

Induction and homotopy initiality for a class of 1-HITs

Gabe Dijkstra

University of Nottingham

May 20th, 2016

j.w.w. Thorsten Altenkirch, Paolo Capriotti and Fredrik Nordvall Forsberg

Aims

Ultimately, we want to have a formal definition of higher inductive types in type theory, such that we can do things like:

- show that initiality and induction coincide
- prove correctness of hub-spokes construction
- show that we get all higher inductive types from some primitive higher inductive type

Aim of this talk

In this talk, we will show:

- how to show that induction and initiality coincide generally
- how to specify 1-HITs in type theory
- what coherence issues show up when we do so
- how to deal with these by restricting ourselves to a smaller class of 1-HITs
- how to derive an induction principle for these 1-HITs

Category theory in type theory

- We want to do category theory in type theory
- We do not want to truncate anything: we work with hom-*types*, not hom-*sets*
- We also do not want to talk about $(\infty, 1)$ -categories
- Instead, we deal with coherence *lazily*: we keep track of how much structure and coherence we need from our categories and functors and provide exactly that

Initiality and induction

Given an endofunctor $F : \text{Type} \rightarrow \text{Type}$, we can think of the *inductive type* T generated by F in two ways:

- T with its constructor $c : FT \rightarrow T$ form an initial object in the category $F\text{-alg}$ of F -algebras
- T satisfies an induction principle for F

Initiality

Define category $F\text{-alg}$:

- objects:

$$|F\text{-alg}| : \text{Type}$$

$$|F\text{-alg}| \equiv (X : \text{Type}) \times (\theta : FX \rightarrow X)$$

- morphisms:

$$F\text{-alg}(-, -) : |F\text{-alg}| \rightarrow |F\text{-alg}| \rightarrow \text{Type}$$

$$F\text{-alg}((X, \theta), (Y, \rho)) \equiv (f : X \rightarrow Y) \times (f_0 : f \circ \theta = \rho \circ Ff)$$

Note that the computation rule holds up to propositional equality.

Definition

An object X of category \mathcal{C} is (*homotopy*) *initial* if we have:

$$(Y : |\mathcal{C}|) \rightarrow \text{is-contr } \mathcal{C}(X, Y)$$

Inductive type T is the carrier of the initial object of $F\text{-alg}$ with its constructor $c : FT \rightarrow T$ as its algebra structure.

Induction principle

T : Type with constructor $c : FT \rightarrow T$ satisfies the induction principle if for any *algebra family*:

- $P : T \rightarrow \text{Type}$
- $m : (x : FT) \times \square_F P\ x \rightarrow P\ (\theta\ x)$

we get a *dependent algebra morphism*:

- $s : (x : T) \rightarrow P\ x$
- $s_0 : (x : FT) \rightarrow s\ (\theta\ x) = m\ x\ (\bar{F}\ s\ x)$

Notation

- $\square_F P : FT \rightarrow \text{Type}$ lifts the family P on T to FT
- $\bar{F} : ((x : T) \rightarrow P\ x) \rightarrow (x : FT) \rightarrow \square_F P\ x$ lifts dependent functions on P to $\square_F P$

Initiality versus induction in type theory

- For ordinary inductive types initiality and induction have been shown to be equivalent (Sojakova et al., 2012)
- For initiality we only need objects and morphisms
- For the induction principle we need more structure...

Induction – categorically

Instead of proving:

$$\text{initiality} \iff \text{induction}$$

directly, we introduce an intermediate concept:

$$\text{initiality} \iff \text{section induction} \iff \text{induction}$$

Definition

An object $X : |\mathcal{C}|$ satisfies the *section induction principle* if for every $Y : |\mathcal{C}|$ any $p : \mathcal{C}(Y, X)$ has a section $s : \mathcal{C}(X, Y)$

Initiality implies induction – categorically

Suppose $X : |\mathcal{C}|$ initial, then for any Y with $p : \mathcal{C}(Y, X)$ we have a unique $s : \mathcal{C}(X, Y)$ which is a section of p :

$$\begin{array}{c} Y \\ \downarrow p \\ X \end{array}$$

Initiality implies induction – categorically

Suppose $X : |\mathcal{C}|$ initial, then for any Y with $p : \mathcal{C}(Y, X)$ we have a unique $s : \mathcal{C}(X, Y)$ which is a section of p :

$$\begin{array}{ccc} X & \xrightarrow{s} & Y \\ & & \downarrow p \\ & & X \end{array}$$

