A Syntax for Mutual Inductive Families

Ambrus Kaposi[1] Jakob von Raumer[2] | July 2, 2020

[1] Eötvös Loránd University, Budapest, [2] University of Nottingham

Setting

We assume that we work in a type theory supporting:

- Universe(s) $\mathcal{U}$
- Π-types (dependent functions) $(x : A) \to B(x)$ for $A : \mathcal{U}$ and $B : A \to \mathcal{U}$
- Σ-types (dependent pairs) $(x : A) \times B(x)$ for $A : \mathcal{U}$ and $B : A \to \mathcal{U}$
- Indexed W-types (generalized well-founded trees)

Inductive Families

Inductive families are a way to define data types in dependently typed languages. Prime example: The type of natural numbers $\mathbb{N} : \mathcal{U}$ with constructors $0 : \mathbb{N}$ and $S : \mathbb{N} \to \mathbb{N}$, admitting an *eliminator* of the following type:

$$\frac{P : \mathbb{N} \to \mathcal{U} \qquad p_0 : P(z) \qquad p_S : (n : \mathbb{N}) \to P(n) \to P(S(n))}{\mathrm{elim}_{\mathbb{N}}(P, p_0, p_S) : (n : \mathbb{N}) \to P(n)}$$

with *reduction rules* stating $\mathrm{elim}_{\mathbb{N}}(P, p_0, p_S)(0) = p_0$ and $\mathrm{elim}_{\mathbb{N}}(P, p_0, p_S)(S(n)) = p_S(\mathrm{elim}_{\mathbb{N}}(P, p_0, p_S)(n))$ for alll $n : \mathbb{N}$.

Inductive Families

We want to find a description of Inductive Types encompassing:

- Plain inductive types like $\mathbb{N} : \mathcal{U}$,
- inductive *families* like the vectors $\mathsf{Vec} : \mathbb{N} \to \mathcal{U}$ on a type $A : \mathcal{U}$ with constructors

$$\begin{aligned} \mathsf{nil} &: \mathsf{Vec}(0) \text{ and} \\ \mathsf{cons} &: \mathsf{Vec}(n) \to A \to \mathsf{Vec}(n+1)\text{, and} \end{aligned}$$

- *mutually defined* type families like predicates for evenness and oddness $\mathsf{isEven}, \mathsf{isOdd} : \mathbb{N} \to \mathcal{U}$ with constructors

$$\begin{aligned} \mathsf{evenZero} &: \mathsf{isEven}(0) \\ \mathsf{evenS} &: \{n : \mathbb{N}\} \to \mathsf{isOdd}(n) \to \mathsf{isEven}(S(n)) \\ \mathsf{oddS} &: \{n : \mathbb{N}\} \to \mathsf{isEven}(n) \to \mathsf{isOdd}(S(n)). \end{aligned}$$

Inductive Families

What the definition should *avoid*:

- Constructors which are not strictly positive, such as a type $A : \mathcal{U}$ with a constructor $f : (A \to \mathbf{2}) \to A$
- Having to encode/uncurry constructors like $f : A \to A \to A$ into $f : A \times A \to A$.
- Having to encode mutual types into a single indexed type.
- Resorting to a *schematic* description that can't easily be internalized.

Inductive Families

What the definition should *avoid*:

- Constructors which are not strictly positive, such as a type $A : \mathcal{U}$ with a constructor $f : (A \to \mathbf{2}) \to A$
- Having to encode/uncurry constructors like $f : A \to A \to A$ into $f : A \times A \to A$.
- Having to encode mutual types into a single indexed type.
- Resorting to a *schematic* description that can't easily be internalized.

Inductive Families

What the definition should *avoid*:

- Constructors which are not strictly positive, such as a type $A : \mathcal{U}$ with a constructor $f : (A \to \mathbf{2}) \to A$
- Having to encode/uncurry constructors like $f : A \to A \to A$ into $f : A \times A \to A$.
- Having to encode mutual types into a single indexed type.
- Resorting to a *schematic* description that can't easily be internalized.

Inductive Families

What the definition should *avoid*:

- Constructors which are not strictly positive, such as a type $A : \mathcal{U}$ with a constructor $f : (A \to \mathbf{2}) \to A$
- Having to encode/uncurry constructors like $f : A \to A \to A$ into $f : A \times A \to A$.
- Having to encode mutual types into a single indexed type.
- Resorting to a *schematic* description that can't easily be internalized.

Inductive Families

Each inductive definition consists of

- A list of sorts to be defined (sort constructors, formation rules)
- A list of (point) constructors (introduction rules)
- A (dependent) elimination rule (induction rule)
- Some computation rules (reduction rules)

Signatures for Inductive Families

- Idea: Inductive definitions look like pieces (contexts) of type theoretic syntax (Kaposi & Kovács, Higher Inductive-Inductive Types)
- Want to have a small type theory for the *description* of inductive families.
- The syntax of this small type theory is to be given inductively as well: Welltyped syntax!

