Towards Formal Foundations of Agda's Data Types

Henning Basold
Radboud University, Nijmegen and CWI, Amsterdam

Joint work with Herman Geuvers

AIM XXIII
20 April 2016

Outline

Introduction

It bothered me for a while that

- there is no type theory with arbitrary mixed inductive-coinductive, dependent types.
- people keep hold on coinductive types à la Coq.
- I never understood the difference between term parameters and indices.

That led me to

- extract the categorical principles behind such types.
- build a type theory resembling the categorical principles.

Relation to Agda

- With some restrictions, the data types of Agda can be directly represented.
- Not yet treated: polymorphism, sized types, universes.

Example

data Vec (A : **Set**) : $\mathbb{N} \rightarrow$ **Set where**
nil : $\top \rightarrow$ Vec 0
cons : (n : $\mathbb{N}^\infty$) $\rightarrow$ A $\times$ Vec $n \rightarrow$ 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: $\mathrm{Vec}\, A = \{A^n\}_{n \in \mathbb{N}}$
- $\mathrm{nil} = \{\mathrm{nil}_*\}$: $\{\mathbf{1}\}_{* \in \mathbf{1}} \rightarrow \left\{A^0\right\}_{* \in \mathbf{1}}$
- $\mathrm{cons} = \{\mathrm{cons}_n\}_{n \in \mathbb{N}} : \{A \times A^n\}_{n \in \mathbb{N}} \rightarrow \left\{A^{n+1}\right\}_{n \in \mathbb{N}}$

Dependent Data Types are Initial/Final Dialgebras

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

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

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

General Setup

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 $J_1, \dots, J_n \in \mathbf{B}$
- 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 $u_1, \dots, u_n$ with $u_k : J_n \to I$ giving the substitutions in the result type of the constructors
- An initial dialgebra $(c_1, \dots, c_n) : F(A) \to G_u(A)$, 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 $(\text{nil}, \text{cons}) : F(\text{Vec}\, A) \to G(\text{Vec}\, A)$ 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

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.

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

How to obtain models?

Theorem

Dependent, strictly positive data types can be interpreted in a locally Cartesian closed category with dependent products 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.

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 : $\mathbb{N}^\infty$) $\to$ A $\times$ Vec $n \to$ Vec $(n+1)$

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

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

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

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

$$X : (n : \mathrm{Nat}) \multimap * \mid \Gamma_2 \vdash A \times X \,@\, k : *$$

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

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

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

Dual for coinductive types

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

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

Computations

Reduction

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

Theorem

The reduction relation is strongly normalising.

For details see

- 'Type Theory based on Dependent Inductive and Coinductive Types', H.B. and Herman Geuvers, in LICS'16, 2016.
- Partial implementation in Agda

Missing

- Missing features: polymorphism, sized types, universes.
- Missing in SN proof: induction.
- Translation of Agda's terms – need to deal with (co)patterns
- Sized types – Some first steps for the categorical logic

Future

- Bisimilarity can be defined canoncially – for function space this is extensionality
- Use this to obtain an OTT or treat like HITs
- For the category theorists: generalise to programming on finitary functors (even more general: accessible)

Thank you very much for your attention!