# 15-819 Homotopy Type Theory
# Lecture Notes

Evan Cavallo and Stefan Muller

November 18 and 20, 2013

## 1 Reconsider Nat in simple types

As a warmup to discussing inductive types, we first review several equivalent presentations of the simple type $\mathsf{Nat}$ seen earlier in the course. The introduction forms for $\mathsf{Nat}$ are 0 and $\mathsf{succ}(M)$ for any $M : \mathsf{Nat}$. The elimination form is the recursor $\mathsf{rec}$.

### 1.1 Traditional Form

$$\frac{}{\Gamma \vdash 0 : \mathsf{Nat}}\ \mathsf{Nat}I_{z1} \qquad\qquad \frac{\Gamma \vdash M : \mathsf{Nat}}{\Gamma \vdash \mathsf{succ}(M) : \mathsf{Nat}}\ \mathsf{Nat}I_{s1}$$

$$\frac{\Gamma \vdash M : \mathsf{Nat} \qquad \Gamma \vdash M_0 : A \qquad \Gamma, x : A \vdash M_1 : A}{\Gamma \vdash \mathsf{rec}[A](M; M_0; x.M_1) : A}\ \mathsf{Nat}E_1$$

We include the motive $A$ in the recursor to motivate the dependently-typed presentation to come although it is not necessary in the simply-typed setting. The dynamic behavior of $\mathsf{rec}$ is defined by the following $\beta$ rules.

$$\mathsf{rec}[A](0; M_0; x.M_1) \equiv M_0$$

$$\mathsf{rec}[A](\mathsf{succ}(M); M_0; x.M_1) \equiv [\mathsf{rec}[A](M; M_0; x.M_1)/x]M_1$$

The recursor on 0 returns the base case $M_0$. On $\mathsf{succ}(M)$, it substitutes the recursive result on $M$ for $x$ in $M_1$. The $\eta$ rule states that any object that "behaves like" the recursor is definitionally equal to the recursor on the appropriate arguments.

$$\frac{[0/y]N \equiv M_0 \qquad \Gamma, z : \mathsf{Nat} \vdash [\mathsf{succ}(z)/y]N \equiv [[z/y]N/x]M_1 : A}{\Gamma, y : \mathsf{Nat} \vdash N \equiv \mathsf{rec}[A](y; M_0; x.M_1)}\ \eta$$

### 1.2 As elements of the exponential

The introduction forms of $\mathsf{Nat}$ may be treated as exponentials in the absence of a context.

$$\cdot \vdash 0 : 1 \to \mathsf{Nat} \qquad (\mathsf{Nat}I_{z3})$$

$$\cdot \vdash \mathsf{succ} : \mathsf{Nat} \to \mathsf{Nat} \qquad (\mathsf{Nat}I_{s3})$$

Note that the type $1 \to \mathsf{Nat}$ is equivalent to the type $\mathsf{Nat}$. There are two ways to present the elimination form in this format. The first moves the $\mathsf{Nat}$ on which the recursion is done to the argument position, implying that $\mathsf{rec}$ has exponential type.

$$\frac{\cdot \vdash M_0 : A \qquad x : A \vdash M_1 : A}{z : \mathsf{Nat} \vdash \mathsf{rec}[A](M_0; x.M_1)(z) : A}\ \mathsf{Nat}E_{3a}$$

This can be presented in a more direct way by omitting the argument.

$$\frac{\cdot \vdash M_0 : A \qquad x : A \vdash M_1 : A}{\cdot \vdash \mathsf{rec}[A](M_0; x.M_1) : \mathsf{Nat} \to A}\ \mathsf{Nat}E_{3b}$$

We can derive rules $\mathsf{Nat}I_{z1}$, $\mathsf{Nat}I_{s1}$ and $\mathsf{Nat}E_1$. For example,

$$\frac{\dfrac{}{\Gamma, M : \mathsf{Nat} \vdash M : \mathsf{Nat}} \qquad \dfrac{}{\Gamma, M : \mathsf{Nat} \vdash \mathsf{succ} : \mathsf{Nat} \to \mathsf{Nat}}}{\Gamma, M : \mathsf{Nat} \vdash \mathsf{succ}(M) : \mathsf{Nat}}\ \mathsf{Nat}I_{s3}$$

## 2 Nat-algebras

We now motivate the idea of $\mathsf{Nat}$-algebras, which are maps of the form $1 + A \to A$. From the name, one would expect there to be a $\mathsf{Nat}$-algebra where $A$ is $\mathsf{Nat}$. Indeed, there is.

