Dependent Inductive-Coinductive Types

Category Theoretical and Syntactic Perspectives

Henning Basold
Radboud University, Nijmegen and CWI, Amsterdam

Joint work with Herman Geuvers

Chocola Seminar, Lyon
22 September 2016

Outline

Upcoming

Introduction

What bothered me for a while

- No type theory with arbitrary inductive-coinductive, dependent types.
- People keep hold on coinductive types à la Coq.

Introduction

What bothered me for a while

- No type theory with arbitrary inductive-coinductive, dependent types.
- People keep hold on coinductive types à la Coq.

That led me to

- Extract the categorical principles behind such types.
- Build a type theory resembling the categorical principles.

Plan for this Talk

- Present dependent inductive-coinductive types as dialgebras
- Construct models for these types through polynomial functors
- Give a syntactic calculus that implements this idea
- Prove strong normalisation for that calculus
- Consider an extension with extensional identity types

Upcoming

Example

data Vec (A : **Set**) : $\mathbb{N} \to$ **Set where**
  nil : $\top \to$ Vec 0
  cons : (n : $\mathbb{N}$) $\to$ A $\times$ Vec $n \to$ Vec $(n+1)$

Interpret Vec as set family

- Lists of length n over A: $A^n = \underbrace{A \times \cdots \times A}_{n \text{ times}}$
- Vector data type: $\mathrm{Vec}\, A = \{A^n\}_{n \in \mathbb{N}}$
- $\mathrm{nil} = \{\mathrm{nil}_*\} : \{\mathbf{1}\}_{* \in \mathbf{1}} \to \{A^0\}_{* \in \mathbf{1}}$
- $\mathrm{cons} = \{\mathrm{cons}_n\}_{n \in \mathbb{N}} : \{A \times A^n\}_{n \in \mathbb{N}} \to \{A^{n+1}\}_{n \in \mathbb{N}}$

Dependent Data Types are Initial/Final Dialgebras

Interpret Vec as set family in $\mathbf{Set}^{\mathbb{N}}$:

- Lists of length n over A: $A^n = \underbrace{A \times \cdots \times A}_{n \text{ times}}$
- $\mathsf{nil} = \{\mathsf{nil}_*\} \colon \{\mathbf{1}\}_{*\in\mathbf{1}} \to \{A^0\}_{*\in\mathbf{1}}$
- $\mathsf{cons} = \{\mathsf{cons}_n\}_{n\in\mathbb{N}} : \{A \times A^n\}_{n\in\mathbb{N}} \to \{A^{n+1}\}_{n\in\mathbb{N}}$

Dependent Data Types are Initial/Final Dialgebras

Interpret Vec as set family in $\mathbf{Set}^{\mathbb{N}}$:

- Lists of length n over A: $A^n = \underbrace{A \times \cdots \times A}_{n \text{ times}}$
- $\text{nil} = \{\text{nil}_*\}$: $\{\mathbf{1}\}_{*\in\mathbf{1}} \to \{A^0\}_{*\in\mathbf{1}}$
- $\text{cons} = \{\text{cons}_n\}_{n\in\mathbb{N}}$: $\{A \times A^n\}_{n\in\mathbb{N}} \to \{A^{n+1}\}_{n\in\mathbb{N}}$

Vectors as dialgebra

Let $F, G : \mathbf{Set}^{\mathbb{N}} \to \mathbf{Set}^{\mathbf{1}} \times \mathbf{Set}^{\mathbb{N}}$ be functors to the product category

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n\in\mathbb{N}}) \text{ and}$$
$$G(X) = (\{X_0\}, \{X_{n+1}\}_{n\in\mathbb{N}})$$

Dependent Data Types are Initial/Final Dialgebras

Interpret Vec as set family in $\mathbf{Set}^{\mathbb{N}}$:

- Lists of length n over A: $A^n = \underbrace{A \times \cdots \times A}_{n \text{ times}}$
- $\text{nil} = \{\text{nil}_*\} \colon \{\mathbf{1}\}_{*\in\mathbf{1}} \to \{A^0\}_{*\in\mathbf{1}}$
- $\text{cons} = \{\text{cons}_n\}_{n\in\mathbb{N}} : \{A \times A^n\}_{n\in\mathbb{N}} \to \{A^{n+1}\}_{n\in\mathbb{N}}$

Vectors as dialgebra

Let $F, G : \mathbf{Set}^{\mathbb{N}} \to \mathbf{Set}^{\mathbf{1}} \times \mathbf{Set}^{\mathbb{N}}$ be functors to the product category

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n\in\mathbb{N}}) \text{ and}$$
$$G(X) = (\{X_0\}, \{X_{n+1}\}_{n\in\mathbb{N}})$$