Initiality implies induction – categorically

Suppose $X : |\mathcal{C}|$ initial, then for any Y with $p : \mathcal{C}(Y, X)$ we have a unique $s : \mathcal{C}(X, Y)$ which is a section of p :

$$\begin{array}{ccc} X & \xrightarrow{s} & Y \\ & \searrow \text{Id}_X & \downarrow p \\ & & X \end{array}$$

Induction implies initiality – categorically

Suppose we have $X : |\mathcal{C}|$ that satisfies the section induction principle. Assuming \mathcal{C} has products, then for any Y we have the projection:

$$X \times Y \xrightarrow{\pi_1} X$$

which has a section s . Define $f : \mathcal{C}(X, Y)$ as the composite:

$$X \xrightarrow{s} X \times Y \xrightarrow{\pi_2} Y$$

We have to show that any other $g : \mathcal{C}(X, Y)$ is equal to this f . Taking the equaliser of f with any g , we get:

$$X \begin{array}{c} \xrightarrow{f} \\ \xRightarrow{\quad} \\ \xrightarrow{g} \end{array} Y$$

Induction implies initiality – categorically

Suppose we have $X : |\mathcal{C}|$ that satisfies the section induction principle. Assuming \mathcal{C} has products, then for any Y we have the projection:

$$X \times Y \xrightarrow{\pi_1} X$$

which has a section s . Define $f : \mathcal{C}(X, Y)$ as the composite:

$$X \xrightarrow{s} X \times Y \xrightarrow{\pi_2} Y$$

We have to show that any other $g : \mathcal{C}(X, Y)$ is equal to this f . Taking the equaliser of f with any g , we get:

$$E \xrightarrow{e} X \begin{array}{c} \xrightarrow{f} \\ \xRightarrow{g} \end{array} Y$$

Induction implies initiality – categorically

Suppose we have $X : |\mathcal{C}|$ that satisfies the section induction principle. Assuming \mathcal{C} has products, then for any Y we have the projection:

$$X \times Y \xrightarrow{\pi_1} X$$

which has a section s . Define $f : \mathcal{C}(X, Y)$ as the composite:

$$X \xrightarrow{s} X \times Y \xrightarrow{\pi_2} Y$$

We have to show that any other $g : \mathcal{C}(X, Y)$ is equal to this f . Taking the equaliser of f with any g , we get:

$$\begin{array}{ccccc} E & \xrightarrow{e} & X & \begin{array}{c} \xrightarrow{f} \\ \xRightarrow{g} \end{array} & Y \\ \uparrow s & \nearrow \text{Id}_X & & & \\ X & & & & \end{array}$$

Category structure needed

To formalise the previous arguments, we need the following:

- For initiality: objects and morphisms
- For sections:
 - Composition
 - Identity morphisms
- To show that initiality and section principle coincide:
 - Associativity
 - Identity laws
 - Products
 - Equalisers

Category structure needed - ordinary inductive types

Given $F : \text{Type} \rightarrow \text{Type}$ as a *container*, we can define $F\text{-alg}$ as:

- objects:

$$|F\text{-alg}| : \text{Type}$$

$$|F\text{-alg}| \equiv (X : \text{Type}) \times (\theta : FX \rightarrow X)$$

- morphisms:

$$F\text{-alg}(-, -) : |F\text{-alg}| \rightarrow |F\text{-alg}| \rightarrow \text{Type}$$

$$F\text{-alg}((X, \theta), (Y, \rho)) \equiv (f : X \rightarrow Y) \times (f_0 : f \circ \theta = \rho \circ Ff)$$

- With some effort, we can define all the categorical structure needed
- However, categorical laws are not satisfied strictly

Higher inductive types versus ordinary inductive types

Ordinary inductive type T with constructors

- $c_0 : F_0 T \rightarrow T$
- \vdots
- $c_k : F_k T \rightarrow T$

where every $F_i : \text{Type} \rightarrow \text{Type}$ is a (strictly positive) functor.

Higher inductive types versus ordinary inductive types

Ordinary inductive type T with constructor:

- $c : F_0 T + \dots + F_k T \rightarrow T$

where every $F_i : \text{Type} \rightarrow \text{Type}$ is a (strictly positive) functor.

Higher inductive types versus ordinary inductive types

Ordinary inductive type type T with constructor:

- $c : FT \rightarrow T$

where $F : \text{Type} \rightarrow \text{Type}$ is a (strictly positive) functor.

