3.36pt

On Inductive Types

Thorsten Altenkirch

Functional Programming Laboratory
School of Computer Science
University of Nottingham

February 18, 2020

Types of types

- Inductive Types
- Coinductive Types
- Universes

Simple inductive types

The type of ordinal notations $\Omega : \mathbf{Set}$ is given by the following constructors:

$$0 : \Omega$$
$$\mathrm{suc} : \Omega \to \Omega$$
$$\lim : (\mathbb{N} \to \Omega) \to \Omega$$

W-types

Given S : **Set** (shapes) and $P : S \to$ **Set** we define $\mathrm{W} : \mathit{Set}$ by the following constructor:

$$\mathrm{sup} : (s : S)(f : P\,s \to W) \to \mathrm{W}$$

All strictly positive simple inductive types can be reduced to W-types.

Example Ω

$$\begin{aligned} S &= \{\mathrm{zero}, \mathrm{suc}, \mathrm{lim}\} \\ P\,\mathrm{zero} &= 0 \\ P\,\mathrm{suc} &= 1 \\ P\,\mathrm{lim} &= \mathrm{Nat} \end{aligned}$$

W-types are the initial algebras of (non-dependent, unary) containers / polynomial functors.

References

Representing Inductively Defined Sets by Wellorderings in Martin-Löf's Type Theory

Peter Dybjer. TCS 1997

Categories of Containers

Michael Gordon Abbott, Thorsten Altenkirch, Neil Ghani . FoSSaCS 2003

Categories of Containers

Michael Gordon Abbott. PhD thesis. 2003

Containers: Constructing strictly positive types

Michael Gordon Abbott, Thorsten Altenkirch, Neil Ghani TCS 2005

Inductive Families

Given inductive defintions $\mathrm{Ty} : \mathit{Set}$ generated from $\circ : \mathrm{Ty}$ and $\_ \Rightarrow \_ : \mathrm{Ty} \to \mathrm{Ty} \to \mathrm{Ty}$ and $\mathrm{Con} : \mathbf{Set}$ with constructors $\bullet : \mathrm{Con}$ and $\_, \_ : \mathrm{Con} \to \mathrm{Ty} \to \mathrm{Con}$ we define

$$\mathrm{Tm} : \mathrm{Con} \to \mathrm{Ty} \to \mathbf{Set}$$

as given by the following constructors

$$\begin{aligned}
\mathrm{zero} &: (\Gamma : \mathrm{Con})(A : \mathrm{Ty}) \to \mathrm{Tm}\,(\Gamma, A)\,A \\
\mathrm{suc} &: (\Gamma : \mathrm{Con})(A\,B : \mathrm{Ty}) \to \mathrm{Tm}\,\Gamma\,A \to \mathrm{Tm}\,(\Gamma, B)\,A \\
\mathrm{app} &: (\Gamma : \mathrm{Con})(A\,B : \mathrm{Ty}) \to \mathrm{Tm}\,\Gamma\,(A \Rightarrow B) \to \mathrm{Tm}\,\Gamma\,A \to \mathrm{Tm}\,\Gamma\,B \\
\mathrm{lam} &: (\Gamma : \mathrm{Con})(A\,B : \mathrm{Ty}) \to \mathrm{Tm}\,(\Gamma, A)\,B \to \mathrm{Tm}\,\Gamma\,(A \Rightarrow B)
\end{aligned}$$

Indexed W-Types

Given

$$I : \mathbf{Set}$$
$$S : I \to Set$$
$$P : (i : I) \to S\, i \to I \to \mathbf{Set}$$

we define $\mathrm{WI} : I \to \mathbf{Set}$ as given by

$$\sup : (i : I)(s : S\, i)(f : (j : I) \to P\, i\, s\, j \to WI\, j) \to WI\, i$$

Inductive Families can be reduced to indexed W-types.

Indexed W-types are initial algebras of indexed containers / dependent polynomial functors.

