Specifying higher inductive inductive types (HIITs)

Ambrus Kaposi
Eötvös Loránd University, Budapest

j.w.w. András Kovács and Thorsten Altenkirch

Université de Nantes
25 April 2015

Contents

Different classes of inductive types by examples

An inductive type

- Type formation:

$$\mathsf{Nat} : \mathsf{Type}$$

- Constructors:

$$\begin{aligned} \mathsf{zero} &: \mathsf{Nat} \\ \mathsf{suc} &: \mathsf{Nat} \to \mathsf{Nat} \end{aligned}$$

- Eliminator:

$$\begin{aligned} \mathsf{ElimNat} : \ & (P : \mathsf{Nat} \to \mathsf{Type}) \\ & (pz : P\, \mathsf{zero}) \\ & (ps : (n : \mathsf{Nat}) \to P\, n \to P\, (\mathsf{suc}\, n)) \\ & (n : \mathsf{Nat}) \to P\, n \end{aligned}$$

- Computation rules:

$$\begin{aligned} \mathsf{ElimNat}\, P\, pz\, ps\, \mathsf{zero} &\equiv pz \\ \mathsf{ElimNat}\, P\, pz\, ps\, (\mathsf{suc}\, n) &\equiv ps\, n\, (\mathsf{ElimNat}\, P\, pz\, ps\, n) \end{aligned}$$

Using the eliminator

- Addition:

$$(m : \mathsf{Nat}) + (n : \mathsf{Nat}) : \mathsf{Nat} :\equiv \mathsf{ElimNat}\,(\lambda x.\mathsf{Nat})\, n\, (\lambda x\, w.\mathsf{suc}\, w)\, m$$

- From the computation rules we get:

$$\begin{aligned} \mathsf{zero} \quad &+ n \equiv n \\ \mathsf{suc}\, m\ &+ n \equiv \mathsf{suc}\,(m+n) \end{aligned}$$

- Associativity of addition:

$$\begin{aligned} &\mathit{asssoc}\,(m\, n\, o : \mathsf{Nat}) : (m+n)+o =_{\mathsf{Nat}} m+(n+o) \\ &:\equiv\ \mathsf{ElimNat}\,(\lambda x.(x+n)+o =_{\mathsf{Nat}} x+(n+o)) \\ &\qquad\qquad \mathsf{refl}_{n+o} \\ &\qquad\qquad (\lambda x\, w.\mathsf{ap}\,\mathsf{suc}\, w) \\ &\qquad\qquad m \end{aligned}$$

An indexed inductive type

- Type formation:

$$\mathrm{Vec}_A : \mathrm{Nat} \to \mathrm{Type}$$

- Constructors:

$$\begin{aligned} \mathrm{nil} &: \mathrm{Vec}_A\,\mathrm{zero} \\ \mathrm{cons} &: A \to \mathrm{Vec}_A\, n \to \mathrm{Vec}_A\,(\mathrm{suc}\, n) \end{aligned}$$

- Eliminator:

$$\begin{aligned} \mathrm{ElimVec} : {} & (P : (n : \mathrm{Nat}) \to \mathrm{Vec}_A\, n \to \mathrm{Type}) \\ & (pnil : P\,\mathrm{zero}\,\mathrm{nil})(pcons : \ldots) \\ & (n : \mathrm{Nat})(xs : \mathrm{Vec}_A\, n) \to P\, n\, xs \end{aligned}$$

- Computation rules

Mutual inductive types

- Type formations:

$$\begin{aligned} \mathsf{IsEven} &: \mathsf{Nat} \to \mathsf{Type} \\ \mathsf{IsOdd} &: \mathsf{Nat} \to \mathsf{Type} \end{aligned}$$

- Constructors:

$$\begin{aligned} \mathsf{zeroEven} &: \mathsf{IsEven}\,\mathsf{zero} \\ \mathsf{sucOdd} &: (n : \mathsf{Nat}) \to \mathsf{IsOdd}\, n \to \mathsf{IsEven}\,(\mathsf{suc}\, n) \\ \mathsf{sucEven} &: (n : \mathsf{Nat}) \to \mathsf{IsEven}\, n \to \mathsf{IsOdd}\,(\mathsf{suc}\, n) \end{aligned}$$

- Eliminators
- Computation rules

An inductive-inductive type

- Type formations:

$$\begin{aligned}
\mathsf{Con} &: \mathsf{Type} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Type}
\end{aligned}$$

- Constructors:

$$\begin{aligned}
\bullet &: \mathsf{Con} \\
- \triangleright - &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
\mathsf{U} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \\
\Pi &: (\Gamma : \mathsf{Con})(A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma \triangleright A) \to \mathsf{Ty}\,\Gamma
\end{aligned}$$