Dependent Data Types are Initial/Final Dialgebras

Interpret Vec as set family in $\mathbf{Set}^{\mathbb{N}}$:

- Lists of length n over A: $A^n = \underbrace{A \times \cdots \times A}_{n \text{ times}}$
- $\mathsf{nil} = \{\mathsf{nil}_*\} \colon \{\mathbf{1}\}_{*\in\mathbf{1}} \to \{A^0\}_{*\in\mathbf{1}}$
- $\mathsf{cons} = \{\mathsf{cons}_n\}_{n\in\mathbb{N}} : \{A \times A^n\}_{n\in\mathbb{N}} \to \{A^{n+1}\}_{n\in\mathbb{N}}$

Tatsuyo Hagino

Vectors as dialgebra

Let $F, G : \mathbf{Set}^{\mathbb{N}} \to \mathbf{Set}^{\mathbf{1}} \times \mathbf{Set}^{\mathbb{N}}$ be functors to the product category

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n\in\mathbb{N}}) \text{ and}$$
$$G(X) = (\{X_0\}, \{X_{n+1}\}_{n\in\mathbb{N}})$$

Then $(\mathsf{nil}, \mathsf{cons}) : F(\mathrm{Vec}\, A) \to G(\mathrm{Vec}\, A)$ is the initial (F, G)-dialgebra.

General Setup (Generalising Hagino, 1986)

Dependent, inductive data type is an initial dialgebra in a (cloven) fibration $P : \mathbf{E} \to \mathbf{B}$ with

General Setup (Generalising Hagino, 1986)

Dependent, inductive data type is an initial dialgebra in a (cloven) fibration $P : \mathbf{E} \to \mathbf{B}$ with

- Dependencies given by an index $I \in \mathbf{B}$
- Local dependencies of constructors $J_1, \ldots, J_n \in \mathbf{B}$

General Setup (Generalising Hagino, 1986)

Dependent, inductive data type is an initial dialgebra in a (cloven) fibration $P : \mathbf{E} \to \mathbf{B}$ with

- Dependencies given by an index $I \in \mathbf{B}$
- Local dependencies of constructors $J_1, \ldots, J_n \in \mathbf{B}$
- Type given as object $A \in \mathbf{E}_I$, i.e., $A \in \mathbf{E}$ with $P(A) = I$

General Setup (Generalising Hagino, 1986)

Dependent, inductive data type is an initial dialgebra in a (cloven) fibration $P : \mathbf{E} \to \mathbf{B}$ with

- Dependencies given by an index $I \in \mathbf{B}$
- Local dependencies of constructors $J_1, \ldots, J_n \in \mathbf{B}$
- Type given as object $A \in \mathbf{E}_I$, i.e., $A \in \mathbf{E}$ with $P(A) = I$
- A functor

$$F : \mathbf{E}_I \to \mathbf{E}_{J_1} \times \cdots \times \mathbf{E}_{J_n}$$

for the constructor arguments

General Setup (Generalising Hagino, 1986)

Dependent, inductive data type is an initial dialgebra in a (cloven) fibration $P : \mathbf{E} \to \mathbf{B}$ with

- Dependencies given by an index $I \in \mathbf{B}$
- Local dependencies of constructors $J_1, \ldots, J_n \in \mathbf{B}$
- Type given as object $A \in \mathbf{E}_I$, i.e., $A \in \mathbf{E}$ with $P(A) = I$
- A functor
$$F : \mathbf{E}_I \to \mathbf{E}_{J_1} \times \cdots \times \mathbf{E}_{J_n}$$
for the constructor arguments
- Morphisms $u_1, \ldots, u_n$ with $u_k : J_n \to I$ giving the substitutions in the result type of the constructors

General Setup (Generalising Hagino, 1986)

Dependent, inductive data type is an initial dialgebra in a (cloven) fibration $P : \mathbf{E} \to \mathbf{B}$ with

- Dependencies given by an index $I \in \mathbf{B}$
- Local dependencies of constructors $J_1, \ldots, J_n \in \mathbf{B}$
- Type given as object $A \in \mathbf{E}_I$, i.e., $A \in \mathbf{E}$ with $P(A) = I$
- A functor

$$F : \mathbf{E}_I \to \mathbf{E}_{J_1} \times \cdots \times \mathbf{E}_{J_n}$$

  for the constructor arguments
- Morphisms $u_1, \ldots, u_n$ with $u_k : J_n \to I$ giving the substitutions in the result type of the constructors
- An initial dialgebra $(c_1, \ldots, c_n) : F(A) \to G_u(A)$, where

$$G_u = \langle u_1^*, \cdots, u_n^* \rangle$$

  substitutes u_k in a type.

