A syntax for higher inductive inductive types (HIITs)

Ambrus Kaposi
Eötvös Loránd University, Budapest

j.w.w. András Kovács and Thorsten Altenkirch

Agda Implementors' Meeting Budapest,
31 January 2018

An inductive type

- Specification

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

- Eliminator:

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

- Computation rules:

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

An indexed inductive type

- Specification

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

- Eliminator:

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

- Computation rules

Mutual inductive types

- Specification

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

- Eliminator
- Computation rules

An inductive-inductive type

- Specification

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

- Eliminator

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

- Computation rules

A higher inductive type

- Specification

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

- Eliminator

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

- Computation rules

A higher inductive-inductive type (HIIT)

Combination of all of the previous ones.

Example usage: intrinsic syntax of type theory. conversion.

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
Quotient inductive-inductive types	Fredrik's talk
Higher inductive-inductive types	???

What is a specification?

A coding scheme.

Codes	$\mathsf{Code} \in \mathsf{Type}$
Constructors	$-^{\mathsf{C}} \in \mathsf{Code} \to \mathsf{Type}$
Methods	$-^{\mathsf{M}} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}} \to \mathsf{Type}$
Eliminators	$-^{\mathsf{E}} \in (\Gamma \in \mathsf{Code})(\gamma \in \Gamma^{\mathsf{C}}) \to \Gamma^{\mathsf{M}}\,\gamma \to \mathsf{Type}$
Existence	$\mathsf{con} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}}$
	$\mathsf{elim} \in (\Gamma \in \mathsf{Code})(m \in \Gamma^{\mathsf{M}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{E}}\,(\mathsf{con}\,\Gamma)\,m$

W-types

A coding scheme.

Codes	$\mathsf{Code} :\equiv (S : \mathsf{Type}) \times (S \to \mathsf{Type})$
Constructors	$-^{\mathsf{C}} \in \mathsf{Code} \to \mathsf{Type}$
Methods	$-^{\mathsf{M}} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}} \to \mathsf{Type}$
Eliminators	$-^{\mathsf{E}} \in (\Gamma \in \mathsf{Code})(\gamma \in \Gamma^{\mathsf{C}}) \to \Gamma^{\mathsf{M}}\,\gamma \to \mathsf{Type}$
Existence	$\mathsf{con} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}}$
	$\mathsf{elim} \in (\Gamma \in \mathsf{Code})(m \in \Gamma^{\mathsf{M}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{E}}\,(\mathsf{con}\,\Gamma)\,m$

W-types

A coding scheme.

Codes	$\mathsf{Code} :\equiv (S : \mathsf{Type}) \times (S \to \mathsf{Type})$
Constructors	$(S, P)^{\mathsf{C}} :\equiv (W \in \mathsf{Set}) \times \big((s \in S) \to (P\,s \to W) \to W\big)$
Methods	$-^{\mathsf{M}} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}} \to \mathsf{Type}$
Eliminators	$-^{\mathsf{E}} \in (\Gamma \in \mathsf{Code})(\gamma \in \Gamma^{\mathsf{C}}) \to \Gamma^{\mathsf{M}}\,\gamma \to \mathsf{Type}$
Existence	$\mathsf{con} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}}$
	$\mathsf{elim} \in (\Gamma \in \mathsf{Code})(m \in \Gamma^{\mathsf{M}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{E}}\,(\mathsf{con}\,\Gamma)\,m$

W-types

A coding scheme.

Codes	$\mathsf{Code} :\equiv (S : \mathsf{Type}) \times (S \to \mathsf{Type})$
Constructors	$(S, P)^{\mathsf{C}} :\equiv (W \in \mathsf{Set}) \times \big((s \in S) \to (P\,s \to W) \to W\big)$
Methods	$(S, P)^{\mathsf{M}}\,(W, sup) :\equiv (W^M \in W \to \mathsf{Set}) \times$
	$\big((s \in S)(f \in P\,s \to W) \to (\forall p. W^M\,(f\,p)) \to W^M\,(sup\,s\,f)\big)$
Eliminators	$-^{\mathsf{E}} \in (\Gamma \in \mathsf{Code})(\gamma \in \Gamma^{\mathsf{C}}) \to \Gamma^{\mathsf{M}}\,\gamma \to \mathsf{Type}$
Existence	$\mathsf{con} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}}$
	$\mathsf{elim} \in (\Gamma \in \mathsf{Code})(m \in \Gamma^{\mathsf{M}}\,(\mathsf{con}\,\Gamma)) \to \Gamma^{\mathsf{E}}\,(\mathsf{con}\,\Gamma)\,m$

W-types

A coding scheme.

