Arabic Numbers in Homotopy Type Theory

Lê Nguyên Hoang

April 21, 2014

This short document is a complement to my Science4All article on univalence. We will define formally the type N of numbers written in the Arabic number system and the addition add of these numbers.

The goal of this document is primarily to get the reader and myself (but mostly myself) familiar with homotopy type-theoretical constructions. It is definitely sketchy and has no aim at being perfectly rigorous nor pedagogical. My original goal included proving the isomorphism $(\mathbb{N},+) \simeq (\mathsf{N},\mathsf{add})$, but the mere constructions are so lengthy that I have given up.

1 Digits and Lists

We first define the type Digit, with the ten constructors for each digit:

- $0,1,2,3,4,5,6,7,8,9 : \mathsf{Digit}$.

The obvious induction principle of that type requires values $f(0) : B(0)$, ..., $f(9) : B(9)$ to determine an outgoing function $f : \prod_{d:\mathsf{Digit}} B(d)$. We use this induction principle to define $\mathsf{unitOfSum} : \mathsf{Digit} \to \mathsf{Digit} \to \mathsf{Digit}$ which computes the unit of the sum, and $\mathsf{carryOver} : \mathsf{Digit} \to \mathsf{Digit} \to \mathsf{Digit}$ the tens of the sum.

Lemma 1. *We have* $\mathsf{unitOfSum}(d,0) = \mathsf{unitOfSum}(0,d) = d$ *and* $\mathsf{carryOver}(d,0) = \mathsf{carryOver}(0,d) = 0$ *for all* $d : \mathsf{Digit}$. *Also, both functions are commutative.*

Proof. By construction, and using refl_d and refl_0. □

We then define the type $\mathsf{List} : \mathcal{U} \to \mathcal{U}$, $A \mapsto \mathsf{List}(A)$ with constructors

- $\emptyset : \mathsf{List}(A)$.
- $\mathsf{addLast} : \mathsf{List}(A) \to A \to \mathsf{List}(A)$.

To construct an outgoing function $f : \prod_{x:\mathsf{List}(A)} B(x)$, the induction principle of $\mathsf{List}(A)$ requires $f(\emptyset) : B(\emptyset)$ and $f_{\mathsf{addLast}} : \prod_{x:\mathsf{List}(A)} \prod_{a:A} B(x) \to B(\mathsf{addLast}(a,x))$. When $B(x)$ does not depend on $x : \mathsf{List}(A)$, we obtain the recursion principle which constructs $f : \mathsf{List}(A) \to B$ from $f(\emptyset) : B$ and $f_{\mathsf{addLast}} : B \to A \to B$.

We use this recursion principle to define $\mathsf{addFirst} : A \to \mathsf{List}(A) \to \mathsf{List}(A)$ by $\mathsf{addFirst}(a,\emptyset) :\equiv \mathsf{addLast}(a,\emptyset)$ and $\mathsf{addFirst}(a,\mathsf{addLast}(x,b)) :\equiv \mathsf{addLast}(\mathsf{addFirst}(a,x),b)$.

Arabic numbers are represented by lists of digit. Thus, we are here interested in the type $\mathsf{List}(\mathsf{Digit})$. In this case, the second constructor constructs a new digit list from a digit list $t : \mathsf{List}(\mathsf{Digit})$ representing the tens and digit $u : \mathsf{Digit}$ representing the unit.

2 Addition of Digit Lists

To define the addition $\mathsf{addDigitList} : \mathsf{List}(\mathsf{Digit}) \to \mathsf{List}(\mathsf{Digit}) \to \mathsf{List}(\mathsf{Digit})$ of digit lists, we first define the addition with carry over $h : \mathsf{List}(\mathsf{Digit}) \to \mathsf{List}(\mathsf{Digit}) \to \mathsf{Digit} \to \mathsf{List}(\mathsf{Digit})$, where the third digit input is the carry over we shall denote c.

