1C. Normal inhabitants

In this section we will give an algorithm that enumerates the set of closed inhabitants in $\boldsymbol{\beta}$-nf 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. The algorithm will be used by concluding that a certain type A is uninhabited or more generally that a certain class of terms exhausts all inhabitants of A.

Because the various versions of $\boldsymbol{\lambda}_{\to}^{\mathbb{A}}$ are equivalent as to inhabitation of closed $\boldsymbol{\beta}$-nfs, we flexibly jump between the set

$$\{M \in \Lambda_{\to}^{\mathrm{Ch}}(A) \mid M \text{ closed and in } \boldsymbol{\beta}\text{-nf}\}$$

and

$$\{M \in \Lambda \mid M \text{ closed, in } \boldsymbol{\beta}\text{-nf, and } \vdash_{\boldsymbol{\lambda}_{\to}}^{\mathrm{Cu}} M : A\},$$

thereby we often write a Curry context $\{x_1{:}A_1, \cdots, x_n{:}A_n\}$ as $\{x_1^{A_1}, \cdots, x_n^{A_n}\}$ and a Church term $\lambda x^0.x^0$ as $\lambda x^0.x$, an intermediate form between the Church and the de Bruijn versions.

We do need to distinguish various kinds of nfs.

1C.1. Definition. Let $A = A_1 \to \cdots A_n \to \alpha$ and suppose $M \in \Lambda_{\to}^{\mathrm{Ch}}(A)$.

(i) Then M is in *long-nf*, notation *lnf*, if $M \equiv \lambda x_1^{A_1} \cdots x_n^{A_n}.xM_1 \cdots M_n$ 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 $M =_{\boldsymbol{\beta\eta}} N$ and N is a lnf.

In Exercise 1E.14 it is proved that if M has a $\boldsymbol{\beta}$-nf, which according to Theorem 2B.4 is always the case, then it also has a unique lnf and this will be its unique $\boldsymbol{\beta\eta}^{-1}$ nf. Here $\boldsymbol{\eta}^{-1}$ is the notion of reduction that is the converse of $\boldsymbol{\eta}$.

1C.2. Examples. (i) $\lambda x^0.x$ is both in $\boldsymbol{\beta\eta}$-nf and lnf.

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

(iii) $\lambda f^1 x^0.fx$ is a lnf but not a $\boldsymbol{\beta\eta}$-nf; its $\boldsymbol{\beta\eta}$-nf is $\lambda f^1.f$.

(iv) The $\boldsymbol{\beta}$-nf $\lambda F_2^2 \lambda f^1.Ff(\lambda x^0.fx)$ 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 $\lambda g^1 x^0.f(\lambda y^0.gy)x =_{\boldsymbol{\eta}} f^{1\to 1}$.

1C.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.

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

where $\vec{z}$ is the longest vector that preserves the type. Then M^ℓ does the job. ∎

We will define a *2-level grammar*, see van Wijngaarden [1981], for obtaining all closed inhabitants in lnf of a given type A. We do this via the system $\boldsymbol{\lambda}_{\to}^{\mathrm{Cu}}$.

1C.4. Definition. Let $\mathcal{L} = \{L(A;\Gamma) \mid A \in \mathbb{T}^{\mathbb{A}}; \Gamma \text{ a context of } \boldsymbol{\lambda}_{\to}^{\mathrm{Cu}}\}$. Let Σ be the alphabet of the untyped lambda terms. Define the following two-level grammar as a notion of reduction over words over $\mathcal{L} \cup \Sigma$. The elements of $\mathcal{L}$ are the non-terminals (unlike in a context-free language there are now infinitely many of them) of the form $L(A;\Gamma)$.

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

Typical productions of this grammar are the following.

$$\begin{aligned} L(3;\emptyset) &\implies \lambda F^2.L(0;F^2)\\ &\implies \lambda F^2.FL(1;F^2)\\ &\implies \lambda F^2.F(\lambda x^0.L(0;F^2,x^0))\\ &\implies \lambda F^2.F(\lambda x^0.x). \end{aligned}$$

But one has also

$$\begin{aligned} L(0;F^2,x^0) &\implies FL(1;F^2,x^0)\\ &\implies F(\lambda x_1^0.L(0;F^2,x^0,x_1^0))\\ &\implies F(\lambda x_1^0.x_1). \end{aligned}$$

Hence ($\Longrightarrow\!\!\!\!\!\!\!\!\;\,\twoheadrightarrow$ denotes the transitive reflexive closure of $\implies$)

$$L(3;\emptyset) \twoheadrightarrow\!\!\!\!\!\!\!\implies \lambda F^2.F(\lambda x^0.F(\lambda x_1^0.x_1)).$$

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

1C.5. Proposition. *Let Γ, M, A be given. Then*

$$L(A;\Gamma) \twoheadrightarrow\!\!\!\!\!\!\!\implies M \quad\Leftrightarrow\quad \Gamma\vdash M:A\ \&\ M \text{ is in lnf.}$$

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

1C.6. Definition. The 2-level grammar N is defined as follows.