Phew, an example after this long list

Example (Vectors)

Recall that $(\mathsf{nil}, \mathsf{cons}) : F(\mathrm{Vec}\, A) \to G(\mathrm{Vec}\, A)$ is the initial initial (F, G)-dialgebra for

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n \in \mathbb{N}}) \text{ and}$$
$$G(X) = (\{X_0\}, \{X_{n+1}\}_{n \in \mathbb{N}})$$

We have $G = G_u$ with

$$u_1 = z : \mathbf{1} \to \mathbb{N} \qquad z(*) = 0$$
$$u_2 = s : \mathbb{N} \to \mathbb{N} \qquad s(n) = n + 1$$

Giving us

$$u_1^*(X) = \{X_{z(i)}\}_{i \in \mathbf{1}} = \{X_0\}_{i \in \mathbf{1}}$$
$$u_2^*(X) = \{X_{s(n)}\}_{n \in \mathbb{N}} = \{X_{n+1}\}_{n \in \mathbb{N}}$$

Phew, an example after this long list

Example (Vectors)

Recall that $(\mathsf{nil}, \mathsf{cons}) : F(\mathrm{Vec}\, A) \to G(\mathrm{Vec}\, A)$ is the initial initial (F, G)-dialgebra for

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n \in \mathbb{N}}) \text{ and}$$
$$G(X) = (\{X_0\}, \{X_{n+1}\}_{n \in \mathbb{N}})$$

We have $G = G_u$ with

$$u_1 = z : \mathbf{1} \to \mathbb{N} \qquad z(*) = 0$$
$$u_2 = s : \mathbb{N} \to \mathbb{N} \qquad s(n) = n + 1$$

Giving us

$$u_1^*(X) = \{X_{z(i)}\}_{i \in \mathbf{1}} = \{X_0\}_{i \in \mathbf{1}}$$
$$u_2^*(X) = \{X_{s(n)}\}_{n \in \mathbb{N}} = \{X_{n+1}\}_{n \in \mathbb{N}}$$

Phew, an example after this long list

Example (Vectors)

Recall that $(\mathsf{nil}, \mathsf{cons}) : F(\mathrm{Vec}\, A) \to G(\mathrm{Vec}\, A)$ is the initial initial (F, G)-dialgebra for

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n \in \mathbb{N}}) \text{ and}$$
$$G(X) = (\{X_0\}, \{X_{n+1}\}_{n \in \mathbb{N}})$$

We have $G = G_u$ with

$$u_1 = z : \mathbf{1} \to \mathbb{N} \qquad z(*) = 0$$
$$u_2 = s : \mathbb{N} \to \mathbb{N} \qquad s(n) = n + 1$$

Giving us

$$u_1^*(X) = \{X_{z(i)}\}_{i \in \mathbf{1}} = \{X_0\}_{i \in \mathbf{1}}$$
$$u_2^*(X) = \{X_{s(n)}\}_{n \in \mathbb{N}} = \{X_{n+1}\}_{n \in \mathbb{N}}$$

Dualise for coinductive data types

- The functors F and G_u swap roles
- Substitutions in the domain of the destructors
- Can block application

Dualise for coinductive data types

- The functors F and G_u swap roles
- Substitutions in the domain of the destructors
- Can block application

Example (Partial Streams)

codata PStr (A : **Set**) : $\mathbb{N}^\infty \to$ **Set where**
  hd : (n : $\mathbb{N}^\infty$) $\to$ PStr (s_∞ n) $\to$ A
  tl : (n : $\mathbb{N}^\infty$) $\to$ PStr (s_∞ n) $\to$ PStr n

where $\mathbb{N}^\infty$ are natural numbers with infinity.

$$G_u, F : \mathbf{P}_{\mathbb{N}^\infty} \to \mathbf{P}_{\mathbb{N}^\infty} \times \mathbf{P}_{\mathbb{N}^\infty}$$
$$G_u = \langle s^*_\infty, s^*_\infty \rangle \qquad F = \langle K_A^{\mathbb{N}^\infty}, \mathrm{Id} \rangle$$

How to obtain models?

Theorem

Dependent, strictly positive data types can be interpreted in a locally Cartesian closed category with dependent products and disjoint coproducts if it has initial algebras for polynomial functors.

How to obtain models?

Theorem

Dependent, strictly positive data types can be interpreted in a locally Cartesian closed category with dependent products and disjoint coproducts if it has initial algebras for polynomial functors.

