Reducing Inductive-Inductive Types via Type Erasure

Thorsten Altenkirch[1] Ambrus Kaposi[2] András Kovács[2]
Jakob von Raumer[1]

[1]University of Nottingham, United Kingdom

[2]Eötvös Loránd University, Budapest, Hungary

June 14, 2019

Inductive Families

- Provers like Lean & Coq are built on *(indexed) inductive Families* (e. g. Vec)
- These allow for mutual inductive definition: Constructors for one type may refer to other types in a non-linear way.
- A type can *not* be indexed over another type being defined (let's call rather call them *sorts*).

Running Example: Type Theory Syntax

Consider this following simplified definition of the contexts and types of a type theoretic syntax:

inductive Con : **Type**
| nil : Con
| ext : Π (Γ : Con), Ty $\Gamma \to$ Con
with Ty : Con $\to$ **Type**
| unit : Π (Γ : Con), Ty Γ
| pi : Π (Γ : Con) (A : Ty Γ), Ty (ext Γ A) $\to$ Ty Γ

Strategy for the Reduction

We will construct this algebra and prove its initiality in seven steps:

1. Erase the typing relation
2. Construct a wellformedness predicate
3. Define the inital algebra
4. Relate objects of the pre-algebra to arbitrary algebras
5. Show right-uniqueness of the relation
6. Show left-totality of the relation
7. Extract the eliminators from the relation

Strategy for the Reduction

We will construct this algebra and prove its initiality in seven steps:

1. **Erase the typing relation**
2. **Construct a wellformedness predicate**
3. **Define the inital algebra**
4. **Relate objects of the pre-algebra to arbitrary algebras**
5. Show right-uniqueness of the relation
6. Show left-totality of the relation
7. Extract the eliminators from the relation

Step 1: Type Erasure

Remove the indexing from the second sort:

inductive Con' : **Type**
| nil' : Con'
| ext' : $\mathsf{Con}' \to \mathsf{Ty}' \to \mathsf{Con}'$

with Ty' : **Type**
| unit' : $\mathsf{Con}' \to \mathsf{Ty}'$
| pi' : $\mathsf{Con}' \to \mathsf{Ty}' \to \mathsf{Ty}' \to \mathsf{Ty}'$

Step 2: Defining the Wellformedness Predicate

Inductively, define a predicate on the erased types to reinstantiate the erased dependencies:

inductive Conw : Con′ $\to$ **Type**
| nilw : Conw nil′
| extw : Π Γ A, Conw Γ $\to$ Tyw Γ A $\to$ Conw (ext′ Γ A)

with Tyw : Con′ $\to$ Ty′ $\to$ **Type**
| unitw : Π Γ, Conw Γ $\to$ Tyw Γ (unit′ Γ)
| piw : Π Γ A B, Conw Γ $\to$ Tyw Γ A $\to$ Tyw (ext Γ A) B
$\to$ Tyw Γ (pi′ Γ A B)

Step 3: Defining the Initial Algebra

Use the wellformedness predicate to select the elements of the types which we want:

$\text{Con} = \Sigma\, \Gamma : \text{Con}', \text{Conw}\ \Gamma$
$\text{nil} = \langle \text{nil}', \text{nilw} \rangle$
$\text{ext}\ \Gamma\ \text{A} = \langle \text{ext}'\ \Gamma.1\ \text{A}.1, \text{extw}\ \Gamma.1\ \text{A}.1\ \Gamma.2\ \text{A}.2 \rangle$

$\text{Ty}\ \Gamma = \Sigma\, \text{A} : \text{Ty}', \text{Tyw}\ \Gamma\ \text{A}$
$\text{unit}\ \Gamma = \langle \text{unit}'\ \Gamma.1, \text{unitw}\ \Gamma.1\ \Gamma.2 \rangle$
$\text{pi}\ \Gamma\ \text{A}\ \text{B} = \langle \text{pi}'\ \Gamma.1\ \text{A}.1\ \text{B}.1, \text{piw}\ \Gamma.1\ \text{A}.1\ \text{B}.1\ \Gamma.2\ \text{A}.2\ \text{B}.2 \rangle$

Step 4: Defining the Eliminator Relation

Given an algebra M for the inductive-inductive type, we want to define a relation r which will help us to prove initiality of the construction:

inductive Conr : Con′ $\rightarrow$ M.Con $\rightarrow$ **Type**
| nilr : Conr nil′ M.nil
| extr : Π Γ A γ α, Conr Γ γ $\rightarrow$ Tyr A α
$\rightarrow$ Conr (ext′ Γ A) (M.ext γ α)

with Tyr : Ty′ $\rightarrow$ Π $\{\gamma$: M.Con$\}$, M.Ty γ
| unitr : Π Γ γ, Conr Γ γ $\rightarrow$ Tyr (unit′ Γ) (M.unit γ)
| pir : Π Γ A B Γ α β , Conr Γ γ $\rightarrow$ Tyr A α $\rightarrow$ Tyr B β
$\rightarrow$ Tyr (pi′ Γ A B) (M.pi γ α β)

Steps 5 – 7: Proving initiality

Use induction to prove

- That the eliminator relation is left-total
- That the eliminator relation is right-unique
- The function which we gain from the previous points is unique

How to Generalize the Construction

1. Make it a well-posed problem!
   - Use a type theoretic syntax to encode IITs
   - Define the semantics (algebras) of these codes
   - Use a type theoretic syntax to encode Inductive Families
   - Postulate or prove their existence
2. Generalize the constructions as syntactic translations between the type theories
3. Prove initiality as a property of these translations (still unfinished)

Codes for Inductive-Inductive Types

- IITs are represented by contexts consisting of sort types $B :: \mathsf{S}$ and point types $A :: \mathsf{P}$
- Excerpts from the syntax: Universe and strictly positive Π-type.