Higher inductive types versus ordinary inductive types

Higher inductive types, e.g. the circle S^1 has constructors:

- $\text{base} : S^1$
- $\text{loop} : \text{base} =_{S^1} \text{base}$

Differences from an ordinary inductive type:

- Dependencies on previous constructors
- *Higher* constructors: target of constructors not always T , but can also be an iterated path space of T .

Single functor $\text{Type} \rightarrow \text{Type}$ no longer suffices

Specifying 1-HITs

A specification of a 1-HIT is either:

- empty (denoted ϵ)
- a specification s extended with a 0-constructor on alg_s ,
- or a specification s extended with a 1-constructor on alg_s

Defined mutually with the type of specifications is the function that maps a specification to its category of algebras:

$$\text{alg} : \text{Spec} \rightarrow \text{Cat}$$

For the empty specification $\text{alg}_\epsilon \equiv \text{Type}$

We also define mutually the underlying functor U_s which gives us the underlying type of the algebra.

Specifying 1-HITs – category of algebras – 0-constructors

Given $s : \text{Spec}$, a 0-constructor is given by a functor:

$$F : \text{alg}_s \rightarrow \text{Type}$$

and the category $\text{alg}_{(s,F)}$ is defined as having:

- objects:
 - $X : |\text{alg}_s|$
 - $\theta : FX \rightarrow U_s X$
- morphisms $(X, \theta) \rightarrow (Y, \rho)$:
 - $f : \text{alg}_s(X, Y)$
 - $f_0 : U_s f \circ \theta = \rho \circ Ff$

Specifying 1-HITs – category of algebras – 1-constructors

Given $s : \text{Spec}$, a 1-constructor is given by specifying its arguments:

$$F : \text{alg}_s \rightarrow \text{Type}$$

and the endpoints of the path, as natural transformations:

$$l, r : F \rightarrow U_s$$

The category of algebras is then:

- objects:

- $X : |\text{alg}_s|$
- $\theta : l_X =_{U_s X} r_X$

- morphisms $(X, \theta) \rightarrow (Y, \rho)$:

- $f : \text{alg}_s(X, Y)$
- $f_0 : Uf \circ l_X \xrightarrow{Uf \circ \theta} Uf \circ r_X$
 $\begin{array}{ccc} \alpha_f \downarrow & & \downarrow \beta_f \\ l_Y \circ Ff & \xrightarrow{\rho \circ Ff} & r_Y \circ Ff \end{array}$

Functors and natural transformations in type theory

- Functors $F : \mathbf{alg}_s \rightarrow \mathbf{Type}$ need to be *strictly positive*
- Strictly positive functors $\mathbf{Type} \rightarrow \mathbf{Type}$: *containers*
- For a given $\mathcal{C} : \mathbf{Cat}$, strictly positive functors $\mathcal{C} \rightarrow \mathbf{Type}$: *\mathcal{C} -containers*
- Natural transformations between strictly positive functors: *container morphisms*
- Forgetful functors $U_s : \mathbf{alg}_s \rightarrow \mathbf{Type}$ are containers if they have a left adjoint

Containers on Type

Strictly positive functors $\text{Type} \rightarrow \text{Type}$: containers

- Shapes $S : \text{Type}$
- Positions $T : S \rightarrow \text{Type}$

$$\llbracket S \triangleleft P \rrbracket_0 : \text{Type} \rightarrow \text{Type}$$

$$\llbracket S \triangleleft P \rrbracket_0 X \equiv (s : S) \times (P\ s \rightarrow X)$$

$$\llbracket S \triangleleft P \rrbracket_1 : (X \rightarrow Y) \rightarrow \llbracket S \triangleleft P \rrbracket_0 X \rightarrow \llbracket S \triangleleft P \rrbracket_0 Y$$

$$\llbracket S \triangleleft P \rrbracket_1 f\ (s, t) \equiv (s, f \circ t)$$

The functor laws follow from the categorical laws of Type , i.e. they are satisfied *strictly*

\mathcal{C} -containers

Strictly positive functors $\mathcal{C} \rightarrow \text{Type}$: \mathcal{C} -containers (or *familiarily representable*)

- Shapes $S : \text{Type}$
- Positions $T : S \rightarrow |\mathcal{C}|$

with

$$\llbracket S \triangleleft P \rrbracket_0 : \mathcal{C} \rightarrow \text{Type}$$

$$\llbracket S \triangleleft P \rrbracket_0 X \equiv (s : S) \times \mathcal{C}(P\ s, X)$$

$$\llbracket S \triangleleft P \rrbracket_1 : \mathcal{C}(X, Y) \rightarrow \llbracket S \triangleleft P \rrbracket_0 X \rightarrow \llbracket S \triangleleft P \rrbracket_0 Y$$

$$\llbracket S \triangleleft P \rrbracket_1 f\ (s, t) \equiv (s, f \circ t)$$

The functor laws follow from the categorical laws of \mathcal{C} , i.e. associativity and identity laws