Indexed W-types can be reduced to W-types

Given

$$I : \mathbf{Set}$$
$$S : I \to Set$$
$$P : (i : I) \to S\, i \to I \to \mathbf{Set}$$

we define

$$\bar{S} : \mathbf{Set}$$
$$\bar{S} = \Sigma i : I.Si$$
$$\bar{P} : \bar{S} \to \mathbf{Set}$$
$$\bar{P}\,(i, s) = \Sigma j : J.P\, i\, s\, j$$

$$W_0 = W\, \bar{S}\, \bar{P}$$
$$W_1 = W\, (I \times \bar{S})\, (\bar{P} \circ \pi_2)$$

$$\begin{aligned}
\mathrm{up} &: W_0 \to W_1 \\
\mathrm{down} &: W_0 \to I \to W_1 \\
\mathrm{up}\,(\mathrm{sup}\,(i,s)\,f) &= \mathrm{sup}\,((i,(i,s)),\mathrm{up}\circ f) \\
\mathrm{down}\,(\mathrm{sup}\,(i,s)\,f)\,j &= \mathrm{sup}\,(j,(i,s))\,\lambda(j,p).\mathrm{down}\,j\,(f\,(j,p))
\end{aligned}$$

$$\begin{aligned}
\mathrm{WI} &: I \to \mathbf{Set} \\
\mathrm{WI}\,i &= \Sigma w : W_0.\mathrm{up}\,w = \mathrm{down}\,i\,w
\end{aligned}$$

Needs UIP!

Observation by C.Sattler

$$\begin{aligned}
\mathrm{up} &: W_0 \to W_1 \\
\mathrm{down} &: W_0 \to I \to W_1 \\
\mathrm{up}\,(\mathrm{sup}\,(i,s)\,f) &= \mathrm{sup}\,((i,(i,s)), \mathrm{up} \circ f) \\
\mathrm{down}\,(\mathrm{sup}\,(i,s)\,f)\,j &= \mathrm{sup}\,(j,(i,s))\,\lambda(j,p).\mathrm{down}\,j\,(f\,(j,p))
\end{aligned}$$

$$\begin{aligned}
\mathrm{WI} &: I \to \mathbf{Set} \\
\mathrm{WI}\,i &= \Sigma w, w' : W_0.\mathrm{up}\,w = \mathrm{down}\,i\,w'
\end{aligned}$$

Correct without UIP.

References

Inductive Families

Peter Dybjer. Formal Asp. Comput. (1994)

Wellfounded Trees and Dependent Polynomial Functors

Nicola Gambino, Martin Hyland: TYPES 2003: 210-225

Indexed Containers

Thorsten Altenkirch, Peter Morris: LICS 2009: 277-285

Indexed containers

Thorsten Altenkirch, Neil Ghani, Peter Hancock, Conor McBride, Peter Morris: J. Funct. Program. 25 (2015)

Inductive-Inductive types

Mutual inductive types, where one type depends upon another.

Arise for example when defining dependently typed syntax.

$$\begin{aligned}
\mathrm{Con} &: \mathbf{Set} \\
\mathrm{Ty} &: \mathrm{Con} \to \mathbf{Set} \\
\\
\bullet &: \mathrm{Con} \\
_,_ &: (\Gamma : \mathrm{Con}) \to \mathrm{Ty}\,\Gamma \to \mathrm{Con} \\
\\
\mathrm{u} &: (\Gamma : \mathrm{Con}) \to \mathrm{Ty}\,\Gamma \\
\pi &: (\Gamma : \mathrm{Con})(A : \mathrm{Ty}\,\Gamma)(B : \mathrm{Ty}\,(\Gamma, A)) \to \mathrm{Ty}\,\Gamma
\end{aligned}$$

We can reduce Inductive-Inductive types to inductive families.