$$z : 1 + \mathsf{Nat} \vdash \mathsf{case}(z; \_.0; x.\mathsf{succ}(x)) : \mathsf{Nat}$$

We can write $\mathsf{case}(z; \_.0; x.\mathsf{succ}(x))$ above as $\{\_.0; x.\mathsf{succ}(x)\}(z)$ or, somewhat abusively, $\{0, \mathsf{succ}\}(z)$. This gives

$$\cdot \vdash \{0, \mathsf{succ}\} : 1 + \mathsf{Nat} \to \mathsf{Nat}$$

as desired. More generally, we can write any $\mathsf{Nat}$-algebra as $\alpha = \{\alpha_0, \alpha_1\}$ where $\alpha_0 : 1 \to \mathsf{Nat}$ (or, equivalently, $\alpha_0 : \mathsf{Nat}$) and $\alpha_1 : \mathsf{Nat} \to \mathsf{Nat}$. We call $\alpha_0$ the *basis* or *pseudo-zero* and $\alpha_1$ the *inductive step* or *pseudo-successor*.

In fact, $\{0, \mathsf{succ}\}$ holds a special position among $\mathsf{Nat}$-algebras. The $\mathsf{Nat}$-algebras form a category and $\{0, \mathsf{succ}\}$ is the initial object in this category. Recall that this means it has a unique morphism to any other object in the category. This requires us to define

morphisms between Nat-algebras, which we will call Nat-*homomorphisms*. Given two Nat-algebras, $\alpha : 1 + A \to A$ and $\beta : 1 + B \to B$, $h : A \to B$ is a Nat-homomorphism if it makes the following diagram commute.

$$\begin{array}{ccc} 1 + A & \xrightarrow{1+h} & 1 + B \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} \\ A & \xrightarrow{h} & B \end{array}$$

The map $1 + h : 1 + A \to 1 + B$ is defined in the natural way:

$$1 + h :\equiv \{\_.\mathsf{inl}\ \langle\rangle; a.\mathsf{inr}\ h(a)\}$$

To show that Nat is an initial algebra, we must show that for every Nat-algebra $\alpha : 1 + A \to A$, there exists a unique Nat-homomorphism $! : \mathsf{Nat} \to A$ such that the following diagram commutes (note that the order of quantifications expressed in the previous sentence is not inherently clear in the diagram, and must be considered to get a full understanding of the diagram.)

$$\begin{array}{ccc} 1 + \mathsf{Nat} & \overset{1+!}{\dashrightarrow} & 1 + A \\ \downarrow{\scriptstyle \{0,\mathsf{succ}\}} & & \downarrow{\scriptstyle \alpha} \\ \mathsf{Nat} & \overset{!}{\dashrightarrow} & A \end{array}$$

Let's consider the requirements on ! for the diagram to commute.

$$!0 = \alpha_0$$

$$!\mathsf{succ}(x) = \alpha_1(!x)$$

where the left sides correspond to following the path $! \circ \{0, \mathsf{succ}\}$ and the right sides correspond to following $\alpha \circ 1{+}!$. Note that these two equations match the $\beta$ rules for rec, so we can define $! :\equiv \mathsf{rec}[A](\alpha_0; \alpha_1)$ or simply $! :\equiv \mathsf{rec}[A](\alpha)$. As we see above, the $\beta$ rules for rec imply commutation of the diagram. Uniqueness of ! follows from the $\eta$ rule for rec.

It's worth noting that commuting diagrams hide exactly the type of equality that is being discussed, which is quite important in HoTT. For Nat, for example, uniqueness of ! holds "on the nose," while, in general, uniqueness may be only up to higher homotopy.

## 3 $F$-algebras

The above discussion can be generalized to any functor $F$. A functor is a mapping between categories $C$ and $D$. Functors act on objects in a category and the morphisms between them, i.e. for all objects $X \in C$, $F(X) \in D$, and for all morphisms $f : X \to Y$ between objects $X$ and $Y$ in $C$, $F(f) : F(X) \to F(Y)$. Functors respect identity and composition:

1. For every object $X \in C$, $F(\mathsf{id}_X) = \mathsf{id}_{F(X)}$
2. For all morphisms $f : X \to Y$ and $g : Y \to Z$, $F(g \circ f) = F(g) \circ F(f)$

[1]

