A Syntax for Higher Inductive-Inductive Types[1]

Ambrus Kaposi, **András Kovács**

Eötvös Loránd University, Budapest

FSCD, Oxford, 10 July 2018

[1]This work was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

Motivation, overview

Higher inductive types (HITs) allow inductive equality (path) constructors.

Motivation, overview

Higher inductive types (HITs) allow inductive equality (path) constructors.

Numerous specific HITs are used in homotopy type theory.

Motivation, overview

Higher inductive types (HITs) allow inductive equality (path) constructors.

Numerous specific HITs are used in homotopy type theory.

We would like a definition for what well-formed HITs are, and what the corresponding induction principles are, for a class of HITs that covers most interesting cases.

Motivation, overview

Higher inductive types (HITs) allow inductive equality (path) constructors.

Numerous specific HITs are used in homotopy type theory.

We would like a definition for what well-formed HITs are, and what the corresponding induction principles are, for a class of HITs that covers most interesting cases.

We propose a syntax for higher inductive-inductive types (HIITs), which covers almost every HIT in the wild.

Motivation, overview

Higher inductive types (HITs) allow inductive equality (path) constructors.

Numerous specific HITs are used in homotopy type theory.

We would like a definition for what well-formed HITs are, and what the corresponding induction principles are, for a class of HITs that covers most interesting cases.

We propose a syntax for higher inductive-inductive types (HIITs), which covers almost every HIT in the wild.

We have formalized this work in Agda, and also implemented a HIIT-checker and eliminator-generator as a standalone program, both available at `https://bitbucket.org/akaposi/elims`.

Motivation, overview

Higher inductive types (HITs) allow inductive equality (path) constructors.

Numerous specific HITs are used in homotopy type theory.

We would like a definition for what well-formed HITs are, and what the corresponding induction principles are, for a class of HITs that covers most interesting cases.

We propose a syntax for higher inductive-inductive types (HIITs), which covers almost every HIT in the wild.

We have formalized this work in Agda, and also implemented a HIIT-checker and eliminator-generator as a standalone program, both available at `https://bitbucket.org/akaposi/elims`.

Future work: (higher) categorical semantics, existence.

Outline

1. Inductive types, in general
2. Syntax and induction for HIITs
3. WIP and future work

Contents

Natural numbers as usual

In pseudo-Agda.

```
data ℕ : Type where
  zero : ℕ
  suc  : ℕ → ℕ

ℕ-ind :
    (P : ℕ → Type)
  → P zero
  → ((n : ℕ) → P n → P (suc n))
  → (n : ℕ) → P n
ℕ-ind P z s zero    = z
ℕ-ind P z s (suc n) = s n (ℕ-ind P z s n)
```

Alternatively

```
ℕ-Algebra : Type
ℕ-Algebra = Σ(N : Type) × N × (N → N)

ℕ-DisplayedAlg : ℕ-Algebra → Type
ℕ-DisplayedAlg (N, z, s) =
  Σ(Nd : N → Type) × Nd z × ((n : N) → Nd n → Nd (s n))

ℕ-Section : (α : ℕ-Algebra) → ℕ-DisplayedAlg α → Type
ℕ-Section (N, z, s) (Nd, zd, sd) =
  Σ(Ns : (n : N) → Nd n) × (Ns z = zd)
                         × ((n : N) → Ns (s n) = sd n (Ns n))
```

Then, the following are definable in Agda/Coq:

```
ℕ           : ℕ-Algebra
ℕ-Induction : (D : ℕ-DisplayedAlg ℕ) → ℕ-Section ℕ D
```

(We borrow "displayed" from [Ahrens and Lumsdaine, 2017])

Initial algebra, displayed algebra over initial algebra, section of displayed algebra.

```
data ℕ : Type where
  zero : ℕ
  suc  : ℕ → ℕ

ℕ-ind :
    (P : ℕ → Type)
  → P zero
  → ((n : ℕ) → P n → P (suc n))
  → (n : ℕ) → P n
ℕ-ind P z s zero    = z
ℕ-ind P z s (suc n) = s n (ℕ-ind P z s n)
```

Induction in general

For each inductive type, we need notions of:

```
Algebra       : Type
DisplayedAlg  : Algebra → Type
Section       : (α : Algebra) → DisplayedAlg α → Type
```

Induction in general

For each inductive type, we need notions of:

```
Algebra       : Type
DisplayedAlg  : Algebra → Type
Section       : (α : Algebra) → DisplayedAlg α → Type
```

Such that the following exist (we don't show this in the current work):

```
InitialAlg : Algebra
Induction  : (D : DisplayedAlg InitialAlg) → Section InitialAlg D
```

Induction in general

For each inductive type, we need notions of:

```
Algebra      : Type
DisplayedAlg : Algebra → Type
Section      : (α : Algebra) → DisplayedAlg α → Type
```

Such that the following exist (we don't show this in the current work):

```
InitialAlg : Algebra
Induction  : (D : DisplayedAlg InitialAlg) → Section InitialAlg D
```

(There are more laws and operations on displayed algebras and sections which we could sensibly require)

Terminology

constructors	initial algebra
induction motives and methods	displayed algebra over initial algebra
eliminators and β-rules	section of displayed algebra

Contents

Desired features of valid HIIT signatures

Possible dependencies

Type/term/path constructors depending on any previous constructor.