1. Preterms

$$\mathrm{Con}_0 : \mathbf{Set}$$
$$\mathrm{Ty}_0 : \mathbf{Set}$$

$$\bullet_0 : \mathrm{Con}_0$$
$$\_,_0\_ : \mathrm{Con}_0 \to \mathrm{Ty}_0 \to \mathrm{Con}_0$$

$$\mathrm{u}_0 : \mathrm{Con}_0 \to \mathrm{Ty}_0$$
$$\pi_0 : \mathrm{Con}_0 \to \mathrm{Ty}_0 \to \mathrm{Ty}_0 \to \mathrm{Ty}_0$$

2. Well-typedness

$$\mathrm{Con}_1 : \mathrm{Con}_0 \to \mathbf{Set}$$
$$\mathrm{Ty}_1 : \mathrm{Con}_0 \to \mathrm{Ty}_0 \to \mathbf{Set}$$

$$\bullet_1 : \mathrm{Con}_1\, \bullet_0$$
$$\_,_1\_ : \{\Gamma : \mathrm{Con}_0\} \to \mathrm{Con}_1\, \Gamma \to \{A : \mathrm{Ty}_0\} \to \mathrm{Ty}_1\, \Gamma\, A \to \mathrm{Con}_1\, (\Gamma ,_0 A)$$

$$\mathrm{u}_1 : \{\Gamma : \mathrm{Con}_0\} \to \mathrm{Con}_1 \Gamma \to \mathrm{Ty}_1\, (\mathrm{u}_0\, \Gamma)$$
$$\pi : \{\Gamma : \mathrm{Con}_0\} \to \mathrm{Con}_1\, \Gamma \to \{A : \mathrm{Ty}_0\} \to \mathrm{Ty}_1\, \Gamma\, A$$
$$\to \{B : \mathrm{Ty}_0\} \to \mathrm{Ty}_1\, (\Gamma ,_0 A)\, B \to \mathrm{Ty}_1\, \Gamma\, (\pi_1\, \Gamma\, A\, B)$$

3. The Σ-construction

$$\begin{aligned}
\mathrm{Con} &: \mathbf{Set} \\
\mathrm{Con} &= \Sigma \Gamma : \mathrm{Con}_0.\mathrm{Con}_1\ G_0 \\
\mathrm{Ty} &: \mathrm{Con} \to \mathbf{Set} \\
\mathrm{Ty}\,(\Gamma, \bar{\Gamma}) &= \Sigma A : \mathrm{Ty}_0.\mathrm{Ty}_1\ \Gamma\ A
\end{aligned}$$

We can derive all the constructors for the specified inductive-inductive types, e.g.

$$\begin{aligned}
_,_ &: (\Gamma : \mathrm{Con}) \to \mathrm{Ty}\,\Gamma \to \mathrm{Con} \\
(\Gamma, \bar{\Gamma}), (A, \bar{A}) &= (\Gamma,_0 A, \bar{\Gamma},_1 \bar{A})
\end{aligned}$$

Initiality

But how can we show initiality / derive the eliminator?

Given an algebra $C : \mathbf{Set}$, $T : C \to \mathbf{Set}$, ... we have to mutually define

$$
\begin{aligned}
f_{\mathrm{Con}} &: \mathrm{Con} \to C \\
&\cong (\Gamma : \mathrm{Con}_0) \to \mathrm{Con}_1\,\Gamma \to C \\
f_{\mathrm{Ty}} &: (\Gamma : \mathrm{Con}) \to \mathrm{Ty}\,\Gamma \to T\,(f_{\mathrm{Con}}\,\Gamma) \\
&\cong (A : \mathrm{Ty}_0)(\Gamma : \mathrm{Con}_0) \\
&\quad \to (\bar{\Gamma} : \mathrm{Con}_1\,\Gamma) \to \mathrm{Ty}_1\,\Gamma\,A \\
&\quad \to T\,(f_{\mathrm{Con}}\,\Gamma\,\bar{\Gamma})
\end{aligned}
$$

Which seems impossible to do directly.

Initiality using relations (A.Kovacs)

We inductively define the graph of the functions:

$$\mathrm{Con}_2 : \mathrm{Con}_0 \to C \to \mathbf{Set}$$
$$\mathrm{Ty}_2 : \mathrm{Ty}_0 \to (c : C) \to Tc \to Set$$

and show mutually (but non-dependently) :

$$(\Gamma : \mathrm{Con}_0) \to \mathrm{Con}_1\,\Gamma \to \mathrm{isContractible}\,(\Sigma c : C.\mathrm{Con}_2\,\Gamma\,c)$$
$$(A : \mathrm{Ty}_1)(\Gamma : \mathrm{Con}_0)$$
$$\to (\bar{\Gamma} : \mathrm{Con}_1\,\Gamma) \to \mathrm{Ty}_1\,\Gamma\,A$$
$$\to \mathrm{isContractible}\,(\Sigma c : C, t : Tc.\mathrm{Con}_2\,\Gamma\,c \times \mathrm{Ty}_2\,A\,c\,t)$$

where $\mathrm{isContractible}\,X = \Sigma x : X.(y : X) \to x = y$

From this we can extract f_{Con} and f_{Ty} and show initiality.