- $h(\emptyset, \emptyset, c) :\equiv \mathsf{addLast}(\emptyset, c)$.
- $h(\mathsf{addLast}(t, u), \emptyset, c) :\equiv \mathsf{addLast}(h(t, \emptyset, \mathsf{carryOver}(u, c)), \mathsf{unitOfSum}(u, c))$.
- $h(\emptyset, \mathsf{addLast}(t, u), c) :\equiv \mathsf{addLast}(h(\emptyset, t, \mathsf{carryOver}(u, c)), \mathsf{unitOfSum}(u, c))$.
- Finally, the last case is the computation of $h(\mathsf{addLast}(t_1, u_1), \mathsf{addLast}(t_2, u_2), c)$, which is slightly trickier. We first determine the unit of the addition $U(u_1, u_2, c) :\equiv \mathsf{unitOfSum}(\mathsf{unitOfSum}(u_1, u_2), c)$. Then, we compute the carry over $C(u_1, u_2, c) :\equiv \mathsf{unitOfSum}(\mathsf{carryOver}(u_1, u_2), \mathsf{carryOver}(\mathsf{unitOfSum}(u_1, u_2), c))$. Finally, we combine it all, yielding

$$h(\mathsf{addLast}(t_1, u_1), \mathsf{addLast}(t_2, u_2), c) :\equiv \mathsf{addLast}(h(t_1, t_2, C(u_1, u_2, c)), U(u_1, u_2, c)). \tag{1}$$

Lemma 2. *C and U are commutative, i.e. the order of the inputs does not matter.*

Proof. By explicit induction. □

Lemma 3. *$C(0, d, c) = C(d, 0, c) = \mathsf{carryOver}(d, c)$ and $U(0, d, c) = U(d, 0, c) = \mathsf{unitOfSum}(d, c)$.*

Proof. Using Lemma 1 □

Lemma 4. *For any $l_1, l_2 : \mathsf{List}(\mathsf{Digit})$ and any $d : \mathsf{Digit}$, we have $h(l_1, l_2, d) = h(l_2, l_1, d)$.*

Proof. The proof boils down to the construction of a function

$$f : \prod_{d:\mathsf{Digit}} \prod_{l_1:\mathsf{List}(\mathsf{Digit})} \prod_{l_2:\mathsf{List}(\mathsf{Digit})} h(l_1, l_2, d) = h(l_2, l_1, d). \tag{2}$$

We do it by induction on l_1 and l_2, accordingly to the four bullet points defining h. The first bullet point is straightforward, using the immediate proof $\mathsf{refl}_{\mathsf{addLast}(\emptyset, d)}$. The two following bullet points are not much harder, using a proof of $h(t, \emptyset, c) = h(\emptyset, t, c)$ by induction.

Finally, in the last case, we need to use the commutativity of $\mathsf{unitOfSum}$ and of $\mathsf{carryOver}$ we proved in Lemma 1. This shows that u and c are equal in both constructions of $h(l_1, l_2, d)$ and $h(l_2, l_1, d)$. Finally, by action on path, and using the proof $h(t_1, t_2, c) = h(t_2, t_1, c)$ obtained by induction, we obtain a proof $h(l_1, l_2, c) = h(l_2, l_1, c)$. □

We then define $\mathsf{add}_{\mathsf{list}}(l_1, l_2) :\equiv h(l_1, l_2, 0)$, where $0 : \mathsf{Digit}$ is the zero digit.

Theorem 1. *The addition $\mathsf{add}_{\mathsf{list}}$ is commutative and $\emptyset$ is a neutral element.*

Proof. Let us start with the proof that $\emptyset$ is a neutral element. We need to construct a function

$$f : \prod_{l:\mathsf{List}(\mathsf{Digit})} (\mathsf{add}_{\mathsf{list}}(l, \emptyset) = l) \times (\mathsf{add}_{\mathsf{list}}(\emptyset, l) = l). \tag{3}$$