For example, $F_{\mathsf{Nat}}(C) :\equiv 1 + C$ is a functor. We check that $F_{\mathsf{Nat}}$ preserves identities and composition. Let $X$ be an object of $C$.

$$F_{\mathsf{Nat}}(\mathsf{id}_X) = 1 + \mathsf{id}_X = \{\langle\rangle, \mathsf{id}_X\}$$

which is indeed an identity on $1 + X$. Let $f : X \to Y, g : Y \to Z$ be morphisms between objects of $C$.

$$F_{\mathsf{Nat}}(g \circ f) = 1 + g \circ f = \{\langle\rangle, g \circ f\} = \{\langle\rangle, g\} \circ \{\langle\rangle, f\} = (1 + g) \circ (1 + f) = F(g) \circ F(f)$$

For any functor $F$, an $F$-algebra is a mapping $F(X) \to X$. Thus, a $\mathsf{Nat}$-algebra is an $F_{\mathsf{Nat}}$-algebra. $F$-algebras form categories as $\mathsf{Nat}$-algebras do. For a functor $F$, objects $A$ and $B$, two $F$-algebras $\alpha : F(A) \to A$ and $\beta : F(B) \to B$, and a morphism $h : A \to B$, the following diagram commutes.

$$\begin{array}{ccc} F(A) & \xrightarrow{F(h)} & F(B) \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} \\ A & \xrightarrow{h} & B \end{array}$$

An *initial* $F$-algebra is an $F$-algebra $i : F(I) \to I$ such that for all other $F$-algebras $\alpha : F(A) \to A$, there exists a unique map $! : I \to A$ such that the following diagram commutes.

$$\begin{array}{ccc} F(I) & \overset{F(!)}{\dashrightarrow} & F(A) \\ \downarrow{\scriptstyle i} & & \downarrow{\scriptstyle \alpha} \\ I & \overset{!}{\dashrightarrow} & A \end{array}$$

There also exists the notion of an $F$-coalgebra, which, dual to the above, is a map $\alpha : A \to F(A)$. A final $F$-coalgebra is a mapping $j : J \to F(J)$ such that for all other $F$-coalgebras $\alpha : A \to F(A)$, there exists a unique map $! : A \to J$ making the following diagram commute.

$$\begin{array}{ccc} A & \overset{!}{\dashrightarrow} & J \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle j} \\ F(A) & \overset{!}{\dashrightarrow} & F(J) \end{array}$$

**Lemma 1** (Lambek)**.** *If $i : F(I) \to I$ is an initial $F$-algebra, then $i$ is an isomorphism. That is, $F(I) \equiv I$.*

*Proof.* To show that $i$ is an isomorphism, we must exhibit an inverse $i^{-1} : I \to F(I)$ such that $i \circ i^{-1} = \mathsf{id}_I$ and $i^{-1} \circ i = \mathsf{id}_{F(I)}$. Consider the $F$-algebra $F(i) : F(F(I)) \to F(I)$. We now treat $i$ as homomorphism between the $F$-algebras $F(i)$ and $i$ making this diagram commute.

$$\begin{array}{ccc} F(F(I)) & \xrightarrow{F(i)} & F(I) \\ \downarrow{\scriptstyle F(i)} & & \downarrow{\scriptstyle i} \\ F(I) & \xrightarrow{i} & I \end{array}$$

Since $i$ is an initial $F$-algebra, however, we also have a unique mapping $! : I \to F(I)$ making the top half of this diagram commute.

$$\begin{array}{ccc} F(I) & \xrightarrow{i} & I \\ \vdots{\scriptstyle F(!)} & & \vdots{\scriptstyle !} \\ F(F(I)) & \xrightarrow{F(i)} & F(I) \\ \downarrow{\scriptstyle F(i)} & & \downarrow{\scriptstyle i} \\ F(I) & \xrightarrow{i} & I \end{array}$$

There is also a unique mapping from $I$ to $I$, which must be the identity. This indicates that the mapping $i \circ !$ along the right side of the diagram must be equal to $\mathsf{id}_I$:

$$i \circ ! = \mathsf{id}_I$$

We also have

$$! \circ i = F(i) \circ F(!) = F(i \circ !) = F(\mathsf{id}_I) = \mathsf{id}_{F(I)}$$

where the first equality follows from the commutativity of the upper half of the diagram, the second and fourth follow from the properties of functors and the third follows from the result above. □

This shows that for any functor $F$, the initial $F$-algebra $I$ is a fixed point of $F$. A dual result can be proven showing that, if $J$ is the final $F$-coalgebra of a functor $F$, then $F(J) \equiv J$.