This construction requires UIP. We should be able to exploit Jasper Hugunin's construction to avoid this.

Reducing Inductive-Inductive Types

Can we do this in general?

What does this mean? What is a general notion of Inductive-Inductive Types?

There is no simple functorial semantics. We can't understand Inductive-Inductive types as an initial algebra of a functor.

References

Inductive-inductive definitions

Fredrik Nordvall Forsberg. PhD thesis. 2013.

Constructing Inductive-Inductive Types in Cubical Type Theory

Jasper Hugunin. FoSSaCS 2019:

For Induction-Induction, Induction is Enough

Ambrus Kaposi, Andras Kovacs. Ambroise Lafont. Submitted to TYPES 2019 postproceedings.

Higher Inductive Types, Inductive Families, and Inductive-Inductive Types

Jakob von Raumer. PhD thesis. 2019+

Quotient Inductive Types (QITs)

Given P : **Set** we define Tree : **Set** (permutable P-branching trees) inductively:

$$\begin{aligned}
\mathrm{leaf} &: \mathrm{Tree} \\
\mathrm{node} &: (P \to \mathrm{Tree}) \to \mathrm{Tree} \\
\mathrm{perm} &: (\pi : P \cong P)(f : P \to \mathrm{Tree}) \to \mathrm{node}\, f = \mathrm{node}\, (f \circ \pi)
\end{aligned}$$

QITs are a special case of HITs (Higher Inductive Types) in a truncated setting (with UIP).

Alternatively we can view them as HITs with a truncation constructor, e.g.

$$\mathrm{trunc} : (tu : \mathrm{Tree}\, P)(pq : t \equiv u) \to p \equiv q$$

We know that it is impossible to reduce QITs to W-types and quotients.

Combining inductive-inductive types and QITs lead to Quotient Inductive-Inductive Types (QIIT)

What is a QIIT exactly?

We define a universal QIIT: the theory of signatures.

We define the intrinsic syntax of type theory as a QIIT.

$$\begin{aligned}\mathrm{Con} &: \mathbf{Set}\\ \mathrm{Ty} &: \mathrm{Con} \to \mathbf{Set}\\ \mathrm{Tm} &: (\Gamma : \mathrm{Con}) \to \mathrm{Ty}\,\Gamma \to \mathbf{Set}\\ \mathrm{Tms} &: \mathrm{Con} \to \mathrm{Con} \to \mathbf{Set}\end{aligned}$$

such that the algebras correspond to categories with families (CwFs)

We add the following features:

A universe

$$\mathrm{u} : \{\Gamma : \mathrm{Con}\} \to \mathrm{Ty}\,\Gamma$$
$$\mathrm{el} : \{\Gamma : \mathrm{Con}\} \to \mathrm{Tm}\,\Gamma\,\mathrm{u} \to \mathrm{Ty}\,\Gamma$$

Π-types with small domains

$$\pi : \{\Gamma : \mathrm{Con}\} \to (a : \mathrm{Tm}\,\Gamma\,\mathrm{u}) \to (\mathrm{Ty}\,(\Gamma, \mathrm{el}\,a)) \to \mathrm{Ty}\,\Gamma$$

with app but no λ.

Equality types with small arguments

$$\mathrm{eq} : \{\Gamma : \mathrm{Con}\} \to \mathrm{Tm}\,\Gamma\,\mathrm{u} \to \mathrm{Tm}\,\Gamma\,\mathrm{u} \to \mathrm{Ty}\,\Gamma$$

with refl but no eliminator.