Proof ingredients

- Reduce dialgebras to algebras/coalgebras
- Dependent polynomial functors are closed under taking fixed points
- Fixed points of dependent polynomials can be constructed from non-dependent
- Final coalgebras of polynomials can be constructed from initial algebras

How to obtain models?

Theorem

Dependent, strictly positive data types can be interpreted in a locally Cartesian closed category with dependent products and disjoint coproducts if it has initial algebras for polynomial functors.

Proof ingredients

- Reduce dialgebras to algebras/coalgebras
- Dependent polynomial functors are closed under taking fixed points
- Fixed points of dependent polynomials can be constructed from non-dependent
- Final coalgebras of polynomials can be constructed from initial algebras

For details see

Dependent Inductive and Coinductive Types are Fibrational Dialgebras, H.B., FICS 2015.

Upcoming

Handling dependencies

Parameter contexts and instantiations

$$\Theta \mid \Gamma_1 \vdash A : \Gamma_2 \multimap * ,$$

Suppose that $\Gamma_2 = x_1 : B_1, \ldots, x_n : B_n$ and $\Gamma_1 \vdash t_k : B_k$, then

$$\Theta \mid \Gamma_1 \vdash A \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n : * .$$

Type constructor variables

$$X : \Gamma_2 \multimap * \mid \Gamma_1 \vdash X : \Gamma_2 \multimap * ,$$

which we can instantiate to

$$X : \Gamma_2 \multimap * \mid \Gamma_1 \vdash X \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n : * .$$

Forming types

$$\frac{k = 1, \ldots, n \qquad \sigma_k : \Gamma_k \to \Gamma \qquad \Theta, X : \Gamma \multimap * \mid \Gamma_k \vdash A_k : *}{\Theta \mid \emptyset \vdash \rho(X : \Gamma \multimap * ; \vec{\sigma}; \vec{A}) : \Gamma \multimap *}$$

- $n \in \mathbb{N}$
- $\rho \in \{\mu, \nu\}$
- $\sigma_k : \Gamma_k \to \Gamma$ substitution of terms in context Γ_k for variables in Γ
- A_k type with extra free variable

Examples I/II

Example (Vectors)

data Vec (A : **Set**) : $\mathbb{N} \to$ **Set where**
  nil : $\top \to$ Vec 0
  cons : (n : $\mathbb{N}$) $\to$ A $\times$ Vec $n \to$ Vec $(n+1)$

$$\mathsf{Vec}\, A := \mu(X : \Gamma \rightarrowtriangle * ; (\sigma_1, \sigma_2); (\mathbf{1}, A \times X \mathbin{@} k))$$

$$\Gamma = n : \mathsf{Nat} \quad \text{and} \quad \Gamma_1 = \emptyset \quad \text{and} \quad \Gamma_2 = k : \mathsf{Nat}$$

$$\sigma_1 = (0) : \Gamma_1 \to (n : \mathsf{Nat}) \quad \text{and} \quad \sigma_2 = (s \mathbin{@} k) : \Gamma_2 \to (n : \mathsf{Nat})$$

$$X : (n : \mathsf{Nat}) \rightarrowtriangle * \mid \Gamma_1 \vdash \mathbf{1} : *$$

$$X : (n : \mathsf{Nat}) \rightarrowtriangle * \mid \Gamma_2 \vdash A \times X \mathbin{@} k : *$$

Examples I/II

Example (Vectors)

data Vec (A : **Set**) : $\mathbb{N} \to$ **Set where**
  nil : $\top \to$ Vec 0
  cons : (n : $\mathbb{N}$) $\to$ A $\times$ Vec $n \to$ Vec $(n+1)$

$$\mathsf{Vec}\, A := \mu(X : \Gamma \rightarrowtriangle * ; (\sigma_1, \sigma_2); (\mathbf{1}, A \times X \mathbin{@} k))$$

$$\Gamma = n : \mathsf{Nat} \quad \text{and} \quad \Gamma_1 = \emptyset \quad \text{and} \quad \Gamma_2 = k : \mathsf{Nat}$$

$$\sigma_1 = (0) : \Gamma_1 \to (n : \mathsf{Nat}) \quad \text{and} \quad \sigma_2 = (s \mathbin{@} k) : \Gamma_2 \to (n : \mathsf{Nat})$$

$$X : (n : \mathsf{Nat}) \rightarrowtriangle * \mid \Gamma_1 \vdash \mathbf{1} : *$$

$$X : (n : \mathsf{Nat}) \rightarrowtriangle * \mid \Gamma_2 \vdash A \times X \mathbin{@} k : *$$