Example for type-type and term-term dependencies: a fragment of a syntax of a type theory [Altenkirch and Kaposi, 2016].

```
Con : Type
Ty  : Con → Type
•   : Con
_▶_ : (Γ : Con) → Ty Γ → Con
Pi  : (Γ : Con)(A : Ty Γ) → Ty (Γ ▶ A) → Ty Γ
...
```

Other examples: Cauchy reals, surreal numbers.

Referring to external signature

We want to refer to already existing “external” constants.

For example, to natural numbers and a given `A` element type for length-indexed vectors.

```
Vec  : ℕ → Type
nil  : Vec zero
cons : (n : ℕ) → A → Vec n → Vec (suc n)
```

Path constructors

At possibly higher dimensions, with recursive paths, e.g. in set truncation for some external `A` type:

```
‖A‖₀   : Type
|_|₀   : A → ‖A‖₀
trunc : (x y : ‖A‖₀)(p q : x = y) → p = q
```

Possibly with path induction on previous paths, as in the definition of the torus, where ▪ denotes path composition:

```
T² : Type
b  : T²
p  : b = b
q  : b = b
t  : p ▪ q = q ▪ p
```

Strict positivity

An illegal signature:

```
Tm  : Type
con : (Tm → Tm) → Tm
```

HIIT signatures

Type dependencies are best described by a type theory.

HIIT signatures

Type dependencies are best described by a type theory.

Every HIIT signature is a context in a *type theory of signatures*.

HIIT signatures

Type dependencies are best described by a type theory.

Every HIIT signature is a context in a *type theory of signatures*.

Strict positivity is enforced by a *universe* and typing rules for functions.

HIIT signatures

Type dependencies are best described by a type theory.

Every HIIT signature is a context in a *type theory of signatures*.

Strict positivity is enforced by a *universe* and typing rules for functions.

We compute notions of algebras, displayed algebras and sections by induction on the syntax.

Theory of signatures: setup

Algebras, displayed algebras, sections given by syntactic translation from ToS to a conventional "target" type theory.

Theory of signatures: setup

Algebras, displayed algebras, sections given by syntactic translation from ToS to a conventional "target" type theory.

The target theory has Σ-types, dependent functions (denoted $(x : A) \to B$), identity, unit type and Russell-style universes, and has expressions in red.

Theory of signatures: setup

Algebras, displayed algebras, sections given by syntactic translation from ToS to a conventional "target" type theory.

The target theory has Σ-types, dependent functions (denoted $(x : A) \to B$), identity, unit type and Russell-style universes, and has expressions in red.

In the ToS, everything additionally depends on a target theory context, which serves as the source of non-inductive external symbols.

Theory of signatures (1)

Getting closed inductive-inductive types: by a universe and a "strictly positive" function space.

$$\frac{}{\Gamma;\Delta \vdash \mathsf{U}} \qquad \frac{\Gamma;\Delta \vdash a : \mathsf{U}}{\Gamma;\Delta \vdash \underline{a}}$$

$$\frac{\Gamma;\Delta \vdash a : \mathsf{U} \qquad \Gamma;\Delta, x : \underline{a} \vdash B}{\Gamma;\Delta \vdash (x : a) \to B} \qquad \frac{\Gamma;\Delta \vdash t : (x : a) \to B \qquad \Gamma;\Delta \vdash u : \underline{a}}{\Gamma;\Delta \vdash t\,u : B[x \mapsto u]}$$

Signature of natural numbers:

$$\cdot \vdash \cdot,\ \ \mathit{Nat} : \mathsf{U},\ \ \mathit{zero} : \underline{\mathit{Nat}},\ \ \mathit{suc} : \mathit{Nat} \to \underline{\mathit{Nat}}$$

Theory of signatures (2)

Universe is closed under equality of small terms, yielding recursive equalities and higher constructors.