Codes	$\mathsf{Code} :\equiv (S : \mathsf{Type}) \times (S \to \mathsf{Type})$
Constructors	$(S, P)^{\mathsf{C}} :\equiv (W \in \mathsf{Set}) \times \big((s \in S) \to (P\, s \to W) \to W\big)$
Methods	$(S, P)^{\mathsf{M}}\, (W, sup) :\equiv (W^M \in W \to \mathsf{Set})\times$
	$\big((s \in S)(f \in P\, s \to W) \to (\forall p. W^M\, (f\ p)) \to W^M\, (sup\, s\, f)\big)$
Eliminators	$(S, P)^{\mathsf{E}}\, (W, sup)\, (W^M, sup^M) :\equiv (W^E \in (w \in W) \to W^M\, w)$
	$\times\ \big(\forall s\, f. W^E\, (sup\, s\, f) = sup^M\, s\, f\, (\lambda p. W^E\, (f\ p))\big)$
Existence	$\mathsf{con} \in (\Gamma \in \mathsf{Code}) \to \Gamma^{\mathsf{C}}$
	$\mathsf{elim} \in (\Gamma \in \mathsf{Code})(m \in \Gamma^{\mathsf{M}}\, (\mathsf{con}\, \Gamma)) \to \Gamma^{\mathsf{E}}\, (\mathsf{con}\, \Gamma)\, m$

What should the type of codes be for HIITs?

Lots of intricate dependencies.

What should the type of codes be for HIITs?

Lots of intricate dependencies.

Tool to describe intricate dependencies: the syntax of type theory.

What should the type of codes be for HIITs?

Lots of intricate dependencies.

Tool to describe intricate dependencies: the syntax of type theory.

A code for an inductive type is a context.

What should the type of codes be for HIITs?

Lots of intricate dependencies.

Tool to describe intricate dependencies: the syntax of type theory.

A code for an inductive type is a context.

E.g. $\cdot,\ \mathit{Nat} : \text{Type},\ \mathit{zero} : \mathit{Nat},\ \mathit{suc} : \mathit{Nat} \to \mathit{Nat}$

Logical predicate translation

Specification:

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

Logical predicate translation

Specification:

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

Implementation:

$$\begin{aligned}
(\Gamma, x : A)^{\mathsf{M}} &:\equiv \Gamma^{\mathsf{M}}, x : A, x_M : A^{\mathsf{M}}\, x \\
x^{\mathsf{M}} &:\equiv x_M \\
(A \to B)^{\mathsf{M}}\, f &:\equiv (x : A)(x_M : A^{\mathsf{M}}\, x) \to B^{\mathsf{M}}\, (f\, x) \\
\mathsf{Type}^{\mathsf{M}}\, A &:\equiv A \to \mathsf{Type} \\
\ldots
\end{aligned}$$

First slide

- Specification

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

- Eliminator:

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

- Computation rules:

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

Logical relation translation

For non-dependent eliminator.

Universe à la Tarski.

For closed A, $A^{\mathsf{R}} : A \to A \to \mathsf{U}$

$\mathsf{U}^{\mathsf{R}}\, A\, B :\equiv A \to B \to \mathsf{U}$

$(\mathsf{El}\, a)^{\mathsf{R}}\, t\, u :\equiv (a^{\mathsf{R}}\, t\, u)$

Logical relation translation changed

For non-dependent eliminator.

Universe à la Tarski.

For closed A, $A^{\mathsf{R}} : A \to A \to \mathsf{U}$

$\mathsf{U}^{\mathsf{R}}\, A\, B :\equiv A \to B$

$(\mathsf{El}\, a)^{\mathsf{R}}\, t\, u :\equiv (a^{\mathsf{R}}\, t = u)$

Natural numbers

Rules:

$$
\begin{aligned}
(\Gamma, x : A)^{\mathsf{R}} \quad &:\equiv \Gamma^{\mathsf{R}}, x_0 : A[0], x_1 : A[1], x_R : A^{\mathsf{R}}\, x_0\, x_1 \\
\mathsf{U}^{\mathsf{R}}\, a\, b \quad &:\equiv a \to El\, b
\end{aligned}
$$

Natural numbers:

$$
\begin{aligned}
&\left(nat : \mathsf{U},\ zero : \mathsf{El}\, nat,\ suc : nat \to \mathsf{El}\, nat\right)^{\mathsf{R}} \\
\equiv\ & nat_0 : \mathsf{U},\ nat_1 : \mathsf{U},\ nat_R : nat_0 \to \mathsf{El}\, nat_1,
\end{aligned}
$$

Natural numbers

Rules:

$$
\begin{aligned}
(\Gamma, x : A)^{\mathsf{R}} &:\equiv \Gamma^{\mathsf{R}}, x_0 : A[0], x_1 : A[1], x_R : A^{\mathsf{R}}\, x_0\, x_1 \\
\mathsf{U}^{\mathsf{R}}\, a\, b &:\equiv a \to El\, b \\
(\mathsf{El}\, a)^{\mathsf{R}}\, t\, u &:\equiv (a^{\mathsf{R}}\, t = u)
\end{aligned}
$$