Examples I/II

Example (Vectors)

data Vec (A : **Set**) : $\mathbb{N} \to$ **Set where**
  nil : $\top \to$ Vec 0
  cons : (n : $\mathbb{N}$) $\to$ A $\times$ Vec $n \to$ Vec $(n+1)$

$$\mathsf{Vec}\, A := \mu(X : \Gamma \rightarrowtriangle * ; (\sigma_1, \sigma_2); (\mathbf{1}, A \times X \mathbin{@} k))$$

$$\Gamma = n : \mathsf{Nat} \quad \text{and} \quad \Gamma_1 = \emptyset \quad \text{and} \quad \Gamma_2 = k : \mathsf{Nat}$$

$$\sigma_1 = (0) : \Gamma_1 \to (n : \mathsf{Nat}) \quad \text{and} \quad \sigma_2 = (s \mathbin{@} k) : \Gamma_2 \to (n : \mathsf{Nat})$$

$$X : (n : \mathsf{Nat}) \rightarrowtriangle * \mid \Gamma_1 \vdash \mathbf{1} : *$$

$$X : (n : \mathsf{Nat}) \rightarrowtriangle * \mid \Gamma_2 \vdash A \times X \mathbin{@} k : *$$

Examples II/II

- Π is a coinductive type
- Existential quantification is its dual
- All of intuitionistic predicate logic can be represented
- Partial streams

Forming terms

For inductive types

$$\frac{\vdash \mu(X : \Gamma \to * ; \vec{\sigma}; \vec{A}) : \Gamma \to * \qquad 1 \leq k \leq n}{\vdash \alpha_k^{\mu(X:\Gamma\to *;\vec{\sigma};\vec{A})} : (\Gamma_k, y : A_k[\mu/X]) \to \mu \mathbin{@} \sigma_k}\ \textbf{(Ind-I)}$$

$$\frac{\vdash C : \Gamma \to * \qquad \Delta, \Gamma_k, y_k : A_k[C/X] \vdash g_k : (C \mathbin{@} \sigma_k) \qquad \forall k = 1, \ldots, n}{\Delta \vdash \mathsf{rec}\ \overrightarrow{(\Gamma_k, y_k).\, g_k} : (\Gamma, y : \mu \mathbin{@} \mathsf{id}_\Gamma) \to C \mathbin{@} \mathsf{id}_\Gamma}$$

Forming terms

For inductive types

$$\frac{\vdash \mu(X : \Gamma \to * ; \vec{\sigma}; \vec{A}) : \Gamma \to * \qquad 1 \leq k \leq n}{\vdash \alpha_k^{\mu(X:\Gamma\to *;\vec{\sigma};\vec{A})} : (\Gamma_k, y : A_k[\mu/X]) \to \mu \mathbin{@} \sigma_k} \textbf{(Ind-I)}$$

$$\frac{\vdash C : \Gamma \to * \qquad \Delta, \Gamma_k, y_k : A_k[C/X] \vdash g_k : (C \mathbin{@} \sigma_k) \qquad \forall k = 1, \ldots, n}{\Delta \vdash \text{rec } \overrightarrow{(\Gamma_k, y_k).\, g_k} : (\Gamma, y : \mu \mathbin{@} \text{id}_\Gamma) \to C \mathbin{@} \text{id}_\Gamma}$$

Dual for coinductive types

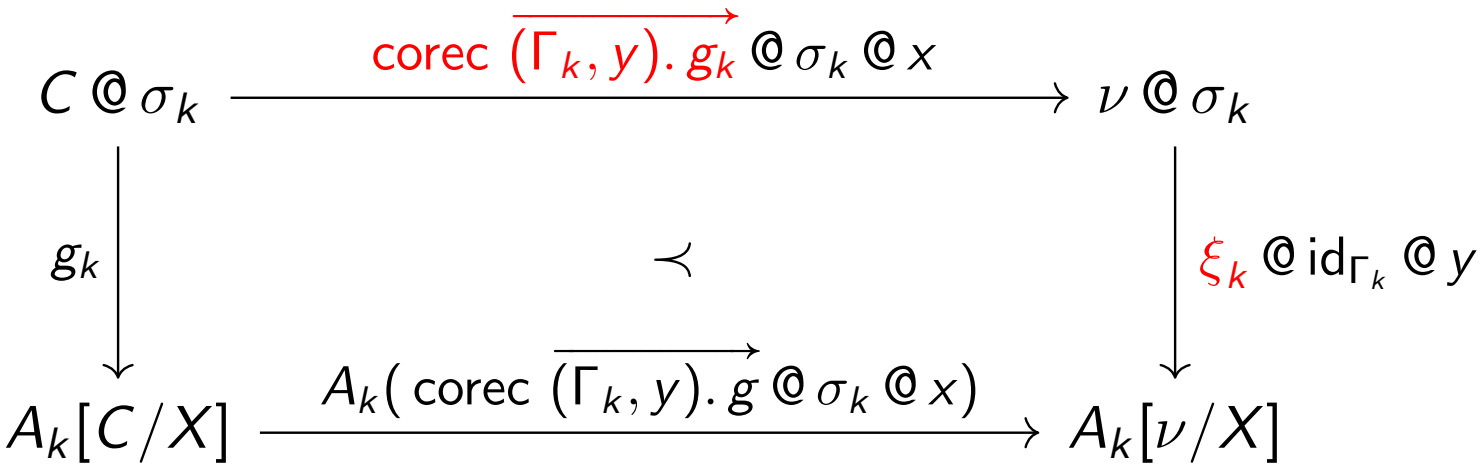
$$\frac{\vdash \nu(X : \Gamma \to * ; \vec{\sigma}; \vec{A}) : \Gamma \to * \qquad 1 \leq k \leq n}{\vdash \xi_k^{\nu(X:\Gamma\to *;\vec{\sigma};\vec{A})} : (\Gamma_k, y : \nu \mathbin{@} \sigma_k) \to A_k[\nu/X]} \textbf{(Coind-E)}$$