- Eliminators:

$$\begin{aligned}
\mathsf{ElimCon} &: (P : \mathsf{Con} \to \mathsf{Type})(Q : P\,\Gamma \to \mathsf{Ty}\,\Gamma \to \mathsf{Type}) \\
&\quad \to \ldots \to (\Gamma : \mathsf{Con}) \to P\,\Gamma \\
\mathsf{ElimTy} &: (P : \ldots)(Q : \ldots) \to \ldots \to (A : \mathsf{Ty}\,\Gamma) \to Q\,(\mathsf{ElimCon}\,\Gamma)\,A
\end{aligned}$$

- Computation rules

Another inductive-inductive type

- Type formations:

$$\begin{aligned} \mathsf{SortedList} \quad &: \mathsf{Type} \\ - \leq_{\mathsf{SortedList}} - &: \mathbb{N} \to \mathsf{SortedList} \to \mathsf{Type} \end{aligned}$$

- Constructors:

$$\begin{aligned} \mathsf{nil} \quad &: \mathsf{SortedList} \\ \mathsf{cons} \quad &: (x : \mathbb{N})(xs : \mathsf{SortedList}) \to x \leq_{\mathsf{SortedList}} xs \to \mathsf{SortedList} \\ \mathsf{nil}_{\leq} \quad &: x \leq_{\mathsf{SortedList}} \mathsf{nil} \\ \mathsf{cons}_{\leq} &: y \leq x \to (p : x \leq_{\mathsf{SortedList}} xs) \to y \leq_{\mathsf{SortedList}} \mathsf{cons}\, x\, xs\, p \end{aligned}$$

- Eliminators
- Computation rules

A higher inductive type

- Type formation:

$$\mathsf{Int} : \mathsf{Type}$$

- Constructors:

$$\begin{aligned}
&\mathsf{pair} &&: \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Int}\\
&\mathsf{eq} &&: (a\, b\, c\, d : \mathsf{Nat}) \to a + d =_{\mathsf{Nat}} b + c \to \mathsf{pair}\, a\, b =_{\mathsf{Int}} \mathsf{pair}\, c\, d\\
&\mathsf{trunc} &&: (x\, y : \mathsf{Int})(p\, q : x =_{\mathsf{Int}} y) \to p =_{x =_{\mathsf{Int}} y} q
\end{aligned}$$

- Eliminator

$$\begin{aligned}
\mathsf{ElimInt} : &(P : \mathsf{Int} \to \mathsf{Type})\big(p : (a\, b : \mathsf{Nat}) \to P\,(\mathsf{pair}\, a\, b)\big)\\
&(e : \text{make sure that } p \text{ respects the eq}) \to \ldots \to (i : \mathsf{Int}) \to P\, i
\end{aligned}$$

- Computation rules

A higher inductive-inductive type (HIIT)

Cauchy reals (we assume $\mathbb{Q}$, $+$, $-$, $<$). Type formations:

$$\begin{aligned} \mathbb{R} &: \mathsf{Type} \\ \sim &: \mathbb{Q} \to \mathbb{R} \to \mathbb{R} \to \mathsf{Type} \end{aligned}$$

Constructors:

$$\begin{aligned}
&\mathsf{rat} &&: \mathbb{Q} \to \mathbb{R} \\
&\mathsf{lim} &&: (x : \mathbb{Q}_+ \to \mathbb{R}) \to ((\delta\, \epsilon : \mathbb{Q}_+) \to x_\delta \sim_{\delta+\epsilon} x_\epsilon) \to \mathbb{R} \\
&\mathsf{eq} &&: (u\, v : \mathbb{R}) \to ((\epsilon : \mathbb{Q}_+) \to u \sim_\epsilon v) \to u =_{\mathbb{R}} v \\
&\mathsf{ratrat} &&: (q\, r : \mathbb{Q})(\epsilon : \mathbb{Q}_+) \to -\epsilon < q - r < \epsilon \to \mathsf{rat}\, q \sim_\epsilon \mathsf{rat}\, r \\
&\mathsf{ratlim} &&: (q : \mathbb{Q})(\epsilon\, \delta : \mathbb{Q}_+)(y : \mathbb{Q} \to \mathbb{R})(py : (\delta_1\, \epsilon_1 : \mathbb{Q}_+) \to y_{\delta_1} \sim_{\delta_1+\epsilon_1} y_{\epsilon_1}) \\
& &&\to \mathsf{rat}\, q \sim_{\epsilon-\delta} y_\delta \to \mathsf{rat}\, q \sim_\epsilon \mathsf{lim}\, y\, py \\
&\mathsf{limrat} &&: \ldots \\
&\mathsf{limlim} &&: (x\, y : \mathbb{Q} \to \mathbb{R})(px : \ldots)(py : \ldots)(\epsilon\, \delta\, \eta : \mathbb{Q}_+) \\
& &&\to x_\delta \sim_{\epsilon-\delta-\eta} y_\eta \to \mathsf{lim}\, x\, px \sim_\epsilon \mathsf{lim}\, y\, py \\
&\mathsf{trunc} &&: (u\, v : \mathbb{R})(\epsilon : \mathbb{Q}_+)(pq : u \sim_\epsilon v) \to p =_{u \sim_\epsilon v} q
\end{aligned}$$