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

Now the $\boldsymbol{\beta}$-nfs are being produced. As an example we make the following production. Remember that $1=0\to0$.

$$\begin{aligned} N(1\to0\to0;\emptyset) &\implies \lambda f^1.N(0\to0;f^1)\\ &\implies \lambda f^1.f. \end{aligned}$$

1C.7. Proposition. *Let Γ, M, A be given. Then*

$$N(A,\Gamma) \twoheadrightarrow\!\!\!\!\!\!\!\implies M \quad\Leftrightarrow\quad \Gamma\vdash M:A\ \&\ M \text{ is in } \boldsymbol{\beta}\text{-nf.}$$

Inhabitation machines

Inspired by this proposition one can introduce for each type A a machine M_A producing the set of closed lnfs of that type. If one is interested in terms containing free variables $x_1^{A_1}, \cdots, x_n^{A_n}$, then one can also find these terms by considering the machine for the type $A_1 \to \cdots \to A_n \to A$ and looking at the sub-production at node A. This means that a normal inhabitant M_A of type A can be found as a closed inhabitant $\lambda \vec{x}.M_A$ of type $A_1 \to \cdots \to A_n \to A$.

1C.8. EXAMPLES. (i) $A = 0\to0\to0$. Then M_A is

$$
\begin{tikzcd}
\boxed{0\to0\to0} \arrow[r, "\lambda x^0\lambda y^0"] & \boxed{0} \arrow[r] \arrow[d] & x \\
& y &
\end{tikzcd}
$$

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 $\boxed{0}$ represent a choice.

(ii) $A = \alpha\to((0\to\beta)\to\alpha)\to\beta\to\alpha$. Then M_A is

$$
\begin{tikzcd}
\boxed{\alpha\to((0\to\beta)\to\alpha)\to\beta\to\alpha} \arrow[d, "\lambda a^\alpha \lambda f^{(0\to\beta)\to\alpha} \lambda b^\beta"] & & \\
\boxed{\alpha} \arrow[rr] \arrow[d, "f"] & & a \\
\boxed{0\to\beta} \arrow[r, "\lambda x^0"] & \boxed{\beta} \arrow[r] & b
\end{tikzcd}
$$

Again there are only two inhabitants, but now the production of them is rather different: $\lambda afb.a$ and $\lambda afb.f(\lambda x^0.b)$.

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

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

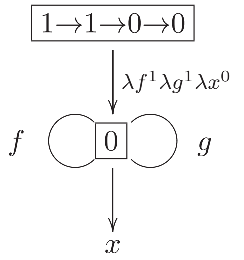
This type, corresponding to Peirce's law, does not have any inhabitants.

(iv) $A = 1\to0\to0$. Then M_A is

$$
\begin{tikzcd}
\boxed{1\to0\to0} \arrow[d, "\lambda f^1\lambda x^0"] & \\
\boxed{0} \arrow[loop left, "f"] \arrow[r] & x
\end{tikzcd}
$$

This is the type Nat having the Church's numerals $\lambda f^1x^0.f^nx$ as inhabitants.

(v) $A = 1{\to}1{\to}0{\to}0$. Then M_A is

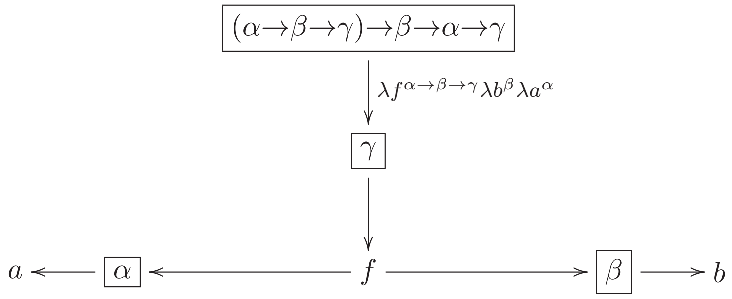


Inhabitants of this type represent words over the alphabet $\Sigma = \{f, g\}$, for example

$$\lambda f^1 g^1 x^0 . fgffgfggx,$$

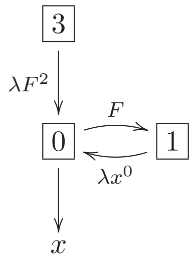
where we have to insert parentheses associating to the right.