By induction, if $l \equiv \emptyset$, then $\mathsf{add}_{\mathsf{list}}(l, \emptyset) \equiv \emptyset \equiv l$, and, similarly $\mathsf{add}_{\mathsf{list}}(\emptyset, l) \equiv l$. So, we may provide the proof $f(\emptyset) :\equiv (\mathsf{refl}_\emptyset, \mathsf{refl}_\emptyset)$. Now, if $l \equiv \mathsf{addLast}(t, u)$, then

$$\begin{aligned} \mathsf{add}_{\mathsf{list}}(l, \emptyset) &\equiv \mathsf{addLast}(h(t, \emptyset, \mathsf{carryOver}(u, 0)), \mathsf{unitOfSum}(u, 0)) && \text{(induction)} && (4)\\ &\equiv \mathsf{addLast}(\mathsf{add}_{\mathsf{list}}(t, \emptyset), u) && \text{(Lemma 1)} && (5) \end{aligned}$$

Yet, by induction we may assume $\mathsf{pr}_1(f(t)) : \mathsf{add}_{\mathsf{list}}(t, \emptyset) = t$, hence we may construct a proof $f_1(l) :\equiv \mathsf{ap}_{\mathsf{add}_{\mathsf{list}}(-, \mathsf{u})}(\mathsf{pr}_1(f(t)))$ of $\mathsf{add}_{\mathsf{list}}(l, \emptyset) = l$. Similarly, we may construct $f_2(l) : \mathsf{add}_{\mathsf{list}}(\emptyset, l) = l$, hence obtaining $f(l) :\equiv (f_1(l), f_2(l))$. This concludes the construction of f, and, hence, the proof that $\emptyset$ is a neutral element.

Proving the commutativity of the addition $\mathsf{add}_{\mathsf{list}}$ corresponds to constructing a function

$$g : \prod_{l_1:\mathsf{List}(\mathsf{Digit})} \prod_{l_2:\mathsf{List}(\mathsf{Digit})} (\mathsf{add}_{\mathsf{list}}(l_1, l_2) = \mathsf{add}_{\mathsf{list}}(l_2, l_1)). \tag{6}$$

We do it by induction on l_1. If $l_1 \equiv \emptyset$, then f can be used to prove that both sides equal l_2, and they are henced equal. Formally, this corresponds to defining $g(\mathsf{emptyset}, l_2) :\equiv \mathsf{pr}_2(f(l_2)) \cdot \mathsf{pr}_1(f(l_2))^{-1}$. Then, we assume $l_1 \equiv \mathsf{addLast}(t, u)$ and that we have constructed $g(t, l_2)$ for all $l_2 : \mathsf{List}(\mathsf{Digit})$. Using Lemma 1, it is easy to see that we have proofs that the units and the carry overs of the additions $\mathsf{add}_{\mathsf{list}}(l_1, l_2)$ and $\mathsf{add}_{\mathsf{list}}(l_2, l_1)$ are equal. □

3 Arabic Numbers

Now, as we have all learned it, two strings of digits may represent the same number. For instance "$01 = 1$". We formalize that by the fact that digit lists still need to be interpreted into numbers. This leads us to define the type N of numbers in the Arabic number system by

- $\mathsf{n} : \mathsf{List}(\mathsf{Digit}) \to \mathsf{N}$.
- For all $l : \mathsf{List}(\mathsf{Digit})$, a path $p(l) : \mathsf{n}(l) = \mathsf{n}(\mathsf{addFirst}(0, l))$.

Lemma 5. *For any $l_1, l_2 : \mathsf{List}(\mathsf{Digit})$ and any $d : \mathsf{Digit}$, we have $\mathsf{n}(h(\mathsf{addFirst}(0, l_1), l_2, d)) = \mathsf{n}(h(l_1, l_2, d))$.*

Proof. We do it by induction on l_1 and l_2. We have four cases to verify, which correspond to the four bullet points of the definition of h.

