth of the type of the closure of M as follows.

$$M^\ell \equiv (\lambda\vec{x}.yM_1\dots M_n)^\ell = \lambda\vec{x}\vec{z}.yM_1^\ell\dots M_n^\ell\vec{z}^\ell.$$

Then M^ℓ does the job. ■

Now we will define a 2-level grammar for obtaining the collection of all lnf's of a given type A.

1.2.4. Definition. Let $N = \{L(A;\Gamma) \mid A\in\mathbb{T}_\mathbb{A}; \Gamma$ a context of $\lambda_\to\}$. Let Σ be the alphabet of the terms of the $\lambda_\to^{\text{Ch}}$. Define the following two-level grammar, see van Wijngaarden et al. [1976], as a notion of reduction over words over $N\cup\Sigma$. The elements of N are the non-terminals (unlike in a context-free language there are now infinitely many of them).

$$\begin{array}{rcll} L(\alpha;\Gamma) & \Rightarrow & xL(B_1;\Gamma)\dots L(B_n;\Gamma), & \text{if } (x{:}\vec{B}\to\alpha)\in\Gamma;\\ L(A\to B;\Gamma) & \Rightarrow & \lambda x^A.L(B;\Gamma,x{:}A). & \end{array}$$

Typical productions of this grammar are the following.

$$\begin{array}{rcl} L(3;\emptyset) & \Rightarrow & \lambda F^2.L(o;F^2)\\ & \Rightarrow & \lambda F^2.FL(1;F^2)\\ & \Rightarrow & \lambda F^2.F(\lambda x^o.L(o;F^2,x^o))\\ & \Rightarrow & \lambda F^2.F(\lambda x^o.x). \end{array}$$

But one has also

$$\begin{array}{rcl} L(o;F^2,x^o) & \Rightarrow & FL(1;F^2,x^o)\\ & \Rightarrow & F(\lambda x_1^o.L(o;F^2,x^o,x_1^o))\\ & \Rightarrow & F(\lambda x_1^o.x_1). \end{array}$$

Hence ($\Rightarrow\!\!\!\!\Rightarrow$ denotes the transitive reflexive closure of $\Rightarrow$)

$$L(3;\emptyset) \Rightarrow\!\!\!\!\Rightarrow \lambda F^2.F(\lambda x^o.F(\lambda x_1^o.x_1)).$$

In fact, $L(3;\emptyset)$ reduces to all possible closed lnf's of type 3. Like in abstract syntax we do not produce parentheses from the $L(A;\Gamma)$, but write them when needed.

1.2.5. Proposition. *Let Γ, M, A be given. Then*

$$L(A,\Gamma) \Rightarrow\!\!\!\!\Rightarrow M \iff \Gamma\vdash M : A \;\&\; M \textit{ is in lnf.}$$

Now we will modify the 2-level grammar and the inhabitation machines in order to produce all $\boldsymbol{\beta}$-nf's.

1.2.6. Definition. The 2-level grammar N is defined as follows.

$$\begin{array}{rcll} N(A;\Gamma) & \Rightarrow & xN(B_1;\Gamma)\ldots N(B_n;\Gamma), & \text{if } (x{:}\vec{B}{\to}A)\in\Gamma; \\ N(A{\to}B;\Gamma) & \Rightarrow & \lambda x^A.N(B;\Gamma,x{:}A). & \end{array}$$

Now the $\boldsymbol{\beta}$-nf's are being produced. As an example we make the following production. Remember that $1 = o{\to}o$.

$$\begin{array}{rcl} L(1{\to}o{\to}o;\emptyset) & \Rightarrow & \lambda f^1.L(o{\to}o; f{:}o{\to}o) \\ & \Rightarrow & \lambda f^1.f. \end{array}$$

1.2.7. Proposition. *Let Γ, M, A be given. Then*

$$N(A,\Gamma) \Rightarrow\!\!\!\Rightarrow M \iff \Gamma\vdash M : A \;\&\; M \textit{ is in } \boldsymbol{\beta}\textit{-nf.}$$

Inhabitation machines

Inspired by this proposition one can introduce for each type A a machine M_A producing the set of closed terms of that type. If one is interested in terms containing variables $x_1^{A_1},\ldots,x_n^{A_n}$, then one can also find these terms by considering the machine for the type $A_1{\to}\ldots{\to}A_n{\to}A$ and look at the subproduction at node A.

1.2.8. Examples. (i) $A = o{\to}o{\to}o$. Then M_A is

$$\begin{tikzcd}
\boxed{o{\to}o{\to}o} \arrow[r, "\lambda x^o\lambda y^o"] & \boxed{o} \arrow[r] \arrow[d] & x \\
 & y &
\end{tikzcd}$$

This shows that the type 1_2 has two closed inhabitants: $\lambda xy.x$ and $\lambda xy.y$. We see that the two arrows leaving $\boxed{o}$ represent a choice.