Syntax of Signatures – Sorts

Have *sort types* $\mathsf{Ty_S}$ generated by

$$\frac{}{\mathcal{U} : \mathsf{Ty_S}} \qquad \frac{T : \mathcal{U} \qquad B : T \to \mathsf{Ty_S}}{\hat{\Pi}_{\mathsf{S}}(T, B) : \mathsf{Ty_S}}$$

and *sort contexts* $\mathsf{Con_S}$ as lists

$$\frac{}{\cdot : \mathsf{Con_S}} \qquad \frac{\Gamma_s : \mathsf{Con_S} \qquad B : \mathsf{Ty_S}}{(\Gamma_s \triangleright B) : \mathsf{Con_S}}$$

Syntax of Signatures – Sort Terms

We can refer to the sorts in a context using *typed de-Bruijn-variables*, and by applying type families:

$$\frac{\Gamma_s : \mathrm{Con}_S \qquad B : \mathrm{Ty}_S}{\mathrm{var}(\mathrm{vz}) : \mathrm{Tm}_S((\Gamma_s, B), B)} \qquad \frac{\mathrm{var}(v) : \mathrm{Tm}_S(\Gamma_s, B)}{\mathrm{var}(\mathrm{vs}(v)) : \mathrm{Tm}_S((\Gamma_s, B'), B)}$$

$$\frac{t : \mathrm{Tm}_S(\Gamma_s, \hat{\Pi}_S(T, B)) \qquad \tau : T}{t(\tau) : \mathrm{Tm}_S(\Gamma_s, B(\tau))}$$

Examples for Sort Contexts

- For natural numbers: $\cdot \rhd \mathcal{U}$
- For vectors: $\cdot \rhd \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \lambda n.\mathcal{U})$
- For parity: $\cdot \rhd \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \lambda n.\mathcal{U}) \rhd \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \lambda n.\mathcal{U})$

Syntax of Signatures – Point Constructors

Point constructors ("point types") Ty_P consist of

- Elements of the sorts in consideration:

$$\frac{a : \mathsf{Tm}_\mathsf{S}(\Gamma_s, \mathcal{U})}{\mathsf{El}(a) : \mathsf{Ty}_\mathsf{P}(\Gamma_s)}$$

- Functions with an "external" domain:

$$\frac{T : \mathcal{U} \qquad B : T \to \mathsf{Ty}_\mathsf{P}(\Gamma_s)}{\hat{\Pi}_\mathsf{P}(T, B) : \mathsf{Ty}_\mathsf{P}(\Gamma_s)}$$

- Recursive functions with a "small" domain:

$$\frac{a : \mathsf{Tm}_\mathsf{S}(\Gamma_s, \mathcal{U}) \qquad A : \mathsf{Ty}_\mathsf{P}(\Gamma_s)}{(a \Rightarrow_\mathsf{P} A) : \mathsf{Ty}_\mathsf{P}(\Gamma_s)}$$

Syntax of Signatures – Point Constructors

We again collect a series of point constructors in a list $\mathrm{Con}_{\mathrm{P}}$:

$$\frac{\Gamma_s : \mathrm{Con}_{\mathrm{S}}}{\cdot : \mathrm{Con}_{\mathrm{P}}(\Gamma_s)} \qquad \frac{\Gamma : \mathrm{Con}_{\mathrm{P}}(\Gamma_s) \qquad A : \mathrm{Ty}_{\mathrm{P}}(\Gamma_s)}{(\Gamma \rhd A) : \mathrm{Con}_{\mathrm{P}}(\Gamma_s)}$$

Examples for Point Contexts

- For natural numbers with a sort context $(\cdot \rhd \mathcal{U})$ we have

$$\mathsf{El}(\mathsf{var}(\mathsf{vz})) \rhd \mathsf{var}(\mathsf{vz}) \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathsf{var}(\mathsf{vz}))$$

- For parity, with a sort context $(\cdot \rhd \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \lambda n.\mathcal{U}) \rhd \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \lambda n.\mathcal{U}))$ we want

$$\begin{gathered}\cdot \rhd \mathsf{El}(\mathsf{var}(\mathsf{vs}(\mathsf{vz}))) \\ \rhd \hat{\Pi}_{\mathsf{P}}(\mathbb{N}, \lambda n.\, \mathsf{var}(\mathsf{vz})(n) \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathsf{var}(\mathsf{vs}(\mathsf{vz}))(n+1))) \\ \rhd \hat{\Pi}_{\mathsf{P}}(\mathbb{N}, \lambda n.\, \mathsf{var}(\mathsf{vs}(\mathsf{vz}))(n) \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathsf{var}(\mathsf{vz})(n+1))).\end{gathered}$$