\mathcal{C} -container morphisms

Natural transformations between containers: *container morphisms*:

For containers $S \triangleleft P$ and $T \triangleleft Q$, container morphisms are:

$$(f : S \rightarrow T) \times (g : (a : S) \rightarrow \mathcal{C}(Q (f a), P a))$$

with

$$\begin{aligned} \text{apply } (f, g) : (X : |\mathcal{C}|) &\rightarrow \llbracket S \triangleleft P \rrbracket_0 X \rightarrow \llbracket T \triangleleft Q \rrbracket_0 X \\ \text{apply } (f, g) X (s, t) &:\equiv (f s, t \circ (g s)) \end{aligned}$$

Naturality follows from associativity of \mathcal{C}

0-HITs and coherence

Consider a specification with three 0-constructors, i.e. the category of algebras has objects:

- $X : \text{Type}$
- $\theta_0 : F_0 X \rightarrow X$
- $\theta_1 : F_1(X, \theta_0) \rightarrow X$
- $\theta_2 : F_2(X, \theta_0, \theta_1) \rightarrow X$

Supposing all functors F_i are given as containers, then:

- identity morphisms in $F_2\text{-alg}$
- \Leftarrow identity laws of functor F_2
- \Leftarrow identity laws of category $F_1\text{-alg}$
- \Leftarrow coherence of identity laws of $F_0\text{-alg}$ and composition law F_1
- \Leftarrow coherence of identity and associativity laws of $F_0\text{-alg}$

Coherence issues increase with the amount of constructors, even if we only have 0-constructors

1-HITs

We will look at 1-HITs T with constructors:

- $c_0 : F_0 T \rightarrow T$
- $c_1 : (x : F_1 T) \rightarrow c_0^* (l_T x) =_T c_0^* (r_T x)$

where:

- $F_0, F_1 : \text{Type} \rightarrow \text{Type}$ functors given as containers
- $F_0^* : \text{Type} \rightarrow \text{Type}$ is the free monad of F_0 , also given as a container
- $c_0^* : F_0^* X \rightarrow X$ is the algebra c_0 lifted to the free monad F_0^*
- $l, r : F_1 \rightarrow F_0^*$ natural transformations given as container morphisms

1-HITs

We will look at 1-HITs T with constructors:

- $c_0 : F_0 T \rightarrow T$
- $c_1 : c_0^* \circ l_T =_T c_0^* \circ r_T$

where:

- $F_0, F_1 : \text{Type} \rightarrow \text{Type}$ functors given as containers
- $F_0^* : \text{Type} \rightarrow \text{Type}$ is the free monad of F_0 , also given as a container
- $c_0^* : F_0^* X \rightarrow X$ is the algebra c_0 lifted to the free monad F_0^*
- $l, r : F_1 \rightarrow F_0^*$ natural transformations given as container morphisms

1-HITs – algebras

Given a specification (F_0, F_1, l, r) , the category of algebras has:

- objects:

- $X : \text{Type}$
- $\theta_0 : F_0 X \rightarrow X$
- $\theta_1 : \theta_0^* \circ l_X = \theta_0^* \circ r_X$

- morphisms $(X, \theta_0, \theta_1) \rightarrow (Y, \rho_0, \rho_1)$:

- $f : X \rightarrow Y$
- $f_0 : f \circ \theta_0 = \rho_0 \circ F_0 f$
- $f_1 : \begin{array}{ccc} f \circ \theta_0^* \circ l_X & \xrightarrow{f \circ \theta_1} & f \circ \theta_0^* \circ r_X \\ \left. \begin{array}{c} f_0^* \circ l_X \\ \hline \rho_0^* \circ l_Y \circ F_1 f \end{array} \right| & & \left. \begin{array}{c} f_0^* \circ r_X \\ \hline \rho_0^* \circ r_Y \circ F_1 f \end{array} \right| \\ & \xrightarrow{\rho_1 \circ F_1 f} & \end{array}$

1-HITs – algebras

Defining the category structure is involved:

- F_0, F_1 and l, r satisfy their respective laws strictly,
- $_* : |F\text{-alg}| \rightarrow |F^*\text{-alg}|$ is not strictly functorial however
- We need to show that its functoriality is coherent with the category structure
- There is a lot of path algebra involved
- Cubical methods from the HoTT-Agda library make life a bit easier

1-HITs – induction principle

We need to define algebra families and dependent algebra morphisms

We have:

$$(X \rightarrow \text{Type}) = (Y : \text{Type}) \times (p : Y \rightarrow X)$$

as witnessed by:

$$\text{to } P \equiv (\Sigma X P, \pi_1)$$

$$\text{from } (Y, p) \equiv p^{-1}$$

Under this equivalence, *sections* correspond to *dependent functions*

We can derive algebra families and dependent algebra families by finding similar equivalences, replacing types and functions with algebras and algebra morphisms.

1-HITs – induction principle – families

Suppose $(X, \theta_0, \theta_1) : |\text{alg}_{(F_0, F_1, l, r)}|$, then the type of algebra families over (X, θ_0, θ_1) is an $M : (X \rightarrow \text{Type}) \rightarrow \text{Type}$ that satisfies satisfies the equation:

$$\begin{aligned} (P : X \rightarrow \text{Type}) \times M P = & (Y : \text{Type}) \\ & \times (\rho : FY \rightarrow Y) \\ & \times (p : Y \rightarrow X) \\ & \times (p_0 : \dots) \\ & \times (p_1 : \dots) \end{aligned}$$

1-HITs – induction principle – families

We can solve the previous equation by applying the witnesses of the equivalence $(P : X \rightarrow \text{Type}) = (Y : \text{Type}) \times (p : Y \rightarrow X)$. An algebra family then consists of:

$$\begin{aligned} & (P : X \rightarrow \text{Type}) \\ & \times (m_0 : (x : F_0 X) \times \Box_{F_0} P \ x \rightarrow P \ (\theta_0 \ x)) \\ & \times (m_1 : (x : F_1 X) \times (y : \Box_{F_1} P \ x) \\ & \quad \rightarrow m_0^* (l^d(x, y)) = m_0^* (r^d(x, y)) [P \downarrow \theta_1 \ x]) \end{aligned}$$

where

- \Box_F satisfies $F(\Sigma X P) = \Sigma(FX)(\Box_F P)$
- $m_0^* : (x : F_0^* X) \times \Box_{F_0^*} P \ x \rightarrow P \ (\theta_0^* \ x)$
- $l^d, r^d : (x : F_1 X) \times \Box_{F_1} P \ x \rightarrow (x : F_0^* X) \times \Box_{F_0^*} P \ x$

1-HITs – induction principle – dependent morphisms

We can also figure out what under this equivalence the sections amount to and we arrive at the following, given an algebra family:

- $P : X \rightarrow \text{Type}$
- $m_0 : (x : F_0 X) \times \square_{F_0} P \ x \rightarrow P (\theta_0 \ x)$
- $m_1 : (x : F_1 X) \times (y : \square_{F_1} P \ x) \rightarrow m_0^* (l^d(x, y)) = m_0^* (r^d(x, y)) [P \downarrow \theta_1 \ x]$

a dependent algebra morphism over (P, m_0, m_1) consists of:

- $s : (x : X) \rightarrow P \ x$
- $s_0 : (x : F_0 X) \rightarrow s (\theta_0 \ x) = m_0 \ x (\bar{F}_0 \ s \ x)$
- $s_1 : (x : F_1 X) \rightarrow$

$$\begin{array}{ccc}
 s (\theta_0^* (l \ x)) & \xrightarrow{\frac{s (\theta_1 \ x)}{[P \downarrow \theta_1 \ x]}} & s (\theta_0^* (r \ x)) \\
 \left| s_0^* (l \ x) \right. & & \left| s_0^* (r \ x) \right. \\
 m_0^* (l^d(x, \bar{F}_1 \ s \ x)) & \xrightarrow{\frac{m_1 \ x (\bar{F}_1 \ s \ x)}{[P \downarrow \theta_1 \ x]}} & m_0^* (r^d(x, \bar{F}_1 \ s \ x))
 \end{array}$$

Conclusions

- We can
 - define 1-HITs in type theory
 - define induction for a class of them
 - show that for this class induction and initiality coincide
- Coherence issues increase with the number of constructors
- Dealing with these generally requires heavy machinery, e.g. $(\infty, 1)$ -categories
- Agda formalisation is a work in progress:
<https://github.com/gdijkstra/homotopy-initiality>