(ii) $A = \alpha{\to}((o{\to}\boldsymbol{\beta}){\to}\alpha){\to}\boldsymbol{\beta}{\to}\alpha$. Then M_A is

$$\begin{tikzcd}
\boxed{\alpha{\to}((o{\to}\boldsymbol{\beta}){\to}\alpha){\to}\boldsymbol{\beta}{\to}\alpha} \arrow[d, "\lambda a^\alpha\lambda f^{(o\to\boldsymbol{\beta})\to\alpha}\lambda b^{\boldsymbol{\beta}}"] & & \\
\boxed{\alpha} \arrow[rr] \arrow[d, "f"] & & a \\
\boxed{o{\to}\boldsymbol{\beta}} \arrow[r, "\lambda y^o"] & \boxed{\boldsymbol{\beta}} \arrow[r] & b
\end{tikzcd}$$

Again there are only two inhabitants, but now the production of them is rather different: $\lambda afb.a$ and $\lambda afb.f(\lambda x^o.b)$.

(iii) $A = ((\alpha\to\boldsymbol{\beta})\to\alpha)\to\alpha$. Then M_A is

$$\begin{tikzcd}
\boxed{((\alpha\to\boldsymbol{\beta})\to\alpha)\to\alpha} \arrow[d, "\lambda F^{(\alpha\to\boldsymbol{\beta})\to\alpha}"] & & \\
\boxed{\alpha} \arrow[r, "F"] & \boxed{\alpha\to\boldsymbol{\beta}} \arrow[r, "\lambda x^{\alpha}"] & \boxed{\boldsymbol{\beta}}
\end{tikzcd}$$

This type, corresponding to Peirce's law, does not have any inhabitants.
(iv) $A = 1\to o\to o$. Then M_A is

$$\begin{tikzcd}
\boxed{1\to o\to o} \arrow[d, "\lambda f^1\lambda x^o"] & \\
\boxed{o} \arrow[loop left, "f"] \arrow[r] & x
\end{tikzcd}$$

This is the type Nat having the Church's numerals $\lambda f^1 x^o.f^n x$ as inhabitants.
(v) $A = 1\to 1\to o\to o$. Then M_A is

$$\begin{tikzcd}
\boxed{1\to 1\to o\to o} \arrow[d, "\lambda f^1\lambda g^1\lambda x^o"] \\
\boxed{o} \arrow[loop left, "f"] \arrow[loop right, "g"] \arrow[d] \\
x
\end{tikzcd}$$

Inhabitants of this type represent words over the alphabet $\Sigma = \{f, g\}$, for example

$$\lambda f^1 g^1 x^o.fgffgfggx,$$

where we have to insert parentheses associating to the right.
(vi) $A = (\alpha\to\boldsymbol{\beta}\to\gamma)\to\boldsymbol{\beta}\to\alpha\to\gamma$. Then M_A is

$$\begin{tikzcd}
 & & \boxed{(\alpha\to\boldsymbol{\beta}\to\gamma)\to\boldsymbol{\beta}\to\alpha\to\gamma} \arrow[d, "\lambda f^{\alpha\to\boldsymbol{\beta}\to\gamma}\lambda b^{\boldsymbol{\beta}}\lambda a^{\alpha}"] & & \\
 & & \boxed{\gamma} \arrow[d] & & \\
a & \boxed{\alpha} \arrow[l] & f \arrow[l] \arrow[r] & \boxed{\beta} \arrow[r] & b
\end{tikzcd}$$

giving as term $\lambda f^{\alpha\to\boldsymbol{\beta}\to\gamma}\lambda b^{\boldsymbol{\beta}}\lambda a^{\alpha}.fab$. Note the way an interpretation should be given to paths going through f: the outgoing arcs (to $\boxed{\alpha}$ and $\boxed{\boldsymbol{\beta}}$) should be completed both separately in order to give f its two arguments.

(vii) $A = 3$. Then M_A is

$$\begin{tikzcd}
\boxed{3} \arrow[d, "\lambda F^2"'] & \\
\boxed{o} \arrow[r, bend left, "F"] \arrow[d] & \boxed{1} \arrow[l, bend left, "\lambda x^o"] \\
x &
\end{tikzcd}$$

This type 3 has inhabitants having more and more binders:

$$\lambda F^2.F(\lambda x_0^o.F(\lambda x_1^o.F(\cdots(\lambda x_n^o.x_i)))).$$

The novel phenomenon that the binder λx^o may go round and round forces us to give new incarnations λx_0^o, $\lambda x_1^o, \ldots$ each time we do this (we need a counter to ensure freshness of the bound variables). The 'terminal' variable x can take the shape of any of the produced incarnations x_k. As almost all binders are dummy, we will see that this potential infinity of binding is rather innocent and the counter is not yet really needed here.