Examples for Point Contexts

- For natural numbers with a sort context $(\cdot \triangleright \mathbb{N} : \mathcal{U})$ we have

$$\mathsf{El}(\mathbb{N}) \triangleright \mathbb{N} \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathbb{N})$$

- For parity, with a sort context $(\cdot \triangleright \mathsf{isEven} : \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \mathcal{U}) \triangleright \mathsf{isOdd} : \hat{\Pi}_{\mathsf{S}}(\mathbb{N}, \mathcal{U}))$ we want

$$\begin{aligned} &\cdot \triangleright \mathsf{El}(\mathsf{isEven}(0)) \\ &\triangleright \hat{\Pi}_{\mathsf{P}}(\mathbb{N}, \lambda n.\, \mathsf{isOdd}(n) \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathsf{isEven}(n+1))) \\ &\triangleright \hat{\Pi}_{\mathsf{P}}(\mathbb{N}, \lambda n.\, \mathsf{isEven}(n) \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathsf{isOdd}(n+1))). \end{aligned}$$

Example: Algebras of the Natural Number Signature

We can now form the standard model of the signatures in the "outer" type theory:

$$(\cdot \rhd \mathbb{N} : \mathcal{U})^{\mathsf{A}} \equiv \mathbf{1} \times \mathcal{U}$$

and for $(\star, N) : \mathbf{1} \times \mathcal{U}$ have

$$(\cdot \rhd \mathsf{El}(\mathbb{N}) \rhd \mathbb{N} \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathbb{N}))^{\mathsf{A}}(\star, N) \equiv \mathbf{1} \times N \times (N \to N)$$

$\leadsto$ Each algebra admits the sort and point constructor. But not necessarily an eliminator!

Example: Algebras of the Natural Number Signature

We can now form the standard model of the signatures in the "outer" type theory:

$$(\cdot \rhd \mathbb{N} : \mathcal{U})^{\mathsf{A}} \equiv \mathbf{1} \times \mathcal{U}$$

and for $(\star, N) : \mathbf{1} \times \mathcal{U}$ have

$$(\cdot \rhd \mathsf{El}(\mathbb{N}) \rhd \mathbb{N} \Rightarrow_{\mathsf{P}} \mathsf{El}(\mathbb{N}))^{\mathsf{A}}(\star, N) \equiv \mathbf{1} \times N \times (N \to N)$$

$\rightsquigarrow$ Each algebra admits the sort and point constructor. But not necessarily an eliminator!

Displayed Algebras

The input data for the eliminator is held in what we call a *displayed algebra* over a given algebra:

Take a sort algebra $(\star, N)$ and a point algebra $(\star, z, s)$ over the natural numbers signature. Then,

$$\Gamma_s{}^{\mathsf{D}}(\star, N) \equiv (P : N \to \mathcal{U}) \text{ and}$$

$$\Gamma^{\mathsf{D}}(\star, z, s)(P) \equiv P(z) \times ((n : N) \to P(n) \to P(s(n)))$$

$\rightsquigarrow$ Displayed algebras specify the *input* for an induction. But what is the type of the corresponding output?

Displayed Algebras

The input data for the eliminator is held in what we call a *displayed algebra* over a given algebra:
Take a sort algebra $(\star, N)$ and a point algebra $(\star, z, s)$ over the natural numbers signature. Then,

$$\Gamma_s{}^{\mathsf{D}}(\star, N) \equiv (P : N \to \mathcal{U}) \text{ and}$$

$$\Gamma^{\mathsf{D}}(\star, z, s)(P) \equiv P(z) \times ((n : N) \to P(n) \to P(s(n)))$$

$\rightsquigarrow$ Displayed algebras specify the *input* for an induction. But what is the type of the corresponding output?

Displayed Algebras

The input data for the eliminator is held in what we call a *displayed algebra* over a given algebra:
Take a sort algebra $(\star, N)$ and a point algebra $(\star, z, s)$ over the natural numbers signature. Then,

$$\Gamma_s{}^{\mathsf{D}}(\star, N) \equiv (P : N \to \mathcal{U}) \text{ and}$$
$$\Gamma^{\mathsf{D}}(\star, z, s)(P) \equiv P(z) \times ((n : N) \to P(n) \to P(s(n)))$$

$\rightsquigarrow$ Displayed algebras specify the *input* for an induction. But what is the type of the corresponding output?

Sections of Displayed Algebras

Sections of displayed algebras represent the output of eliminators:

Assume

- sort and point algebras $(\star, N)$ and $(\star, z, s)$ for the natural numbers,
- and furthermore displayed algebras $(\star, P)$ and $(\star, p_z, p_s)$ with $P : N \to \mathcal{U}$, $p_z : P(z)$ and $p_s : (n : N) \to P(n) \to P(s(n))$.