$$\frac{\vdash C : \Gamma \to * \qquad \Delta, \Gamma_k, y_k : (C \mathbin{@} \sigma_k) \vdash g_k : A_k[C/X] \qquad \forall k = 1, \ldots, n}{\Delta \vdash \text{corec } \overrightarrow{(\Gamma_k, y_k).\, g_k} : (\Gamma, y : C \mathbin{@} \text{id}_\Gamma) \to \nu \mathbin{@} \text{id}_\Gamma}$$

Computations

Reduction

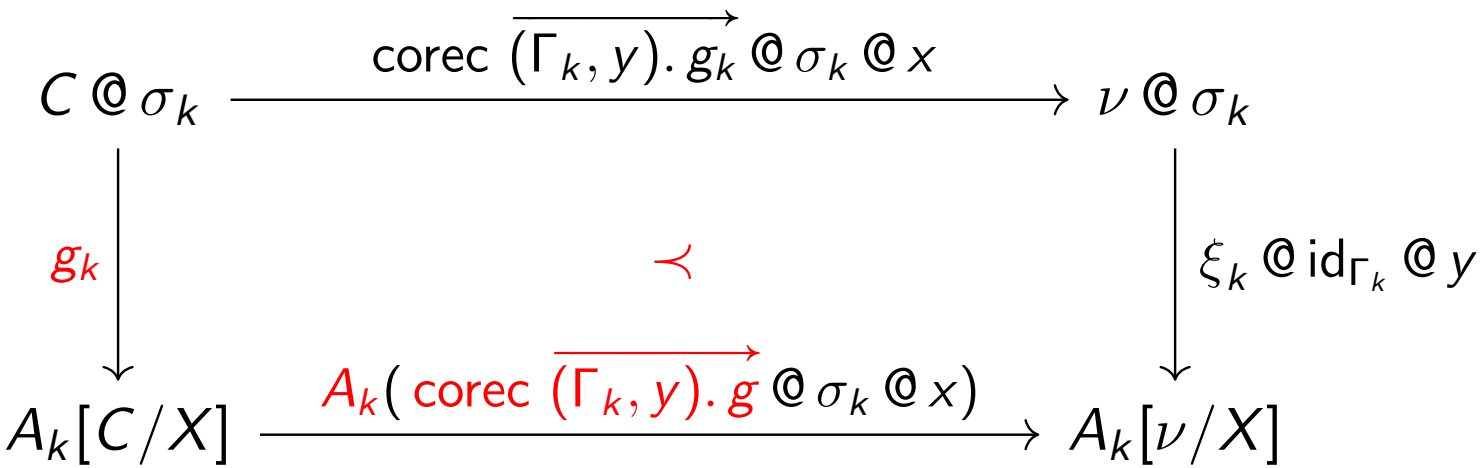
- Defined as closure of contraction relation
- Essentially follows homomorphism diagrams
- Preserves types



Computations

Reduction

- Defined as closure of contraction relation
- Essentially follows homomorphism diagrams
- Preserves types



Theorem

The reduction relation is strongly normalising.

Theorem

The reduction relation is strongly normalising.

Proof Outline

- Define saturated set model for types

Theorem

The reduction relation is strongly normalising.

Proof Outline

- Define saturated set model for types
- Set X of terms is saturated if
  - $X \subseteq \mathbf{SN}$
  - contains neutral terms (no reductions possible)
  - backwards closed under eliminators, i.e., destructors and recursion

Theorem

The reduction relation is strongly normalising.