Another HIIT

The well-typed syntax of type theory quotiented by conversion. Type formations:

$$
\begin{aligned}
\mathsf{Con} &: \mathsf{Type} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Type} \\
\mathsf{Tm} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Type} \\
\mathsf{Tms} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type}
\end{aligned}
$$

Constructors (truncation constructors omitted):

$$
\begin{array}{llll}
\cdot & : \mathsf{Con} & [\mathsf{id}] & : A[\mathsf{id}] = A \\
- \rhd - & : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} & [\circ] & : A[\sigma \circ \delta] = A[\sigma][\delta] \\
-[-] & : \mathsf{Ty}\,\Delta \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma & \mathsf{ass} & : (\sigma \circ \delta) \circ \nu = \sigma \circ (\delta \circ \nu) \\
\mathsf{id} & : \mathsf{Tms}\,\Gamma\,\Gamma & \mathsf{idl} & : \mathsf{id} \circ \sigma = \sigma \\
- \circ - & : \mathsf{Tms}\,\Theta\,\Delta \to \mathsf{Tms}\,\Gamma\,\Theta \to \mathsf{Tms}\,\Gamma\,\Delta & \mathsf{idr} & : \sigma \circ \mathsf{id} = \sigma \\
\epsilon & : \mathsf{Tms}\,\Gamma\,\cdot & \cdot\eta & : \{\sigma : \mathsf{Tms}\,\Gamma\,\cdot\} \to \sigma = \epsilon \\
-,- & : (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]) \to \mathsf{Tms}\,\Gamma\,(\Delta \rhd A) & \rhd\beta_1 & : \pi_1\,(\sigma, t) = \sigma \\
\pi_1 & : \mathsf{Tms}\,\Gamma\,(\Delta \rhd A) \to \mathsf{Tms}\,\Gamma\,\Delta & \rhd\beta_2 & : \pi_2\,(\sigma, t) = t \\
\pi_2 & : (\sigma : \mathsf{Tms}\,\Gamma\,(\Delta \rhd A)) \to \mathsf{Tm}\,\Gamma\,(A[\pi_1\,\sigma]) & \rhd\eta & : (\pi_1\,\sigma, \pi_2\,\sigma) = \sigma \\
-[-] & : \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]) & \rhd\mathsf{nat} & : (\sigma, t) \circ \delta = (\sigma \circ \delta, t[\delta]) \\
\hline
\mathsf{U} & : \mathsf{Ty}\,\Gamma & \mathsf{U}[] & : \mathsf{U}[\sigma] = \mathsf{U} \\
\mathsf{El} & : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma & \mathsf{El}[] & : (\mathsf{El}\,a)[\sigma] = \mathsf{El}\,(a[\sigma]) \\
\hline
\Pi & : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma \rhd A) \to \mathsf{Ty}\,\Gamma & \Pi[] & : (\Pi\,A\,B)[\sigma] = \Pi\,(A[\sigma])\,(B[\sigma^{\uparrow}]) \\
\mathsf{lam} & : \mathsf{Tm}\,(\Gamma \rhd A)\,B \to \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) & \mathsf{lam}[] & : (\mathsf{lam}\,t)[\sigma] = \mathsf{lam}\,(t[\sigma^{\uparrow}]) \\
\mathsf{app} & : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \to \mathsf{Tm}\,(\Gamma \rhd A)\,B & \Pi\beta & : \mathsf{app}\,(\mathsf{lam}\,t) = t \\
 & & \Pi\eta & : \mathsf{lam}\,(\mathsf{app}\,t) = t
\end{array}
$$

What is a specification?

Specifications for different classes of inductive types

Inductive types	W-types
Indexed inductive types	Indexed W-types
Mutual inductive types	reduced to indexed W-types
Inductive-inductive types	Fredrik Forsberg's, 10 pages
Higher inductive types (subsets)	Sojakova, Basold-Geuvers-Weide, Shulman-Lumsdaine, Dybjer-Moeneclaey
Quotient inductive-inductive types	Altenkirch-Capriotti-Dijkstra-Forsberg
Higher inductive-inductive types	???

What is a specification?

A coding scheme.