Contexts in this theory correspond to QIITs, e.g. we can define the natural numbers as the following context $\Gamma_{\mathbb{N}} : \mathrm{Con}$:

$$\mathrm{N} : \mathrm{u}, z : \mathrm{el}\,\mathrm{N}, s : \pi(x : \mathrm{N})(\mathrm{el}\,\mathrm{N})$$

To be able also to define types and constructors with external parameters (such as lists or vectors) we add:

Π-types over meta-level types

$$\Pi : \{\Gamma : \mathrm{Con}\}(X : \mathbf{Set}) \to (X \to \mathrm{Ty}\,\Gamma) \to \mathrm{Ty}\,\Gamma$$

Semantics of the theory of signatures

We can define the semantics of QIITs by induction over the syntax. I only give the types for $\Gamma : \mathrm{Con}$ here:

Algebras

$$\Gamma^A : \mathbf{Set}$$

$$\Gamma^A_{\mathbb{N}} = \Sigma N : \mathbf{Set}, z : N, s : N \to N$$

Algebra morphisms

$$\Gamma^M : \Gamma^A \to \Gamma^A \to \mathbf{Set}$$

$$\Gamma^M_{\mathbb{N}}\,(N, z, s)\,(N', z', s') = \Sigma f : N \to N', f\,z = z', (x : N) \to f\,(s\,x) = s'\,(f\,x)$$

Displayed algebras

$$\Gamma^D : \Gamma^A \to \mathbf{Set}$$

$$\Gamma^D_{\mathbb{N}}\,(N, z, s) =$$
$$\Sigma M : N \to \mathbf{Set}, M\,z, (x : N) \to M\,x \to M\,(s\,x)$$

Sections of displayed algebras

$$\Gamma^S : (A : \Gamma^A) \to \Gamma^D\,A \to \mathbf{Set}$$

$$\Gamma^S_{\mathbb{N}}\,(N, z, s)\,(N', z', s') =$$
$$\Sigma f : (x : N) \to N'\,x, f\,z = z',$$
$$(x : N)(x' : N'\,x) \to f\,(s\,x) = s'\,x\,x'$$

Initiality vs eliminator

We say an algebra $A : \Gamma^A$ is initial if for all algebras $X : \Gamma^A$ the type $\Gamma^M\, A\, X$ is contractible.

An algebra $A : \Gamma^A$ has an eliminator if for all displayed algebras $M : \Gamma^D\, A$ has a section $\Gamma^S\, A\, M$.

We can show that the two notions are equivalent.

Universality

We can actually construct initial algebras for all QIITs definable in the theory of signatures using the theory of signatures itself.

So for example we can construct the initial algebra for natural numbers:

$$\mu_{\mathbb{N}} : \Gamma^{A}_{\mathbb{N}}$$
$$\mu_{\mathbb{N}} = (\mathrm{Tm}\,\Gamma_{\mathbb{N}}\,N, z, \lambda t.s\,t)$$

Infinitely branching QIITs

However, we can only define finitely branching QIITs in the theory presented so far.

To construct infinitely branching QIITs we need to add

Small Π-types over external sets

$$\Pi : \{\Gamma : \mathrm{Con}\}(X : \mathbf{Set}) \to (X \to \mathrm{Tm}\,\Gamma\,\mathrm{u}) \to \mathrm{Tm}\,\Gamma\,\mathrm{u}$$

However, in this case our proof doesn't work (we need univalence and set truncation at the same time).

From QIITs to HIITs

Conservative 2-level type theory:
strict types $\mathbf{Type}^{\mathbf{s}}$ with a strict equality ($\cong$).
fibrant types **Type** with a univalent equality ($=$).

We define the theory of codes as an $\mathbf{Type}^{\mathbf{s}}$ QIIT.
The interpretation Γ^A etc maps into fibrant types.
However, we cannot prove the equivalence of induction and elimination because it relies on truncation.
Current work (C.Sattler): use complete Segal types (in a semisimplicial setting). The universe is interpreted by left fibrations.