Proof Outline

- Define saturated set model for types
- Set X of terms is saturated if
  - $X \subseteq \mathbf{SN}$
  - contains neutral terms (no reductions possible)
  - backwards closed under eliminators, i.e., destructors and recursion
- Dependent types are interpreted families of saturated sets

Theorem

The reduction relation is strongly normalising.

Proof Outline

- Define saturated set model for types
- Set X of terms is saturated if
  - $X \subseteq \mathbf{SN}$
  - contains neutral terms (no reductions possible)
  - backwards closed under eliminators, i.e., destructors and recursion
- Dependent types are interpreted families of saturated sets
- Show soundness: If $\Gamma \vdash t : A$, then $\forall \rho \in [\![\Gamma]\!].t \in [\![A]\!]_\rho$.

Theorem

The reduction relation is strongly normalising.

Proof Outline

- Define saturated set model for types
- Set X of terms is saturated if
  - $X \subseteq \mathbf{SN}$
  - contains neutral terms (no reductions possible)
  - backwards closed under eliminators, i.e., destructors and recursion
- Dependent types are interpreted families of saturated sets
- Show soundness: If $\Gamma \vdash t : A$, then $\forall \rho \in [\![\Gamma]\!].t \in [\![A]\!]_\rho$.

For details see

'Type Theory based on Dependent Inductive and Coinductive Types', H.B. and H. Geuvers, in LICS'16, 2016 and arXiv:CS.LO/1605.02206 for proofs.

Upcoming

The Good

- Inductive and coinductive types exactly dual
- No special type constructors for function space etc. necessary

The Good

- Inductive and coinductive types exactly dual
- No special type constructors for function space etc. necessary

The Bad

- No induction principle – though this can be easily added
- But what about coinduction then?

The Good

- Inductive and coinductive types exactly dual
- No special type constructors for function space etc. necessary

The Bad

- No induction principle – though this can be easily added
- But what about coinduction then?

The Ugly

- Silent introduction of propositional equality as inductive type
- Thus no good way of dealing with coinduction

The Good

- Inductive and coinductive types exactly dual
- No special type constructors for function space etc. necessary

The Bad

- No induction principle – though this can be easily added
- But what about coinduction then?

The Ugly

- Silent introduction of propositional equality as inductive type
- Thus no good way of dealing with coinduction

The Fix

- Blend Observational Type Theory, Cubical TT and Dependent Inductive-Coinductive Types (inspired by Conor McBride)
- See my talk at TYPES'16 and, hopefully, my thesis

Blending OTT, Cubical TT and Dependent Inductive-Coinductive Types

Thank you for the inspiration Conor![1]

[1] McBride, that is. *A Cubical Crossroads* @ Agda Implementors' Meeting XXIII.

Plan

- Remove possibility to introduce propositional equality
- Paths between types through abstraction over points in interval (Cubical)
- Coercion operator (OTT)
- Heterogeneous path type (OTT)
- Coinduction: Constructor for paths

Removing Equality

Easy: Replace ...

$$\frac{k = 1, \ldots, n \qquad \sigma_k : \Gamma_k \to \Gamma \qquad \Theta, X : \Gamma \rightarrowtriangle * \mid \Gamma_k \vdash A_k : *}{\Theta \mid \emptyset \vdash \rho(X : \Gamma \rightarrowtriangle * ; \vec{\sigma}; \vec{A}) : \Gamma \rightarrowtriangle *}$$

... by

$$\frac{k = 1, \ldots, n \qquad \Theta, X : \Gamma \rightarrowtriangle * \mid \Gamma, \Gamma_k \vdash A_k : *}{\Theta \mid \emptyset \vdash \rho(X : \Gamma \rightarrowtriangle * ; \vec{A}) : \Gamma \rightarrowtriangle *}$$

Interval, Paths between Types and Coercion

$$\frac{}{\Gamma \vdash 0 : \mathbb{I}} \qquad \frac{}{\Gamma \vdash 1 : \mathbb{I}} \qquad \frac{i \in \Delta}{\Gamma \mid \Delta \vdash i : \mathbb{I}} \qquad \frac{\vdash p, q, r : \mathbb{I}}{\vdash p(q - r) : \mathbb{I}}$$

$$\frac{\Gamma \vdash S : * \qquad \Gamma \vdash T : *}{\Gamma \vdash S \sim T : *} \qquad \frac{\vdash Q : S \sim T \qquad \vdash p : \mathbb{I}}{\vdash Q\, p : *}$$

$$\frac{\Gamma \mid \Delta, i : \mathbb{I} \vdash M : * \qquad M[0/i] \equiv S \qquad M[1/i] \equiv T}{\Gamma \mid \Delta \vdash \langle i \rangle .\, M : S \sim T}$$

$$\frac{\vdash Q : S \sim T \qquad \vdash p, q : \mathbb{I} \qquad \vdash s : Q\, p}{\vdash s\, [p \mid Q \mid q\rangle : Q\, q}$$