Codes	Code	$: \mathsf{Type}$
Algebras	$-^{\mathsf{A}}$	$: \mathsf{Code} \to \mathsf{Type}$
Families	$-^{\mathsf{F}}$	$: (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}} \to \mathsf{Type}$
Sections	$-^{\mathsf{S}}$	$: (\Gamma : \mathsf{Code})(\gamma : \Gamma^{\mathsf{A}}) \to \Gamma^{\mathsf{F}}\,\gamma \to \mathsf{Type}$
Existence	con	$: (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}}$
	elim	$: (\Gamma : \mathsf{Code})(m : \Gamma^{\mathsf{F}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\mathsf{con}\,\Gamma)\,m$

Results of an example specification

We assume that we have a type of codes Code and Γ is encoding Nat.

$$
\begin{array}{lll}
\text{Codes} & \Gamma & : \mathsf{Code} \\
\text{Algebras} & -^{\mathsf{A}} & : \mathsf{Code} \to \mathsf{Type} \\
\text{Families} & -^{\mathsf{F}} & : (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}} \to \mathsf{Type} \\
 & & \\
\text{Sections} & -^{\mathsf{S}} & : (\Gamma : \mathsf{Code})(\gamma : \Gamma^{\mathsf{A}}) \to \Gamma^{\mathsf{F}}\,\gamma \to \mathsf{Type} \\
 & & \\
\text{Existence} & \mathsf{con} & : (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}} \\
 & \mathsf{elim} & : (\Gamma : \mathsf{Code})(m : \Gamma^{\mathsf{F}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\mathsf{con}\,\Gamma)\,m
\end{array}
$$

Results of an example specification

We assume that we have a type of codes Code and Γ is encoding Nat.

$$
\begin{array}{lll}
\text{Codes} & \Gamma & : \mathsf{Code} \\
\text{Algebras} & \Gamma^{\mathsf{A}} & \equiv (N : \mathsf{Type}) \times N \times (N \to N) \\
\text{Families} & -^{\mathsf{F}} & : (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}} \to \mathsf{Type} \\
 & & \\
\text{Sections} & -^{\mathsf{S}} & : (\Gamma : \mathsf{Code})(\gamma : \Gamma^{\mathsf{A}}) \to \Gamma^{\mathsf{F}}\,\gamma \to \mathsf{Type} \\
 & & \\
\text{Existence} & \mathsf{con} & : (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}} \\
 & \mathsf{elim} & : (\Gamma : \mathsf{Code})(m : \Gamma^{\mathsf{F}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\mathsf{con}\,\Gamma)\,m
\end{array}
$$

Results of an example specification

We assume that we have a type of codes Code and Γ is encoding Nat.

$$
\begin{array}{lll}
\text{Codes} & \Gamma & : \mathsf{Code} \\
\text{Algebras} & \Gamma^{\mathsf{A}} & \equiv (N : \mathsf{Type}) \times N \times (N \to N) \\
\text{Families} & \Gamma^{\mathsf{F}} & (N, z, s) \equiv (P : N \to \mathsf{Type}) \times P\,z \\
 & & \qquad \times \big((n : N) \to P\,n \to P\,(s\,n)\big) \\
\text{Sections} & -^{\mathsf{S}} & : (\Gamma : \mathsf{Code})(\gamma : \Gamma^{\mathsf{A}}) \to \Gamma^{\mathsf{F}}\,\gamma \to \mathsf{Type}
\end{array}
$$

$$
\begin{array}{lll}
\text{Existence} & \mathsf{con} & : (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}} \\
 & \mathsf{elim} & : (\Gamma : \mathsf{Code})(m : \Gamma^{\mathsf{F}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\mathsf{con}\,\Gamma)\,m
\end{array}
$$

Results of an example specification

We assume that we have a type of codes Code and Γ is encoding Nat.

Codes	Γ	$: \mathsf{Code}$
Algebras	Γ^{A}	$\equiv (N : \mathsf{Type}) \times N \times (N \to N)$
Families	Γ^{F}	$(N, z, s) \equiv (P : N \to \mathsf{Type}) \times P\,z \times \big((n : N) \to P\,n \to P\,(s\,n)\big)$
Sections	Γ^{S}	$(N, z, s)\,(P, pz, ps) \equiv (f : (n : N) \to P\,n) \times (f\,z = pz) \times \big((n : N) \to f\,(s\,n) = ps\,n\,(f\,n)\big)$
Existence	con	$: (\Gamma : \mathsf{Code}) \to \Gamma^{\mathsf{A}}$
	elim	$: (\Gamma : \mathsf{Code})(m : \Gamma^{\mathsf{F}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\mathsf{con}\,\Gamma)\,m$

Results of an example specification

We assume that we have a type of codes Code and Γ is encoding Nat.