First case. If $l_1 \equiv \emptyset$, we have $\mathsf{addFirst}(0, l_1) \equiv \mathsf{addLast}(\emptyset, 0)$. If $l_2 \equiv \emptyset$, using the second bullet point definition of h, we have

$$\begin{aligned} h(\mathsf{addFirst}(0, l_1), l_2, d) &\equiv \mathsf{addLast}(h(\emptyset, \emptyset, \mathsf{carryOver}(0, d)), \mathsf{unitOfSum}(0, d)) && (7)\\ &\equiv \mathsf{addLast}(\mathsf{addLast}(\emptyset, 0), \mathsf{addLast}(\emptyset, d)) && (8)\\ &\equiv \mathsf{addFirst}(0, \mathsf{addLast}(\emptyset, d)). && (9) \end{aligned}$$

This is the digit list "$0d$". Yet, $h(l_1, l_2, d) \equiv \mathsf{addLast}(\emptyset, d)$, which is "$d$". But the path constructor of N yields a path $p(\mathsf{addLast}(\emptyset, d)) : \mathsf{addLast}(\emptyset, d) = \mathsf{n}(\mathsf{addFirst}(0, \mathsf{addLast}(\emptyset, d)))$.

Second case. Now, if $l_1 \equiv \emptyset$ and $l_2 \equiv \mathsf{addLast}(t_2, u_2)$, then we may use a proof of that the lemma holds for $\emptyset$, t_2 and d. But then,

$$\begin{aligned} h(\mathsf{addFirst}(0, l_1), l_2, d) &\equiv h(\mathsf{addLast}(\emptyset, 0), \mathsf{addLast}(t_2, u_2), d) && (10)\\ &\equiv \mathsf{addLast}(h(\emptyset, t_2, C(0, u_2, d)), U(0, u_2, d)) && (11)\\ &= \mathsf{addLast}(h(\emptyset, t_2, \mathsf{carryOver}(u_2, d)), \mathsf{unitOfSum}(u_2, d)), && (12) \end{aligned}$$

by using Lemma 3. Yet, this last expression is precisely $h(l_1, l_2, d)$, which proves the second case.

Third case. If $l_1 \equiv \mathsf{addLast}(t_1, u_1)$ and $l_2 \equiv \emptyset$, then we may assume by induction that the lemma holds of t_1, $\emptyset$ and c. But then,

$$\begin{aligned} h(\mathsf{addFirst}(0, l_1), l_2, d) &\equiv \mathsf{addLast}(h(\mathsf{addFirst}(0, t_1), \emptyset, \mathsf{carryOver}(u_1, d)), \mathsf{unitOfSum}(u_1, d)) && (13)\\ &= \mathsf{addLast}(h(t_1, \emptyset, \mathsf{carryOver}(u_1, d)), \mathsf{unitOfSum}(u_1, d)), && (14) \end{aligned}$$

which is exactly the expression for l_1, l_2 and d, and proves the Lemma for this case too.

Fourth case. Finally, assume $l_1 \equiv \mathsf{addLast}(t_1, u_1)$ and $l_2 \equiv \mathsf{addLast}(t_2, u_2)$, and, by induction, that the lemma holds for t_1, t_2 and c. Then,

$$\begin{aligned} h(\mathsf{addFirst}(0, l_1), l_2, d) &\equiv \mathsf{addLast}(h(\mathsf{addFirst}(0, t_1), t_2, C(u_1, u_2, d)), U(u_1, u_2, d)) && (15)\\ &= \mathsf{addLast}(h(t_1, t_2, C(u_1, u_2, d)), U(u_1, u_2, d)), && (16) \end{aligned}$$

which is exactly the right expression to conclude the proof. □

To construct $f : \prod_{x:\mathsf{N}} B(x)$, the induction principle requires a function $f_{list} : \prod_{l:\mathsf{List}(\mathsf{Digit})} B(\mathsf{n}(l))$ and a path $f_{path} : \prod_{l:\mathsf{List}(\mathsf{Digit})} f(l) =^{B}_{p(l)} f(\mathsf{addFirst}(0, l))$. Let us apply it to define $\mathsf{add} : \mathsf{N} \to \mathsf{N} \to \mathsf{N}$:

