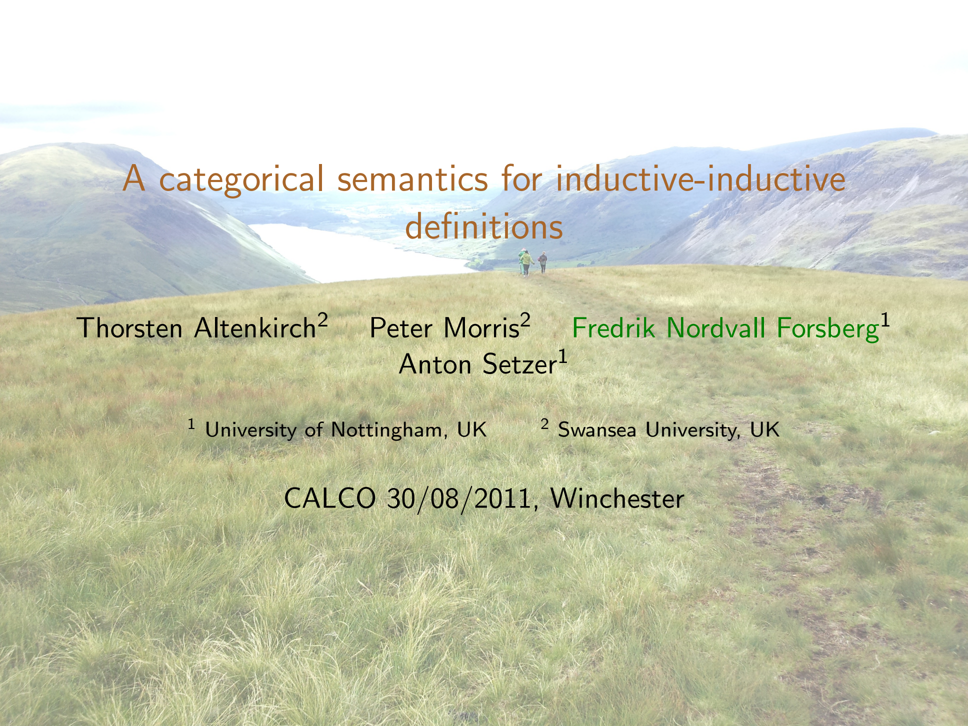

A categorical semantics for inductive-inductive definitions

Thorsten Altenkirch[2] Peter Morris[2] Fredrik Nordvall Forsberg[1]
Anton Setzer[1]

[1] University of Nottingham, UK [2] Swansea University, UK

CALCO 30/08/2011, Winchester

Plan of the talk

1. What are inductive-inductive definitions?
2. How can they be described, categorically?
3. Exploiting initiality.

Notation

- Work in the framework of Martin-Löf type theory.
- Unit type $\mathbf{1}$, disjoint union $A + B$.
- Dependent function spaces $(x : A) \to B(x)$.
  - Elements are functions f such that $f(a) : B(a)$ whenever $a : A$.
- Dependent pairs $\Sigma x\!:\!A.B(x)$.
  - Elements are pairs $\langle a, b\rangle$ where $a : A$, $b : B(a)$.
  - Projections $\pi_0 : \Sigma x\!:\!A.B(x) \to A$ and $\pi_1 : (y : \Sigma A\ B) \to B(\pi_0(y))$.
- Set the type of (small) types / propositions.

Inductive-inductive definitions

What is an inductive-inductive definition?

- Induction-induction is a principle for defining datatypes $A : \mathsf{Set}$, $B : A \to \mathsf{Set}$.
- Both A and B are defined inductively, i.e. “built up from below”.

What is an inductive-inductive definition?

- Induction-induction is a principle for defining datatypes $A : \mathsf{Set}$, $B : A \to \mathsf{Set}$.
- Both A and B are defined inductively, i.e. “built up from below”.
- A and B are defined simultaneously, so the constructors for A can refer to B and vice versa.
- In addition, the constructors for B can even refer to the constructors for A.