$$\frac{\Gamma;\Delta\vdash a:\mathsf{U}\qquad\Gamma;\Delta\vdash t:\underline{a}\qquad\Gamma;\Delta\vdash u:\underline{a}}{\Gamma;\Delta\vdash t=_a u:\mathsf{U}}\qquad\frac{\Gamma;\Delta\vdash t:\underline{a}}{\Gamma;\Delta\vdash \mathsf{refl}:\underline{t=_a t}}$$

(+ path induction with propositional β-rule)

Signature of circle:

$$\cdot\vdash\cdot,\ \ S^1:\mathsf{U},\ \ base:\underline{S^1},\ \ loop:\underline{base=_{S^1} base}$$

Theory of signatures (3)

Non-inductive parameters, infinitary constructors (application rules omitted):

$$\frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma \vdash \Delta \qquad (\Gamma, x : A); \Delta \vdash B}{\Gamma; \Delta \vdash (x : A) \to B}$$

$$\frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma \vdash \Delta \qquad (\Gamma, x : A); \Delta \vdash b : \mathsf{U}}{\Gamma; \Delta \vdash (x : A) \to b : \mathsf{U}}$$

Signature of *W*-types:

$$S : \mathsf{Type},\ P : S \to \mathsf{Type} \vdash \ \cdot, \ \ W : \mathsf{U}, \ \ sup : (s : S) \to ((p : P\, s) \to W) \to \underline{W}$$

Algebras: standard model

Specification:

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta^{\mathsf{A}} : \mathsf{Type}} \qquad \frac{\Gamma;\Delta \vdash A}{\Gamma \vdash A^{\mathsf{A}} : \Delta^{\mathsf{A}} \to \mathsf{Type}} \qquad \frac{\Gamma;\Delta \vdash t : A}{\Gamma \vdash t^{\mathsf{A}} : (\delta : \Delta^{\mathsf{A}}) \to A^{\mathsf{A}}\,\delta}$$

Action:

$$\begin{aligned}
\cdot^{\mathsf{A}} &:\equiv \top \\
(\Delta, x : A)^{\mathsf{A}} &:\equiv \Sigma(\delta : \Delta^{\mathsf{A}}) \times A^{\mathsf{A}}\,\delta \\
x^{\mathsf{A}}\,\delta &:\equiv x^{\text{th}} \text{ component in } \delta \\
\mathsf{U}^{\mathsf{A}}\,\delta &:\equiv \mathsf{Type} \\
(\underline{a})^{\mathsf{A}}\,\delta &:\equiv a^{\mathsf{A}}\,\delta \\
((x : a) \to B)^{\mathsf{A}}\,\delta &:\equiv (x : a^{\mathsf{A}}\,\delta) \to B^{\mathsf{A}}\,(\delta, x) \\
&\ldots
\end{aligned}$$

Displayed algebras: logical predicate interpretation

Analogously to [Bernardy et al., 2012]. Specification:

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta^{\mathsf{D}} : \Delta^{\mathsf{A}} \to \mathsf{Type}} \qquad \frac{\Gamma ; \Delta \vdash A}{\Gamma \vdash A^{\mathsf{D}} : (\delta : \Delta^{\mathsf{A}}) \to \Delta^{\mathsf{D}}\,\delta \to A^{\mathsf{A}}\,\delta \to \mathsf{Type}}$$

$$\frac{\Gamma ; \Delta \vdash t : A}{\Gamma \vdash t^{\mathsf{D}} : (\delta : \Delta^{\mathsf{A}}) \to (\delta^{D} : \Delta^{\mathsf{D}}\,\delta) \to A^{\mathsf{D}}\,\delta\,\delta^{D}\,(t^{\mathsf{A}}\,\delta)}$$

Action:

$$\begin{aligned}
\cdot^{\mathsf{D}}\;\delta &:\equiv \top \\
(\Delta, x : A)^{\mathsf{D}}\,\delta\,\delta^{D} &:\equiv \Sigma(\delta^{D} : \Delta^{\mathsf{D}}\,\delta) \times A^{\mathsf{D}}\,\delta\,\delta^{\mathsf{D}} \\
\mathsf{U}^{\mathsf{D}}\,\delta\,\delta^{\mathsf{D}}\,A &:\equiv A \to \mathsf{Type} \\
((x : a) \to B)^{\mathsf{D}}\,\delta\,\delta^{D}\,f &:\equiv (x : a^{\mathsf{A}}\,\delta)(x^{D} : a^{\mathsf{D}}\,\delta\,\delta^{D}\,x) \to B^{\mathsf{D}}\,(\delta\,x)\,(\delta^{D}\,x^{D})\,(f x) \\
&\dots
\end{aligned}$$

- Algebras and displayed algebras are given by known interpretations.

- Algebras and displayed algebras are given by known interpretations.
- Moreover, these interpretations work regardless of strict positivity.