- $\mathsf{add}(\mathsf{n}(l_1), \mathsf{n}(l_2)) :\equiv \mathsf{n}(\mathsf{addDigitList}(l_1, l_2))$.
- We now need to prove that we have the identities $\mathsf{add}(\mathsf{n}(l_1), \mathsf{n}(l_2)) = \mathsf{add}(\mathsf{n}(\mathsf{addFirst}(0, l_1)), \mathsf{n}(l_2))$ and $\mathsf{add}(\mathsf{n}(l_1), \mathsf{n}(l_2)) = \mathsf{add}(\mathsf{n}(l_1), \mathsf{n}(\mathsf{addFirst}(0, l_2)))$ for all $l_1, l_2 : \mathsf{List}(\mathsf{Digit})$. But this is given by Lemma 5, using $d \equiv 0$ and the commutativity of h proven by Lemma 4.

4 Isomorphism

We define $f : \mathbb{N} \to \mathsf{N}$ by $f(0) :\equiv \mathsf{addLast}(\emptyset, 0)$ and $f(\mathsf{succ}(n)) :\equiv \mathsf{add}(f(n), \mathsf{addLast}(\emptyset, 1))$.

Reciprocally, we need to define multiplication on $\mathbb{N}$. We do it by induction, with $0 \times m = 0$ and $\mathsf{succ}(n) \times m = (n \times m) + m$. Then, we define $\mathsf{digitToN} : \mathsf{Digit} \to \mathbb{N}$ by $\mathsf{digitToN}(0) :\equiv 0$, $\mathsf{digitToN}(1) :\equiv \mathsf{succ}(0)$, $\mathsf{digitToN}(2) :\equiv \mathsf{succ}(\mathsf{succ}(0))\ldots$ and so on until 9. Next, we define $\mathsf{digitListToN} : \mathsf{List}(\mathsf{Digit}) \to \mathbb{N}$ by induction by $\mathsf{digitListToN}(\emptyset) :\equiv 0$ and

$$\mathsf{digitListToN}(\mathsf{addLast}(t, u)) :\equiv \mathsf{digitToN}(u) + (\mathsf{digitListToN}(t) \times (\mathsf{succ}(\mathsf{digitToN}(9)))). \tag{17}$$

Now, to define $g : \mathsf{N} \to \mathbb{N}$, we first need to prove the following lemma:

Lemma 6. *For any $l : \mathsf{List}(\mathsf{Digit})$, we have $\mathsf{digitListToN}(\mathsf{addFirst}(0, l)) = \mathsf{digitListToN}(l)$.*

Proof. By induction on l. If $l \equiv \emptyset$, then

$$\mathsf{digitListToN}(\mathsf{addFirst}(0, l)) \equiv \mathsf{digitListToN}(\mathsf{addLast}(\emptyset, 0)) \equiv 0 + (0 \times 10) \equiv 0 \equiv \mathsf{digitListToN}(l). \tag{18}$$

Now, if $l \equiv \mathsf{addLast}(t, u)$, then we may assume by induction that the Lemma holds for t. But then, by action on path,

$$\begin{aligned} \mathsf{digitListToN}(\mathsf{addFirst}(0, l)) &\equiv \mathsf{digitListToN}(\mathsf{addLast}(\mathsf{addFirst}(0, t), u) && (19)\\ &= \mathsf{digitListToN}(\mathsf{addLast}(t, u)) && (20)\\ &\equiv \mathsf{digitListToN}(l), && (21) \end{aligned}$$

which concludes the proof. □

Let us $p_g(l)$ the proof we constructed. Then, we may finally define $g : \mathsf{N} \to \mathbb{N}$ by $g(\mathsf{n}(l)) :\equiv \mathsf{digitListToN}(l)$ and the proofs $p_g(l) : g(\mathsf{n}(l)) = g(\mathsf{n}(\mathsf{addFirst}(0, l)))$.

To prove the isomorphism of $(\mathbb{N}, +)$ and $(\mathsf{N}, \mathsf{add})$, all we have left to do is to give proofs of

- $g(f(n)) = n$ for all $n : \mathbb{N}$.
- $f(g(n)) = n$ for all $n : \mathsf{N}$.
- $f(n + m) = f(n) + f(m)$ for all $n, m : \mathbb{N}$.