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

Goal

- Ultimately: have a way to specify higher inductive types
- For every specification we should be able to generate:
  - the category of algebras
  - introduction rules and induction principle
- *Sanity check*: show that initiality and induction coincide
- In this talk: initiality and induction coincide for a class of 1-HITs
- We want to do this all in type theory itself

Category theory in type theory

- We want to talk about categories 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
- We will 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

Put this on two slides?

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

Initiality

Define category F-alg:

- objects:
  $(X : \text{Type}) \times (\theta : FX \to X)$
- morphisms $(X, \theta) \to (Y, \rho)$:
  $(f : X \to Y) \times (f_0 : f \circ \theta = \rho \circ Ff)$

T is the carrier of the initial object of F-alg with its constructor $c : FT \to T$ its algebra structure.

Induction

T : Type with constructor $c : FT \to T$ satisfies the induction principle if for all:

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

we get:

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

Initiality versus induction in type theory

- An object X of category $\mathcal{C}$ is *(homotopy) initial* if we have: $(Y : |\mathcal{C}|) \to \text{is-contr}(\mathcal{C}(X, Y))$
- For ordinary inductive types initiality and induction are equivalent (Sojakova et al.)

  Get proper year citation and stuff
- For initiality we only need objects and morphisms
- For induction we need more structure

Induction categorically

Instead of proving:

initiality $\iff$ induction

directly, we instead show:

initiality $\iff$ *section* induction $\iff$ induction

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

Show animation for this

Induction implies initiality

Show animation for this

Structure needed

- To talk about initiality, defining objects and morphisms suffices
- Defining sections requires:
  - Composition
  - Identity morphisms
- Showing that initiality and section principle coincide requires:
  - Products
  - Equalisers
  - Associativity
  - Identity laws

Higher inductive types versus ordinary inductive types

Ordinary inductive type T with constructors

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

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

Higher inductive types versus ordinary inductive types

Ordinary inductive type T with constructor:

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

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

Higher inductive types versus ordinary inductive types

Ordinary inductive type type T with constructor:

- $c : FT \to T$

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

Higher inductive types versus ordinary inductive types

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

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

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

Single functor Type $\rightarrow$ Type no longer suffices

General framework

Constructors are *dependent dialgebras*, type T with constructors:

- $c_0 : (x : F_0\, T) \to G_0(T, x)$
- $c_1 : (x : F_1(T, c_0)) \to G_1((T, c_0), x)$
- $\vdots$
- $c_k : (x : F_k(T, c_0, \ldots, c_{k-1})) \to G_k((T, c_0, \ldots, c_{k-1}), x)$