- Algebras and displayed algebras are given by known interpretations.
- Moreover, these interpretations work regardless of strict positivity.
- This was an early motivation of [Reynolds, 1983] for logical relations vs. homomorphisms, since the latter don't work for negative signatures.

- Algebras and displayed algebras are given by known interpretations.
- Moreover, these interpretations work regardless of strict positivity.
- This was an early motivation of [Reynolds, 1983] for logical relations vs. homomorphisms, since the latter don't work for negative signatures.
- For strictly positive signatures, we can recover homomorphisms and sections (which are "dependent" homomorphisms).

Sections (1)

Specification:

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta^{\mathsf{S}} : (\delta : \Delta^{\mathsf{A}}) \to \Delta^{\mathsf{D}}\, \delta \to \mathsf{Type}}$$

$$\frac{\Gamma ; \Delta \vdash A}{\Gamma \vdash A^{\mathsf{S}} : (\delta : \Delta^{\mathsf{A}})(\delta^{D} : \Delta^{\mathsf{D}}\, \delta)(\delta^{S} : \Delta^{\mathsf{S}} \delta\, \delta^{D})(\alpha : A^{\mathsf{A}}\, \delta) \to A^{\mathsf{D}}\, \delta\, \delta^{D}\, \alpha \to \mathsf{Type}}$$

$$\frac{\Gamma ; \Delta \vdash t : A}{\Gamma \vdash t^{\mathsf{S}} : (\delta : \Delta^{\mathsf{A}})(\delta^{D} : \Delta^{\mathsf{D}}\, \delta)(\delta^{S} : \Delta^{\mathsf{S}} \delta\, \delta^{D}) \to A^{\mathsf{S}}\, \delta\, \delta^{D}\, \delta^{S}\, (t^{\mathsf{A}}\, \delta)\, (t^{\mathsf{D}}\, \delta\, \delta^{D})}$$

Every context is is interpreted as a dependent relation between an algebra and a displayed algebra over it.

Sections (2)

$$
\begin{aligned}
&\mathsf{U}^{\mathsf{S}} \delta^S\, A\, A^D && :\equiv (x : A) \to A^D\, x \\
&(\underline{a})^{\mathsf{S}} \delta^S\, t\, t^D && :\equiv a^{\mathsf{S}}\, \delta^S\, t = t^D \\
&((x : a) \to B)^{\mathsf{S}} \delta^S\, f\, f^D && :\equiv \\
&\qquad (x : A^{\mathsf{A}}\, \delta) \to B^{\mathsf{S}}\, (\delta, x)\, (\delta^D, a^{\mathsf{S}}\, \delta^S\, x)\, (\delta^S, \mathsf{refl})\, (f\, x)\, (f^D\, x\, (a^{\mathsf{S}}\, \delta^S\, x)) \\
&\ldots
\end{aligned}
$$

For identity types, refl, path induction: we construct $n+1$-level paths by induction on n-level paths from induction hypotheses.

(See details in article/formalization)

Induction for HIITs

Now, for a $\Gamma \vdash \Delta$ signature and an algebra $\Gamma \vdash \mathit{initAlg} : \Delta^{\mathsf{A}}$, the type of induction is $(D : \Delta^{\mathsf{D}}\, \mathit{initAlg}) \to \Delta^{\mathsf{S}}\, \mathit{initAlg}\, D$.

Contents

- Homomorphisms of algebras are not homotopy sets in general, so we would need higher categories for HIIT semantics.

- Homomorphisms of algebras are not homotopy sets in general, so we would need higher categories for HIIT semantics.
- This is hard, so let's first assume uniqueness of identity proofs in the target theory, and develop semantics for QIITs using strict categories of algebras.

- Homomorphisms of algebras are not homotopy sets in general, so we would need higher categories for HIIT semantics.
- This is hard, so let's first assume uniqueness of identity proofs in the target theory, and develop semantics for QIITs using strict categories of algebras.
- This is WIP, but categorical semantics appears to work out nicely, and existence of initial algebras as well.

Thank you!

References I

Ahrens, B. and Lumsdaine, P. L. (2017).
Displayed categories.

Altenkirch, T. and Kaposi, A. (2016).
Type theory in type theory using quotient inductive types.
In Bodik, R. and Majumdar, R., editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 18–29. ACM.

Bernardy, J.-P., Jansson, P., and Paterson, R. (2012).
Proofs for free — parametricity for dependent types.
Journal of Functional Programming, 22(02):107–152.

Reynolds, J. C. (1983).
Types, abstraction and parametric polymorphism.
In Mason, R. E. A., editor, *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, September 19-23, 1983*, pages 513–523. Elsevier Science Publishers B. V. (North-Holland), Amsterdam.