## 4 Internalizing Nat-Algebras

Inside type theory, we can define the notion of Nat-algebra as

$$\mathsf{NatAlg} :\equiv \Sigma A{:}\mathcal{U}.\, ((1 + A) \to A)$$

We can define the type of $\mathsf{Nat}$-homomorphisms between two $\mathsf{Nat}$-algebras $(A, \alpha)$ and $(B, \beta)$ as

$$\mathsf{NatHom}(\alpha, \beta) :\equiv \Sigma h{:}A \to B.\, (\beta \circ (1 + h) = h \circ \alpha)$$

The fact that $\nu :\equiv (\mathsf{Nat}, \{0, \mathsf{succ}\})$ is initial in the category of $\mathsf{Nat}$-algebras is expressed by the fact that $\mathsf{NatHom}(\nu, \alpha)$ is contractible for all $\alpha$: this means that there exists a $\mathsf{Nat}$-homomorphism from $\nu$ to $\alpha$ which is unique up to higher homotopy.

# 5 W-types

We'd like to be able to take a functor $F$ and define the initial $F$-algebra within HoTT (if one exists). For the class of *polynomial functors*, we can do this using *Brouwer ordinals*, also called W-*types*.

W-types are inspired by the mathematical concept of well-founded induction. In classical mathematics, a partially ordered set $\langle A, < \rangle$ is said to be well-founded if every subset of $A$ has a $<$-minimal element (equivalently, there are no infinite descending chains). Well-founded sets are useful because they admit an induction principle:

> **Proposition (Well-Founded Induction):**
> Let $\langle A, < \rangle$ be a well-founded set and $P(x)$ be a proposition. If for any $a \in A$ we can prove $P(a)$ by assuming $P(b)$ for all $b < a$, then $P(a)$ holds for all $a \in A$.

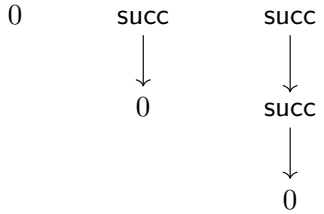
The set of natural numbers $\langle \mathbb{N}, < \rangle$ together with its usual ordering is an example of a well-founded set, and the induction principle is the familiar mathematical induction.

Classically, the proof that induction holds goes by contradiction, so this definition is unsatisfactory for a constructive theory. We will instead characterize well-founded sets as those for which we have a (constructive) induction principle.

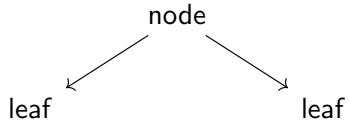
To better understand what we mean by this, we will define W-types. To form a W-type, we require a type $A$ and a type family $B$ over $A$:

$$\frac{\Gamma \vdash A : \mathcal{U} \quad \Gamma, x{:}A \vdash B : \mathcal{U}}{\Gamma \vdash \mathsf{W}x{:}A.B : \mathcal{U}}\ \mathsf{W}F$$

$A$ is the type of *node sorts*. Each node sort represents a different way of forming an element of $\mathsf{W}x{:}A.B$. For example, in the case of the natural numbers, the node sorts are 0 and $\mathsf{succ}$. We can think of each element of the natural numbers as a tree built from these two sorts of nodes. For example, the numbers 0 through 2 can be represented as



Another natural example is the type of binary trees. This type can be defined by two node sorts $\mathsf{node}$ and $\mathsf{leaf}$, with the elements taking the form of trees such as this:




In order to fully specify these two types, we need to have some notion of a node's arity. This is given by the type family $B$. For each node sort $a : A$, $B(a)$ describes the *branching factor* of $a$, the index type to specify the predecessors of a node of sort $a$. For example, the branching factor of the sort 0 or $\mathsf{leaf}$ would be 0, the branching factor of $\mathsf{succ}$ would be 1, and the branching factor of $\mathsf{node}$ would be $1+1$. Thus we can write $\mathsf{Nat}$ as $\mathsf{W}x{:}2.\,\mathsf{if}(x; 0; 1)$, where $\mathsf{tt}$ represents 0 and $\mathsf{ff}$ represents $\mathsf{succ}$.