Then, the sections are

$$\Gamma_s{}^S(\star, P) \equiv ((n : N) \to P(n)) \text{ and}$$

$$\Gamma^S(\star, p_z, p_s)(f) \equiv (f(z) = p_z) \times ((n : N) \to f(s(n)) = p_s(n)).$$

Sections of Displayed Algebras

Sections of displayed algebras represent the output of eliminators:
Assume

- sort and point algebras $(\star, N)$ and $(\star, z, s)$ for the natural numbers,
- and furthermore displayed algebras $(\star, P)$ and $(\star, p_z, p_s)$ with $P : N \to \mathcal{U}$, $p_z : P(z)$ and $p_s : (n : N) \to P(n) \to P(s(n))$.

Then, the sections are

$$\Gamma_s{}^S(\star, P) \equiv ((n : N) \to P(n)) \text{ and}$$

$$\Gamma^S(\star, p_z, p_s)(f) \equiv (f(z) = p_z) \times ((n : N) \to f(s(n)) = p_s(n)).$$

Sections of Displayed Algebras

Sections of displayed algebras represent the output of eliminators:
Assume

- sort and point algebras $(\star, N)$ and $(\star, z, s)$ for the natural numbers,
- and furthermore displayed algebras $(\star, P)$ and $(\star, p_z, p_s)$ with $P : N \to \mathcal{U}$, $p_z : P(z)$ and $p_s : (n : N) \to P(n) \to P(s(n))$.

Then, the sections are

$$\Gamma_s{}^{\mathsf{S}}(\star, P) \equiv ((n : N) \to P(n)) \text{ and}$$
$$\Gamma^{\mathsf{S}}(\star, p_z, p_s)(f) \equiv (f(z) = p_z) \times ((n : N) \to f(s(n)) = p_s(n)).$$

Existence of Inductive Families

Theorem (Existence of Inductive Families)

For every signature of inductive families given by a sort context $\Omega_s : \mathsf{Con}_\mathsf{S}$ *and point context* $\Omega : \mathsf{Con}_\mathsf{P}(\Omega_s)$*, there are are sort and point constructors in the form of*

$$\begin{aligned} \mathsf{con}_\mathsf{S}(\Omega) &: {\Omega_s}^\mathsf{A} \text{ and} \\ \mathsf{con}(\Omega) &: \Omega^\mathsf{A}\,(\mathsf{con}_\mathsf{S}(\Omega)) \end{aligned}$$

such that for each displayed algebra given by motives $\omega_s^d : {\Omega_s}^\mathsf{D}(\mathsf{con}_\mathsf{S}(\Omega))$ *and methods* $\omega^d : \Omega^\mathsf{D}(\omega_s^d, \mathsf{con}(\Omega))$ *we have an eliminator given by sections*

$$\begin{aligned} \mathsf{elim}_\mathsf{S}(\Omega, \omega_s^d) &: {\Omega_s}^\mathsf{S}(\mathsf{con}_\mathsf{S}(\Omega), \omega_s^d) \text{ with} \\ \mathsf{elim}(\Omega, \omega^d) &: \Omega^\mathsf{S}(\mathsf{elim}_\mathsf{S}(\Omega, \omega_s^d), \mathsf{con}(\Omega), \omega^d). \end{aligned}$$

Proof of Existence

We derive the existence from the presence of *indexed W-types*

1. Extend syntax with a *substitution calculus* and *terms for point constructors* Tm_P.
2. Represent extended syntax using indexed W-types.
3. Construct $\mathsf{con}(\Omega)$ *term model* of the syntax: For a term $a : \mathsf{Tm}_\mathsf{S}(\Omega_s, \mathcal{U})$ set

$$\mathsf{con}(a) :\equiv W_{\mathsf{Tm}_\mathsf{P}}(\Omega_s, \mathsf{El}(a)).$$

Further Results

What did we do with this notion of Inductive Families?

- Syntax and existence proof have been *formalized* in Agda.
- We used our definition as a target for a formal treatment of *type erasure* on inductive-inductive types.

Further ideas:

- Prove various results about inductive types (Lambek's lemma, ...)
- Allow *infinitary* constructors like $f : (\mathbb{N} \to A) \to A$

Further Results

What did we do with this notion of Inductive Families?

- Syntax and existence proof have been *formalized* in Agda.
- We used our definition as a target for a formal treatment of *type erasure* on inductive-inductive types.

Further ideas:

- Prove various results about inductive types (Lambek's lemma, ...)
- Allow *infinitary* constructors like $f : (\mathbb{N} \to A) \to A$

Thank you for your attention!