$$
\begin{array}{lll}
\text{Codes} & \Gamma & : \mathsf{Code} \\
\text{Algebras} & \Gamma^{\mathsf{A}} & \equiv (N : \mathsf{Type}) \times N \times (N \to N) \\
\text{Families} & \Gamma^{\mathsf{F}} & (N, z, s) \equiv (P : N \to \mathsf{Type}) \times P\,z \\
& & \qquad \times \big((n : N) \to P\,n \to P\,(s\,n)\big) \\
\text{Sections} & \Gamma^{\mathsf{S}} & (N, z, s)\,(P, pz, ps) \equiv (f : (n : N) \to P\,n) \\
& & \qquad \times (f\,z = pz) \times \big((n : N) \to f\,(s\,n) = ps\,n\,(f\,n)\big) \\
\text{Existence} & \multicolumn{2}{l}{\mathsf{con}\ \ \Gamma \equiv (\mathsf{Nat}, \mathsf{zero}, \mathsf{suc})} \\
& \multicolumn{2}{l}{\mathsf{elim}\ : (\Gamma : \mathsf{Code})(m : \Gamma^{\mathsf{F}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\mathsf{con}\,\Gamma)\,m}
\end{array}
$$

Results of an example specification

We assume that we have a type of codes Code and Γ is encoding Nat.

$$
\begin{array}{lll}
\text{Codes} & \Gamma & : \mathsf{Code} \\
\text{Algebras} & \Gamma^{\mathsf{A}} & \equiv (N : \mathsf{Type}) \times N \times (N \to N) \\
\text{Families} & \Gamma^{\mathsf{F}} & (N, z, s) \equiv (P : N \to \mathsf{Type}) \times P\, z \\
 & & \qquad \times \big((n : N) \to P\, n \to P\,(s\, n)\big) \\
\text{Sections} & \Gamma^{\mathsf{S}} & (N, z, s)\,(P, pz, ps) \equiv (f : (n : N) \to P\, n) \\
 & & \qquad \times (f\, z = pz) \times \big((n : N) \to f\,(s\, n) = ps\, n\,(f\, n)\big) \\
\text{Existence} & \multicolumn{2}{l}{\mathsf{con}\ \Gamma \equiv (\mathsf{Nat}, \mathsf{zero}, \mathsf{suc})} \\
 & \multicolumn{2}{l}{\mathsf{elim}\,\Gamma\,(P, pz, ps) \equiv (\mathsf{ElimNat}\, P\, pz\, ps, \mathsf{refl}, \lambda n.\mathsf{refl})}
\end{array}
$$

W-types

A coding scheme.

Codes $\text{Code} :\equiv (S : \text{Type}) \times (S \to \text{Type})$

Algebras $(S,P)^{\mathsf{A}} :\equiv (W : \text{Type}) \times \big((s : S) \to (P\,s \to W) \to W\big)$

Families $(S,P)^{\mathsf{F}}\,(W, sup) :\equiv (W^M : W \to \text{Type}) \times$
$\big((s : S)(f : P\,s \to W) \to (\forall p. W^M\,(f\ p)) \to W^M\,(sup\,s\,f)\big)$

Sections $(S,P)^{\mathsf{S}}\,(W, sup)\,(W^M, sup^M) :\equiv (W^E : (w : W) \to W^M\,w)$
$\times \big(\forall s\,f. W^E\,(sup\,s\,f) = sup^M\,s\,f\,(\lambda p. W^E\,(f\ p))\big)$

Existence $\text{con} : (\Gamma : \text{Code}) \to \Gamma^{\mathsf{A}}$
$\text{elim} : (\Gamma : \text{Code})(m : \Gamma^{\mathsf{F}}\,(\text{con}\,\Gamma)) \to \Gamma^{\mathsf{S}}\,(\text{con}\,\Gamma)\,m$

A specification of HIITs

Our coding scheme for HIITs

A code is a context.

Our coding scheme for HIITs

A code is a context.

For example, Nat is encoded by

$$(\cdot,\ \mathit{Nat} : \mathsf{U},\ \mathit{zero} : \mathit{Nat},\ \mathit{suc} : \mathit{Nat} \to \mathit{Nat}).$$

(Nat, zero and suc are variable names.)