(vi) $A = (\alpha{\to}\beta{\to}\gamma){\to}\beta{\to}\alpha{\to}\gamma$. Then M_A is



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

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



This type 3 has inhabitants having more and more binders:

$$\lambda F^2 . F(\lambda x_0^0 . F(\lambda x_1^0 . F(\cdots (\lambda x_n^0 . x_i)))).$$

The novel phenomenon that the binder λx^0 may go round and round forces us to give new incarnations λx_0^0, $\lambda x_1^0, \cdots$ 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) $A = 3{\to}0{\to}0$. Then M_A is

$$\begin{tikzcd} & \boxed{3{\to}0{\to}0} \arrow[d, "\lambda\Phi^3\lambda c^0"'] & \\ & \boxed{0} \arrow[loop left, "f"] \arrow[r, bend left, "\Phi"] \arrow[d] & \boxed{2} \arrow[l, bend left, "\lambda f^1"] \\ & c & \end{tikzcd}$$

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

$$\lambda\Phi^3 c^0.\Phi(\lambda f_1^1.f_1\Phi(\lambda f_2^1.f_2 f_1\Phi(\cdots(\lambda f_n^1.f_n\cdots f_2 f_1 c)..))),$$

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 $\boxed{0}$ to itself needs to be one that has already been incarnated.

(ix) $A = 1_2{\to}0{\to}0$. Then M_A is

$$\begin{tikzcd} \boxed{1_2{\to}0{\to}0} \arrow[r, "\lambda p^{1_2}\lambda c^0"] & \boxed{0} \arrow[r] \arrow[d] & c \\ & p \arrow[u, bend left] \arrow[u, bend right] & \end{tikzcd}$$

This is the type of binary trees, having as elements, e.g. $\lambda p^{1_2}c^0.c$ and $\lambda p^{1_2}c^0.pc(pcc)$. Again, as in example (vi) the outgoing arcs from p (to $\boxed{0}$) should be completed both separately in order to give p its two arguments.

(x) $A = 1_2{\to}2{\to}0$. Then M_A is

$$\begin{tikzcd} & \boxed{1} \arrow[d, bend left, "\lambda x^0"] & \\ \boxed{1_2{\to}2{\to}0} \arrow[r, "\lambda F^{1_2}\lambda G^2"] & \boxed{0} \arrow[r] \arrow[u, bend left, "G"] \arrow[d] & x \\ & F \arrow[u, bend left] \arrow[u, bend right] & \end{tikzcd}$$

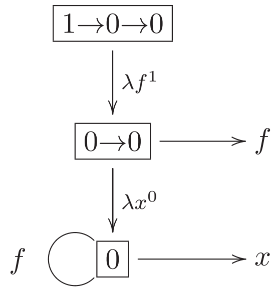
The inhabitants of this type, which we call L, can be thought of as codes for untyped lambda terms. For example the untyped terms $\omega \equiv \lambda x.xx$ and $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ can be translated to $(\omega)^t \equiv \lambda F^{1_2}G^2.G(\lambda x^0.Fxx)$ and

$$\begin{array}{rcl} (\Omega)^t & \equiv & \lambda F^{1_2}G^2.F(G(\lambda x^0.Fxx))(G(\lambda x^0.Fxx)) \\ & =_{\boldsymbol{\beta}} & \lambda FG.F((\omega)^t FG)((\omega)^t FG) \\ & =_{\boldsymbol{\beta}} & (\omega)^t \cdot_L (\omega)^t, \end{array}$$

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 in this example.

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

1C.9. Example. We show how the production machine for $\boldsymbol{\beta}$-nfs differs from the one for lnfs. Let $A = 1{\to}0{\to}0$. 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 $M_A^{\boldsymbol{\beta}}$.



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

1D. Representing data types

In this section it will be shown that first order algebraic data types can be represented in $\boldsymbol{\lambda}_{\to}^0$. This means that an algebra $\mathcal{A}$ can be embedded into the set of closed terms in $\boldsymbol{\beta}$-nf in $\Lambda_{\to}^{\mathrm{Cu}}(A)$. That we work with the Curry version is as usual not essential.

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 $A = 0$.

Booleans

1D.1. Definition. Define $\mathsf{Bool} \equiv \mathsf{Bool}_A$

$$\mathsf{Bool} \triangleq A{\to}A{\to}A;$$
$$\mathsf{true} \triangleq \lambda xy.x;$$
$$\mathsf{false} \triangleq \lambda xy.y.$$

Then $\mathsf{true} \in \Lambda_{\to}^{\emptyset}(\mathsf{Bool})$ and $\mathsf{false} \in \Lambda_{\to}^{\emptyset}(\mathsf{Bool})$.

1D.2. Proposition. *There are terms* not, and, or, imp, iff *with the expected behavior on Booleans. For example* $\mathsf{not} \in \Lambda_{\to}^{\emptyset}(\mathsf{Bool}{\to}\mathsf{Bool})$ *and*

$$\mathsf{not}\ \mathsf{true} =_{\boldsymbol{\beta}} \mathsf{false},$$
$$\mathsf{not}\ \mathsf{false} =_{\boldsymbol{\beta}} \mathsf{true}.$$

Proof. Take $\mathsf{not} \triangleq \lambda axy.ayx$ and $\mathsf{or} \triangleq \lambda abxy.ax(bxy)$. From these two operations the other Boolean functions can be defined. For example, implication can be represented by

$$\mathsf{imp} \triangleq \lambda ab.\mathsf{or}(\mathsf{not}\ a)b.$$

A shorter representation is $\lambda abxy.a(bxy)x$, the normal form of imp. ■