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathcal{U} :: \mathsf{S}} \qquad \frac{\Gamma \vdash a : \mathcal{U}}{\Gamma \vdash \mathrm{El}(a) :: \mathsf{P}}$$

$$\frac{\Gamma \vdash a : \mathcal{U} \qquad \Gamma, \mathrm{El}(a) \vdash B :: k}{\Gamma \vdash \Pi(a, B) :: k}$$

$$\frac{T : \mathcal{U} \qquad (\tau : T) \to \Gamma \vdash B(\tau) :: k}{\Gamma \vdash \hat{\Pi}(T, B) :: k}$$

- Example: $(\mathbb{N} : \mathcal{U} :: \mathsf{S},\ \mathit{zero} : \mathrm{El}(\mathbb{N}) :: \mathsf{P},\ \mathit{suc} : \Pi(\mathbb{N}, \mathrm{El}(\mathbb{N})) :: \mathsf{P})$

Semantics of the Codes

- Each code Γ gets assigned a type of possible interpretations $\Gamma^{\mathsf{A}} : \mathcal{U}$ in the metatheory – its type of *algebras*
- Each element of the IIT sytax gets translated to its metatheoretic counterpart
- Algebras form a category
- We want to construct the initial such algebra

Codes for Inductive Families

- Previous specifications based on indexed W-types
- Instead tweak the approach for IITs
- Have *sort* and *point contexts*

Inductive Families – Sort Contexts

- Consist of *sort types* which are either the universe $\mathcal{U}$ or external functions $\hat{\Pi}_{\mathsf{S}}(T, B)$ where B is another sort type over $T : \mathcal{U}$.
- Have *sort terms* via typed de-Bruijn indices and

$$\frac{\Gamma_{\mathsf{S}} \vdash_{\mathsf{S}} t : \hat{\Pi}_{\mathsf{S}}(T, B) \qquad \tau : T}{\Gamma_{\mathsf{S}} \vdash_{\mathsf{S}} t(\tau) : B(\tau)}$$

Inductive Families – Point Contexts

- Consist of *point types* being either elements of a sort, an external function type, or an internal non-dependent function type:

$$\frac{\Gamma_S \vdash_S a : \mathcal{U}}{\Gamma_S \vdash_S \mathsf{El}(a)} \qquad \frac{T : \mathcal{U} \qquad (\tau : T) \to \Gamma_S \vdash_S B(\tau)}{\Gamma_S \vdash_S \hat{\Pi}_P(T, B)}$$

$$\frac{\Gamma_S \vdash_S a : \mathcal{U} \qquad \Gamma_S \vdash_S A}{\Gamma_S \vdash_S a \Rightarrow_P A}$$

- No need for terms or substitutions
- Add semantification ${\Gamma_S}^A, \Gamma^A$ for point ans sort contexts
- Assume initial algebras $\mathsf{con}_S(\Gamma_S) : {\Gamma_S}^A$ and $\mathsf{con}(\Gamma) : \Gamma^A$

Type Erasure

- Map IIT contexts to IF sort and point contexts:

$$\frac{\vdash \Gamma}{\vdash_{S} \Gamma^{E}_{S}} \qquad \frac{\vdash \Gamma}{\vdash_{\Gamma^{E}_{S}} \Gamma^{E}}$$

- Replace each sort type of a context by a plain universe token

The Wellformedness Predicate

- Another map into IF codes, this time depending on an algebra of the type erasure

$$\frac{\vdash \Gamma \qquad \gamma_{\mathsf{S}} : \Gamma_{\mathsf{S}}^{\mathsf{E}\,\mathsf{A}} \qquad \gamma : \Gamma^{\mathsf{E}\,\mathsf{A}}(\gamma_{\mathsf{S}})}{\vdash_{\mathsf{S}} \Gamma_{\mathsf{S}}^{\mathsf{W}}(\gamma)}$$

$$\frac{\vdash \Gamma \qquad \gamma_{\mathsf{S}} : \Gamma_{\mathsf{S}}^{\mathsf{E}\,\mathsf{A}} \qquad \gamma : \Gamma^{\mathsf{E}\,\mathsf{A}}(\gamma_{\mathsf{S}})}{\vdash_{\Gamma_{\mathsf{S}}^{\mathsf{W}}(\gamma)} \Gamma^{\mathsf{W}}(\gamma)}$$

- In the end, we will set γ_{S} and γ to be initial

Remaining Steps

Done (and formalized in Agda):

- Define the initial algebra as "Σ-type"
- Define the eliminator relation similar to the wellformedness

Future work:

- Prove initiality for the general case
- Use as a basis for provers without IIT support (Lean (4))