Our coding scheme for HIITs

A code is a context.
For example, Nat is encoded by

$$(\cdot,\ \mathit{Nat} : \mathsf{U},\ \mathit{zero} : \mathit{Nat},\ \mathit{suc} : \mathit{Nat} \to \mathit{Nat}).$$

(Nat, zero and suc are variable names.)
The well-typed syntax of type theory quotiented by conversion is

$$\begin{aligned}
\big(&\cdot,\ \mathit{Con} : \mathsf{U},\ \mathit{Ty} : \mathit{Con} \to \mathsf{U},\ \mathit{Tm} : (\Gamma : \mathit{Con}) \to \mathit{Ty}\,\Gamma \to \mathsf{U} \\
&,\ \mathit{Tms} : \mathit{Con} \to \mathit{Con} \to \mathsf{U},\ \bullet : \mathit{Con},\ -\triangleright- : (\Gamma : \mathit{Con}) \to \mathit{Ty}\,\Gamma \to \mathit{Con} \\
&,\ -[-] : \mathit{Ty}\,\Delta \to \mathit{Tms}\,\Gamma\,\Delta \to \mathit{Ty}\,\Gamma,\ \mathit{id} : \mathit{Tms}\,\Gamma\,\Gamma \\
&,\ - \circ - : \mathit{Tms}\,\Theta\,\Delta \to \mathit{Tms}\,\Gamma\,\Theta \to \mathit{Tms}\,\Gamma\,\Delta \\
&,\ \dots \\
&,\ \Pi\eta : \mathit{lam}\,(\mathit{app}\,t) = t \\
\big)&.
\end{aligned}$$

A domain-specific type theory (DSTT)

Variables: $\dfrac{}{\vdash \cdot}$ $\dfrac{\Gamma \vdash A}{\vdash \Gamma, x : A}$ $\dfrac{\Gamma \vdash A}{\Gamma, x : A \vdash x : A}$ $\dfrac{\Gamma \vdash x : A \quad \Gamma \vdash B}{\Gamma, y : B \vdash x : A}$

Universe: $\dfrac{}{\Gamma \vdash \mathsf{U}}$ $\dfrac{\Gamma \vdash a : \mathsf{U}}{\Gamma \vdash \underline{a}}$

Inductive params: $\dfrac{\Gamma \vdash a : \mathsf{U} \quad \Gamma, x : \underline{a} \vdash B}{\Gamma \vdash (x : a) \to B}$ $\dfrac{\Gamma \vdash t : (x : a) \to B \quad \Gamma \vdash u : \underline{a}}{\Gamma \vdash t\, u : B[x \mapsto u]}$

Non-inductive params: $\dfrac{T : \mathsf{Type} \quad \forall(\alpha : T).\Gamma \vdash B_\alpha}{\Gamma \vdash (\alpha : T) \to B_\alpha}$ $\dfrac{\Gamma \vdash t : (\alpha : T) \to B_\alpha \quad \alpha' : T}{\Gamma \vdash t\, \alpha' : B_{\alpha'}}$

Equality: $\dfrac{\Gamma \vdash a : \mathsf{U} \quad \Gamma \vdash t, u : \underline{a}}{\Gamma \vdash t =_a u : \mathsf{U}}$ $\dfrac{\Gamma \vdash t : \underline{a}}{\Gamma \vdash \mathsf{refl} : \underline{t =_a t}}$ small J with propositional β

Infinitary params: $\dfrac{T : \mathsf{Type} \quad \forall(\alpha : T).\Gamma \vdash b_\alpha : \mathsf{U}}{\Gamma \vdash (\alpha : T) \to b_\alpha : \mathsf{U}}$ $\dfrac{\Gamma \vdash t : \underline{(\alpha : T) \to b_\alpha} \quad \alpha' : T}{\Gamma \vdash t\, \alpha' : \underline{b_{\alpha'}}}$

Algebras: standard model

Specification:

$$\frac{\Gamma \vdash}{\Gamma^{\mathsf{A}} : \mathsf{Type}} \qquad \frac{\Gamma \vdash A}{A^{\mathsf{A}} : \Gamma^{\mathsf{A}} \to \mathsf{Type}}$$

$$\frac{\Gamma \vdash t : A}{t^{\mathsf{A}} : (\gamma : \Gamma^{\mathsf{A}}) \to A^{\mathsf{A}}\,\gamma}$$

Implementation:

$$
\begin{array}{ll}
\cdot^{\mathsf{A}} & :\equiv \top \\
(\Gamma, x : A)^{\mathsf{A}} & :\equiv (\gamma : \Gamma^{\mathsf{A}}) \times A^{\mathsf{A}}\,\gamma \\
x^{\mathsf{A}}\,\gamma & :\equiv x^{\text{th}} \text{ component in } \gamma \\
\mathsf{U}^{\mathsf{A}}\,\gamma & :\equiv \mathsf{Type} \\
\underline{a}^{\mathsf{A}}\,\gamma & :\equiv a^{\mathsf{A}}\,\gamma \\
((x : a) \to B)^{\mathsf{A}}\,\gamma & :\equiv (\alpha : a^{\mathsf{A}}\,\gamma) \to B^{\mathsf{A}}\,(\gamma, \alpha)
\end{array}
$$

...