Natural numbers:

$$
\begin{aligned}
& (nat : \mathsf{U},\ zero : \mathsf{El}\, nat,\ suc : nat \to \mathsf{El}\, nat)^{\mathsf{R}} \\
\equiv\ & nat_0 : \mathsf{U},\ nat_1 : \mathsf{U},\ nat_R : nat_0 \to \mathsf{El}\, nat_1, \\
& zero_0 : \mathsf{El}\, nat_0,\ zero_1 : \mathsf{El}\, nat_1,\ zero_R : nat^R\, zero_0 = zero_1,
\end{aligned}
$$

Natural numbers

Rules:

$$
\begin{aligned}
(\Gamma, x : A)^{\mathsf{R}} &:\equiv \Gamma^{\mathsf{R}}, x_0 : A[0], x_1 : A[1], x_R : A^{\mathsf{R}}\, x_0\, x_1 \\
\mathsf{U}^{\mathsf{R}}\, a\, b &:\equiv a \to El\, b \\
(\mathsf{El}\, a)^{\mathsf{R}}\, t\, u &:\equiv (a^{\mathsf{R}}\, t = u) \\
(a \to B)^{\mathsf{R}}\, f_0\, f_1 &:\equiv (x_0 : a[0]) \to B^{\mathsf{R}}\, (f_0\, x_0)\, (f_1\, (a^{\mathsf{E}}\, x_0))
\end{aligned}
$$

Natural numbers:

$$
\begin{aligned}
& (nat : \mathsf{U},\ zero : \mathsf{El}\, nat,\ suc : nat \to \mathsf{El}\, nat)^{\mathsf{R}} \\
\equiv\ & nat_0 : \mathsf{U},\ nat_1 : \mathsf{U},\ nat_R : nat_0 \to \mathsf{El}\, nat_1, \\
& zero_0 : \mathsf{El}\, nat_0,\ zero_1 : \mathsf{El}\, nat_1,\ zero_R : nat^R\, zero_0 = zero_1, \\
& suc_0 : nat_0 \to \mathsf{El}\, nat_0,\ suc_1 : nat_1 \to \mathsf{El}\, nat_1, \\
& suc_R : (x_0 : nat_0) \to nat^R\, (suc_0\, x_0) = suc_1\, (nat^R\, x_0)
\end{aligned}
$$

A domain-specific type theory

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: $\dfrac{T \in \mathsf{Set}_0 \quad \forall(\alpha \in T).\Gamma \vdash B_\alpha}{\Gamma \vdash (\alpha \in T) \to B_\alpha}$ $\dfrac{\Gamma \vdash t : (\alpha \in T) \to B_\alpha \quad \alpha' \in T}{\Gamma \vdash t\,\alpha' : B_{\alpha'}}$

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: $\dfrac{T \in \mathsf{Set}_0 \quad \forall(\alpha \in T).\Gamma \vdash b_\alpha : \mathsf{U}}{\Gamma \vdash (\alpha \in T) \to b : \mathsf{U}}$ $\dfrac{\Gamma \vdash t : \underline{(\alpha \in T) \to b} \quad \alpha' \in T}{\Gamma \vdash t\,\alpha' : \underline{b_{\alpha'}}}$

A domain-specific type theory

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

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

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

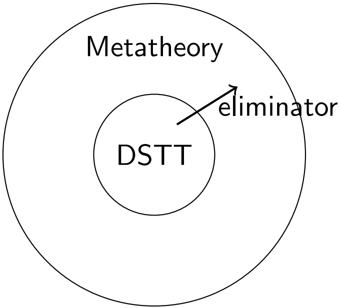
Non-inductive params:
$$\frac{T \in \mathsf{Set}_0 \quad \forall(\alpha \in T).\Gamma \vdash B_\alpha}{\Gamma \vdash (\alpha \in T) \to B_\alpha} \qquad \frac{\Gamma \vdash t : (\alpha \in T) \to B_\alpha \quad \alpha' \in T}{\Gamma \vdash t\,\alpha' : B_{\alpha'}}$$

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

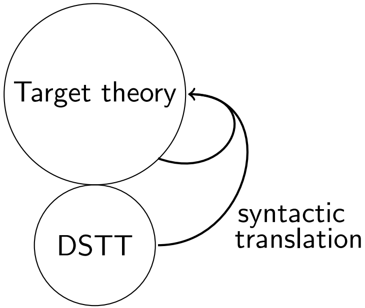
Infinitary params:
$$\frac{T \in \mathsf{Set}_0 \quad \forall(\alpha \in T).\Gamma \vdash b_\alpha : \mathsf{U}}{\Gamma \vdash (\alpha \in T) \to b : \mathsf{U}} \qquad \frac{\Gamma \vdash t : \underline{(\alpha \in T) \to b} \quad \alpha' \in T}{\Gamma \vdash t\,\alpha' : \underline{b_{\alpha'}}}$$

We can define the $-^{\mathsf{C}}$, $-^{\mathsf{M}}$, $-^{\mathsf{E}}$ operations for this theory and they give the expected elimination principles (Haskell implementation).

Variants of the DSTT



Coherence problem



Weak beta rules