(viii) $A = 3{\to}o{\to}o$. Then M_A is

$$\begin{tikzcd}
\boxed{3{\to}o{\to}o} \arrow[d, "\lambda\Phi^3\lambda c^o"'] & \\
\boxed{o} \arrow[loop left, "f"] \arrow[r, bend left, "\Phi"] \arrow[d] & \boxed{2} \arrow[l, bend left, "\lambda f^1"] \\
c &
\end{tikzcd}$$

This type, called the *monster* M, does have a potential infinite amount of binding, having as terms e.g.

$$\lambda\Phi^3 c^o.\Phi\lambda f_1^1.f_1\Phi\lambda f_2^1.f_2f_1\Phi\ldots\lambda f_n^1.f_n\ldots f_2f_1c,$$

again with inserted parentheses associating to the right. Now a proper bookkeeping of incarnations (of f^1 in this case) becomes necessary, as the f going from $\boxed{o}$ to itself needs to be one that has already been incarnated.

(ix) $A = 1_2{\to}o{\to}o$. Then M_A is

$$\begin{tikzcd}
\boxed{1_2{\to}o{\to}o} \arrow[r, "\lambda p^{1_2}\lambda c^o"] & \boxed{o} \arrow[r] \arrow[d] & c \\
& p \arrow[u, bend left] \arrow[u, bend right] &
\end{tikzcd}$$

This is the type of binary trees, having as elements, e.g. $\lambda p^{1_2}c^o.c$ and $\lambda p^{1_2}c^o.pc(pcc)$. Again, as in example (vi) the outgoing arcs from p (to $\boxed{o}$) should be completed both separately in order to give p its two arguments.

(x) $A = 1_2\to2\to o$. Then M_A is

$$\begin{tikzcd}
 & \boxed{1} \arrow[d, bend left, "\lambda x^o"] & \\
\boxed{1_2\to2\to o} \arrow[r, "\lambda F^{1_2}\lambda G^2"] & \boxed{o} \arrow[u, bend left, "G"] \arrow[r] \arrow[d] & x \\
 & F \arrow[u, bend left] \arrow[u, bend right] &
\end{tikzcd}$$

This is the type L corresponding to untyped lambda terms. For example the untyped terms $\omega \equiv \lambda x.xx$ and $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ can be translated to $(\omega)^t \equiv \lambda F^{1_2}G^2.G(\lambda x^o.Fxx)$ and

$$\begin{array}{rcl}
(\Omega)^t & \equiv & \lambda F^{1_2}G^2.F(G(\lambda x^o.Fxx))(G(\lambda x^o.Fxx)) \\
 & =_\beta & \lambda FG.F((\omega)^t FG)((\omega)^t FG) \\
 & =_\beta & (\omega)^t \cdot_L (\omega)^t,
\end{array}$$

where for $M, N \in L$ one defines $M \cdot_L N = \lambda FG.F(MFG)(NFG)$. All features of producing terms inhabiting types (bookkeeping bound variables, multiple paths) are present here.

Following the 2-level grammar N one can make inhabitation machines for $\boldsymbol{\beta}$-nf $M_A^{\boldsymbol{\beta}}$.

1.2.9. Example. We show how the production machine for $\boldsymbol{\beta}$-nf's differs from the one for lnf's. Let $A = 1\to o\to o$. Then $\lambda f^1.f$ is the (unique) $\boldsymbol{\beta}$-nf of type A that is not a lnf. It will come out from the following machine $M_A^{\boldsymbol{\beta}}$.

$$\begin{tikzcd}
\boxed{1\to o\to o} \arrow[d, "\lambda f^1"] & \\
\boxed{o\to o} \arrow[r] \arrow[d, "\lambda x^o"] & f \\
\boxed{o} \arrow[loop left, "f"] \arrow[r] & x
\end{tikzcd}$$

So in order to obtain the $\boldsymbol{\beta}$-nf's, one has to allow output at types that are not atomic.

1.3. Representing data types

In this section it will be shown that first order algebraic data types can be represented in $\lambda_\to^o$. We start with several examples: Booleans, the natural numbers, the free monoid over n generators (words over a finite alphabet with n elements) and trees with at the leafs labels from a type A. The following definitions depend on a given type A. So in fact $\mathsf{Bool} = \mathsf{Bool}_A$ etcetera. Often one takes $A = o$.