Families: logical predicate interpretation

Specification:

$$\frac{\Gamma \vdash}{\Gamma^{\mathsf{F}} : \Gamma^{\mathsf{A}} \to \mathsf{Type}} \qquad \frac{\Gamma \vdash A}{A^{\mathsf{F}} : \Gamma^{\mathsf{F}}\,\gamma \to A^{\mathsf{A}}\,\gamma \to \mathsf{Type}}$$

$$\frac{\Gamma \vdash t : A}{t^{\mathsf{F}} : (\gamma_F : \Gamma^{\mathsf{F}}\,\gamma) \to A^{\mathsf{F}}\,\gamma_F\,(t^{\mathsf{A}}\,\gamma)}$$

Implementation:

$$\begin{aligned}
\cdot^{\mathsf{F}}\,\mathsf{tt} &:\equiv \top \\
(\Gamma, x : A)^{\mathsf{F}}\,(\gamma, \alpha) &:\equiv (\gamma_F : \Gamma^{\mathsf{F}}\,\gamma) \times A^{\mathsf{F}}\,\gamma_F\,\alpha \\
x^{\mathsf{F}}\,\gamma_F &:\equiv x^{\mathsf{th}} \text{ component in } \gamma_F \\
\mathsf{U}^{\mathsf{F}}\,\gamma_F\,a &:\equiv a \to \mathsf{Type} \\
\underline{a}^{\mathsf{F}}\,\gamma_F\,\alpha &:\equiv a^{\mathsf{F}}\,\gamma_F\,\alpha \\
((x : a) \to B)^{\mathsf{F}}\,\gamma_F\,f &:\equiv (\alpha_F : a^{\mathsf{F}}\,\gamma_F\,\alpha) \to B^{\mathsf{F}}\,(\gamma_F, \alpha_F)\,(f\,\alpha)
\end{aligned}$$

...

The logical relation interpretation

Specification:

$$\frac{\Gamma \vdash}{\Gamma^{\mathsf{R}} : \Gamma^{\mathsf{A}} \to \Gamma^{\mathsf{A}} \to \mathsf{Type}} \qquad \frac{\Gamma \vdash A}{A^{\mathsf{R}} : \Gamma^{\mathsf{R}}\,\gamma_0\,\gamma_1 \to A^{\mathsf{A}}\,\gamma_0 \to A^{\mathsf{A}}\,\gamma_1 \to \mathsf{Type}}$$

$$\frac{\Gamma \vdash t : A}{t^{\mathsf{R}} : (\gamma_R : \Gamma^{\mathsf{R}}\,\gamma_0\,\gamma_1) \to A^{\mathsf{R}}\,\gamma_R\,(t^{\mathsf{A}}\,\gamma_0)\,(t^{\mathsf{A}}\,\gamma_1)}$$

Implementation:

$$\begin{aligned}
\cdot^{\mathsf{R}}\,\mathsf{tt}\,\mathsf{tt} &:\equiv \top \\
(\Gamma, x : A)^{\mathsf{R}}\,(\gamma_0, \alpha_0)\,(\gamma_1, \alpha_1) &:\equiv (\gamma_R : \Gamma^{\mathsf{R}}\,\gamma_0\,\gamma_1) \times A^{\mathsf{R}}\,\gamma_R\,\alpha_0\,\alpha_1 \\
x^{\mathsf{R}}\,\gamma_R &:\equiv x^{\mathsf{th}} \text{ component in } \gamma_R \\
\mathsf{U}^{\mathsf{R}}\,\gamma_R\,a_0\,a_1 &:\equiv a_0 \to a_1 \to \mathsf{Type} \\
\underline{a}^{\mathsf{R}}\,\gamma_R\,\alpha_0\,\alpha_1 &:\equiv a^{\mathsf{R}}\,\gamma_R\,\alpha_0\,\alpha_1 \\
((x : a) \to B)^{\mathsf{R}}\,\gamma_R\,f_0\,f_1 &:\equiv (\alpha_R : a^{\mathsf{R}}\,\gamma_R\,\alpha_0\,\alpha_1) \\
&\qquad \to B^{\mathsf{R}}\,(\gamma_R, \alpha_R)\,(f_0\,\alpha_0)\,(f_1\,\alpha_1)
\end{aligned}$$

Morphisms: the logical relation interpretation modified

(A simpler version of the interpretation for sections)

Specification:

$$\frac{\Gamma \vdash}{\Gamma^{\mathsf{M}} : \Gamma^{\mathsf{A}} \to \Gamma^{\mathsf{A}} \to \mathsf{Type}} \qquad \frac{\Gamma \vdash A}{A^{\mathsf{M}} : \Gamma^{\mathsf{M}}\,\gamma_0\,\gamma_1 \to A^{\mathsf{A}}\,\gamma_0 \to A^{\mathsf{A}}\,\gamma_1 \to \mathsf{Type}}$$

$$\frac{\Gamma \vdash t : A}{t^{\mathsf{M}} : (\gamma_M : \Gamma^{\mathsf{M}}\,\gamma_0\,\gamma_1) \to A^{\mathsf{M}}\,\gamma_R\,(t^{\mathsf{A}}\,\gamma_0)\,(t^{\mathsf{A}}\,\gamma_1)}$$