References

Type theory in type theory using quotient inductive types

Thorsten Altenkirch, Ambrus Kaposi. POPL 2016

Extending Homotopy Type Theory with Strict Equality

Thorsten Altenkirch, Paolo Capriotti, Nicolai Kraus. CSL 2016

Quotient Inductive-Inductive Types

Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, Fredrik Nordvall Forsberg. FoSSaCS 2018

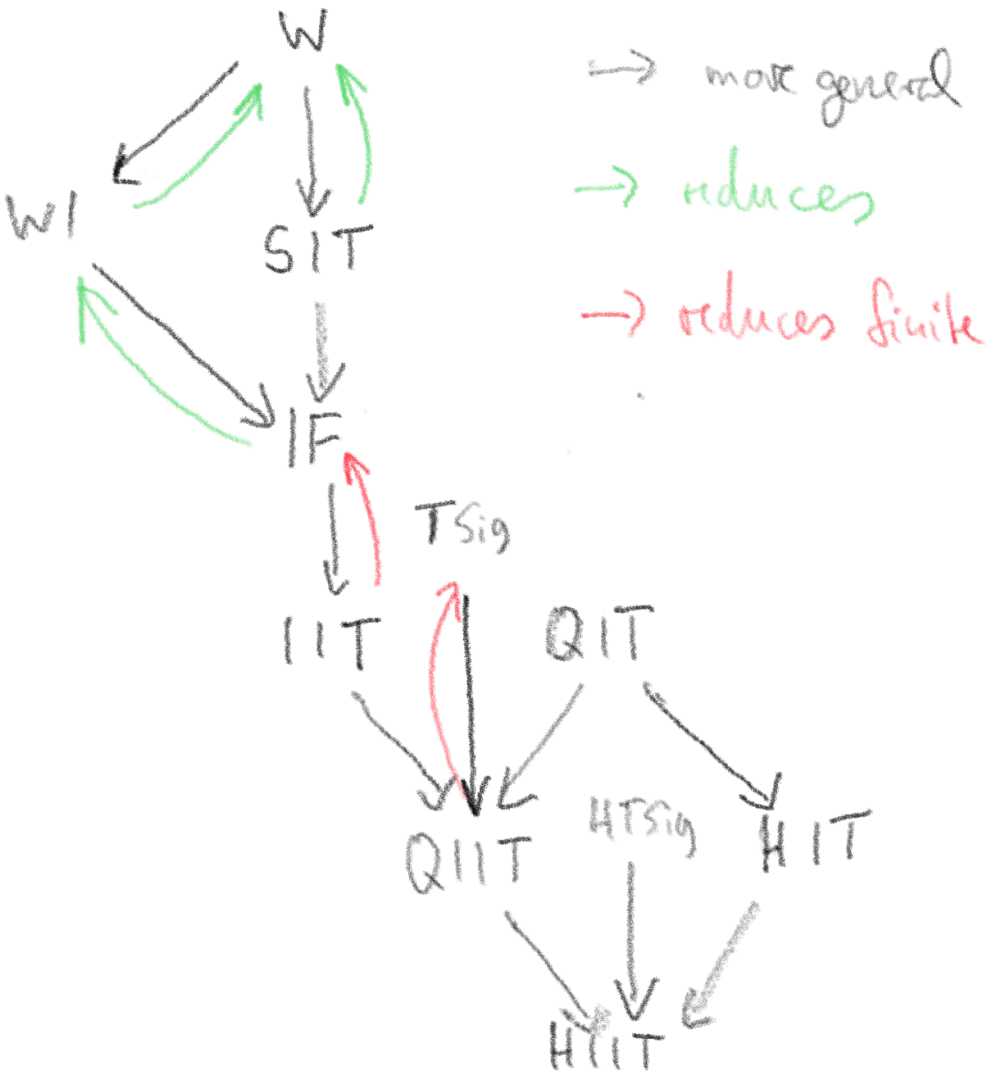
A Syntax for Higher Inductive-Inductive Types

Ambrus Kaposi, András Kovács. FSCD 2018

Constructing quotient inductive-inductive types

Ambrus Kaposi, András Kovács, Thorsten Altenkirch. POPL 2019

A map of inductive types



Open questions

- Reduce infinitary inductive-inductive types to W-types.
- Define a universal QIIT with infinitary branching trees.
  (A. Kovacs is working on this)
- Extend the results for QIITs to HIITs.
- Find simple universal types for QITs (QW) and HITs (HW).