Now that we have the purposes of $A$ and $B$ in hand, we can see how to define the introduction rule for $\mathsf{W}x{:}A.B$.

$$\frac{a : A \quad x{:}B(a) \vdash w : \mathsf{W}x{:}A.B}{\Gamma \vdash \sup[a](x.w) : \mathsf{W}x{:}A.B}\ \mathsf{W}I$$

In other words, in order to construct a node of sort $a$, we must give an element $\mathsf{W}x{:}A.B$ for each predecessor as specified by the branching factor $B(a)$. Note that when $B(a)$ is 0 we can construct a new node without any predecessor information. For example, we can construct elements of the naturals as follows:

$$\begin{aligned}
\overline{0} &:\equiv \sup[\mathsf{tt}](x.\mathsf{abort}_{\mathsf{W}x:2.\,\mathsf{if}(x;0;1)}(x)) \\
\overline{1} &:\equiv \sup[\mathsf{ff}](_.\overline{0}) \\
\overline{2} &:\equiv \sup[\mathsf{ff}](_.\overline{1})
\end{aligned}$$

The recursor for $\mathsf{W}x{:}A.B$ follows the idea of well-founded recursion: in order to define the result of a function $f$ on an element $w : \mathsf{W}x{:}A.B$, we can assume we've already computed $f$ for all of $w$'s predecessors.

$$\frac{\Gamma \vdash C : \mathcal{U} \quad \Gamma, a : A, r : B(a) \to C \vdash M : C}{\Gamma, z : \mathsf{W}x{:}A.B \vdash \mathsf{wrec}[C](a, r.M)(z) : C}\ \mathsf{W}R$$

In the hypothesis, we assume that we are dealing with a node of type $a$ and that we have the value $r(b)$ for each $b : B(a)$ indexing a predecessor. We use this information to construct the value $M$ of the recursor at the current node. The recursor comes with the compuation rule

$$\mathsf{wrec}[C](a, r.M)(\sup[a](w)) \equiv [a, \lambda z.\mathsf{wrec}[C](a, r.M)(w(z))/a, r]M$$

which, as expected, gives the value of $\mathsf{wrec}[C](a, r.M)$ on $\sup[a](w)$ in terms of the value on each predecessor $w(z)$ for $z : B(a)$.

The dependent eliminator has a similar form, expressing the idea of well-founded induction.

$$\frac{\Gamma, z : \mathsf{W}x{:}A.B \vdash P : \mathcal{U} \quad \Gamma, a{:}A, p : B(a) \to \mathsf{W}x{:}A.B, h : \prod_{b:B(a)} P(p(b)) \vdash M : P(\sup[a](p))}{\Gamma, z : \mathsf{W}x{:}A.B \vdash \mathsf{wind}[x.P](a, p, h.M) : P(z)}\ \mathsf{W}E$$

Here, in order to formulate the hypothesis, we need to assume the additional data $p : B(a) \to \mathsf{W}x{:}A.B$ which gives us the predecessors of the element we are considering. The computation rule takes the form

$$\mathsf{wind}[x.P](a, p, h.M)(\sup[a](w)) = [a, w, \lambda z.\mathsf{wind}[x.P](a, p, h.M)(w(z))/a, p, h]M$$

In general, we can only assert that this computation rule holds propositionally.

Each $\mathsf{W}$-type determines a functor, in particular a *polynomial functor*. This is a functor of the form $F(X) = \Sigma a{:}A.(B(a) \to X)$ for some type $A$ and type family $B$. We can see that the $\mathsf{W}$-type $\mathsf{W}x{:}A.B$ defines an $F$-algebra where $F(X) :\equiv \Sigma a{:}A.(B(a) \to X)$: we have $\lambda(a, w).\sup[a](w) : F(X) \to X$. The map $\lambda(a, w).\sup[a](w)$ is in fact an equivalence, and $\mathsf{W}x{:}A.B$ is a homotopy-initial $F$-algebra.

In the case of our $\mathsf{W}$-type definition of $\mathsf{Nat}$, observe that the functor determined by $\mathsf{W}x{:}\mathsf{2}.\,\mathsf{if}(x; 0; 1)$ is $F(X) = \Sigma b{:}\mathsf{2}.(\mathsf{if}(b; 0; 1) \to X)$. One can check that $\Sigma b{:}\mathsf{2}.(\mathsf{if}(b; 0; 1) \to X) \simeq 1 + X$. Thus, this type satisfies the equation $F(X) = 1 + X$, our original definition of a $\mathsf{Nat}$-algebra.

## References

[1] Wikipedia. Functor. `http://en.wikipedia.org/wiki/Functor`, 2013.