Implementation:

$$\begin{aligned}
\cdot^{\mathsf{M}}\,\mathsf{tt}\,\mathsf{tt} &:\equiv \top \\
(\Gamma, x : A)^{\mathsf{M}}\,(\gamma_0, \alpha_0)\,(\gamma_1, \alpha_1) &:\equiv (\gamma_M : \Gamma^{\mathsf{M}}\,\gamma_0\,\gamma_1) \times A^{\mathsf{M}}\,\gamma_M\,\alpha_0\,\alpha_1 \\
x^{\mathsf{M}}\,\gamma_M &:\equiv x^{\mathrm{th}} \text{ component in } \gamma_M \\
\mathsf{U}^{\mathsf{M}}\,\gamma_M\,a_0\,a_1 &:\equiv a_0 \to a_1 \\
\underline{a}^{\mathsf{M}}\,\gamma_M\,\alpha_0\,\alpha_1 &:\equiv (a^{\mathsf{M}}\,\gamma_M\,\alpha_0 = \alpha_1) \\
((x : a) \to B)^{\mathsf{M}}\,\gamma_M\,f_0\,f_1 &:\equiv (\alpha_0 : a^{\mathsf{A}}\,\gamma_0) \\
&\quad \to B^{\mathsf{M}}\,(\gamma_M, \mathsf{refl}_{a^{\mathsf{M}}\,\gamma_M\,\alpha_0})\,(f_0\,\alpha_0)\,(f_1\,(a^{\mathsf{M}}\,\gamma_M\,\alpha_0))
\end{aligned}$$

Remarks

Summary so far

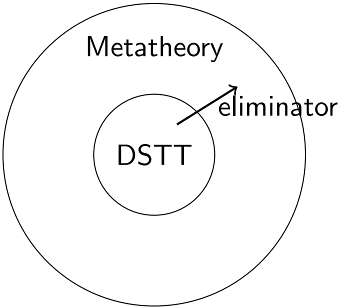
- Examples of inductive types
- What is a specification: codes, algebras, families, sections
- We specify HIITs specified by contexts in the DSTT
  - algebras are given by the standard model
  - families are given by the logical predicate interpretation
  - sections are given by a modified logical relation interpretation

Category model

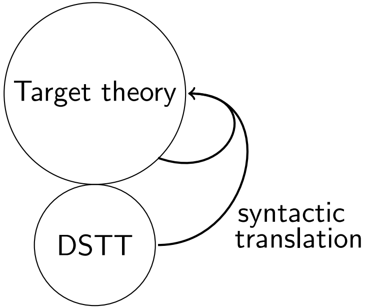
The DSTT also has a category model. For a context $\vdash \Gamma$, the intepretation gives the category of algebras.

- objects: Γ^{A}
- morphisms between γ_0 and γ_1: $\Gamma^{\mathsf{M}}\,\gamma_0\,\gamma_1$
- identity, composition, etc.

Variants of the DSTT



Coherence problem



Weak β rule

Existence (WIP)

If natural numbers are given by

$$\Gamma :\equiv (\cdot,\ \mathit{Nat} : \mathsf{U},\ \mathit{zero} : \mathit{Nat},\ \mathit{suc} : \mathit{Nat} \to \mathit{Nat}),$$

then the initial algebra (an element of $(N : \mathsf{Type}) \times N \times (N \to N)$) is given by

$$\mathsf{con}_\Gamma :\equiv \big(\{t \mid \Gamma \vdash t : \underline{\mathit{Nat}}\}, \mathit{zero}, \lambda n.\mathit{suc}\ n\big).$$

Given another algebra $\gamma : \Gamma^{\mathsf{A}}$, we get a morphism from con_Γ to γ by

$$(\lambda t.t^{\mathsf{A}}\,\gamma, \mathsf{refl}, \mathsf{refl}).$$

Similarly, given $\gamma_F : \Gamma^{\mathsf{F}}\,\mathsf{con}_\Gamma$, we get a section of m by

$$(\lambda t.t^{\mathsf{F}}\,\gamma_F, \mathsf{refl}, \mathsf{refl}).$$

Questions and further work

- Equalities are weak, how to strictify them? Do we need strict equalities?
- For HIITs, the initial algebra is not terms but terms quotiented by equality constructors. How to do this?
- Define a type theory which supports HIITs.
- Dualise the construction for coinductive types.
- Combine with cubical type theory.