Paths between terms

$$\frac{\vdash s : S \qquad \vdash t : T}{\vdash s \sim t : *} \qquad \frac{\vdash s, t : T \qquad \vdash Q : s \sim t \qquad \vdash p : \mathbb{I}}{\Gamma \vdash Q\, p : T}$$

$$\frac{\vdash t : T}{\vdash \mathsf{refl}_t : t \sim t}$$

Paths between terms

$$\frac{\vdash s : S \qquad \vdash t : T}{\vdash s \sim t : *} \qquad \frac{\vdash s, t : T \qquad \vdash Q : s \sim t \qquad \vdash p : \mathbb{I}}{\Gamma \vdash Q\, p : T}$$

$$\frac{\vdash t : T}{\vdash \mathsf{refl}_t : t \sim t}$$

Coinduction (for non-dependent types):

$$R : (x : \nu, y : \nu) \multimap *$$

$$\frac{x_k : \nu, y_k : \nu, z_k : R \mathbin{@} x_k \mathbin{@} y_k \vdash g_k : \overline{A_k}[R][\xi_k \mathbin{@} x_k / x, \xi_k \mathbin{@} y_k / y]}{\vdash \mathsf{coind}\ \overrightarrow{(x_k, y_k, z_k).\, g_k} : (x : \nu, y : \nu, z : R \mathbin{@} x \mathbin{@} y) \multimap x \sim y}$$

Paths between terms

$$\frac{\vdash s : S \qquad \vdash t : T}{\vdash s \sim t : *} \qquad \frac{\vdash s, t : T \qquad \vdash Q : s \sim t \qquad \vdash p : \mathbb{I}}{\Gamma \vdash Q\, p : T}$$

$$\frac{\vdash t : T}{\vdash \mathsf{refl}_t : t \sim t}$$

Coinduction (for non-dependent types):

$$R : (x : \nu, y : \nu) \to *$$

$$\frac{x_k : \nu, y_k : \nu, z_k : R \mathbin{@} x_k \mathbin{@} y_k \vdash g_k : \overline{A_k}[R][\xi_k \mathbin{@} x_k / x, \xi_k \mathbin{@} y_k / y]}{\vdash \mathsf{coind}\ \overrightarrow{(x_k, y_k, z_k).\, g_k} : (x : \nu, y : \nu, z : R \mathbin{@} x \mathbin{@} y) \to x \sim y}$$

Remark

Using bisimulations is rather brute-force. Maybe better: later modality.

Reductions

$$0(p - q) \succ p$$

$$1(p - q) \succ q$$

$$(\langle i \rangle . M)\, p \succ M[p/i]$$

$$\text{If } Q : s \sim t, \text{then } Q\,0 \succ s$$

$$\text{If } Q : s \sim t, \text{then } Q\,1 \succ t$$

$$s\,[0 \mid Q \mid 0\rangle \succ s$$

$$s\,[1 \mid Q \mid 1\rangle \succ s$$

$$(\alpha_k \mathbin{@} t)[0 \mid \langle i \rangle . \mu(X : * ; \overrightarrow{A[i]}) \mid 1\rangle \succ \alpha_k \mathbin{@} (t[0 \mid \langle i \rangle . A_k[i] \mid 1\rangle)$$

$$\left(\text{corec } \overrightarrow{(y_k) . g_k} \mathbin{@} t\right)[0 \mid \langle i \rangle . \nu(X : * ; \overrightarrow{A[i]}) \mid 1\rangle \succ (\text{corec } \overrightarrow{(y_k) . g'_k} \mathbin{@} t')$$

Example (Function Extensionality)

Define

$$R : (h : A \to B, k : A \to B) \rightarrowtriangle *$$
$$R := (h).\,(k).\,\Pi x : A.h\,x \sim k\,x$$

and

$$\frac{\dfrac{x : A, h, k : A \to B, z : R\,@\,h\,@\,k \vdash z : \Pi x : A.h\,x \sim k\,x}{x : A, h, k : A \to B, z : R\,@\,h\,@\,k \vdash z\,x : h\,x \sim k\,x}}{f, g, p : \Pi x : A.f\,x \sim g\,x \vdash \mathsf{coind}\,((x, h, k, z).\,z\,x)\,@\,f\,@\,g\,@\,p : f \sim g}$$

Thank you very much for your attention!