But isn't that. . . ?

An inductive-inductive definition is in general not:

- An ordinary inductive definition
    - Because we define A : Set and $B : A \to$ Set simultaneously.

But isn't that...?

An inductive-inductive definition is in general not:

- An ordinary inductive definition
  - Because we define $A : \text{Set}$ and $B : A \to \text{Set}$ simultaneously.
- An ordinary mutual inductive definition
  - Because $B : A \to \text{Set}$ is indexed by A.

But isn't that. . . ?

An inductive-inductive definition is in general not:

- An ordinary inductive definition
  - Because we define A : Set and $B : A \to$ Set simultaneously.
- An ordinary mutual inductive definition
  - Because $B : A \to$ Set is indexed by A.
- An indexed inductive definition
  - Because the index set A : Set is defined along with $B : A \to$ Set, and not fixed beforehand.
  - However, conjecture that it can be reduced to IID.

But isn't that. . . ?

An inductive-inductive definition is in general not:

- An ordinary inductive definition
  - Because we define A : Set and $B : A \to$ Set simultaneously.
- An ordinary mutual inductive definition
  - Because $B : A \to$ Set is indexed by A.
- An indexed inductive definition
  - Because the index set A : Set is defined along with $B : A \to$ Set, and not fixed beforehand.
  - However, conjecture that it can be reduced to IID.
- An inductive-recursive definition
  - Because $B : A \to$ Set is defined inductively, not recursively.

Induction-recursion vs induction-induction

- **Inductive-recursive definition:** Need to define $B(c(\vec{x}))$ completely when introducing $c(\vec{x})$.
  - For each constructor c of A, must define $B(c(\vec{x})) = \ldots B \ldots$.
  - But can refer to $B(x)$ both positively and negatively in type of c.
  - Example: $B(\sigma(s,t)) = \Sigma\, x : B(s)\,.\, B(t(x))$.
- **Inductive-inductive definition:** Elements of $B(x)$ can be defined any time after x is introduced.
  - So might depend on elements introduced after x.
  - We can refer to $B(x)$ only positively.
  - Example: $B : A \rightarrow \mathrm{Set}$ where $d : (x : A) \rightarrow (y : B(x)) \rightarrow B(c(x,y))$.

An example

Instances of induction-induction have been used implicitly by

- Dybjer (Internal type theory, 1996),
- Danielsson (A formalisation of a dependently typed language as an inductive-recursive family, 2007), and
- Chapman (Type theory should eat itself, 2009)

to model dependent type theory inside itself.

Type theory inside type theory

- Context : Set
- Type : Context → Set
- Term : (Γ : Context) → Type(Γ) → Set

defined inductively

- . . .
- Substitutions, . . .
- . . .

The crucial point

- The empty context ε is a well-formed context.
- If τ is a well-formed type in context Γ, then $\Gamma, x : \tau$ is a well-formed context.

$$\frac{}{\varepsilon : \mathsf{Context}}$$

$$\frac{\Gamma : \mathsf{Context} \qquad \tau : \textcolor{red}{\mathsf{Type}(\Gamma)}}{\Gamma \triangleright \tau : \mathsf{Context}}$$

Constructor for Type referring to constructor for Context

$$\frac{\Gamma \text{ context} \qquad \Gamma \vdash \sigma \text{ type} \qquad \Gamma, x : \sigma \vdash \tau(x) \text{ type}}{\Gamma \vdash \Pi\, x{:}\sigma\,.\,\tau(x) \text{ type}}$$

Constructor for Type referring to constructor for Context

$$\frac{\Gamma \text{ context} \qquad \Gamma \vdash \sigma \text{ type} \qquad \Gamma, x : \sigma \vdash \tau(x) \text{ type}}{\Gamma \vdash \Pi\, x{:}\sigma\,.\,\tau(x) \text{ type}}$$

$$\frac{\Gamma : \text{Context}}{}$$

Constructor for Type referring to constructor for Context

$$\frac{\Gamma\ \text{context} \qquad \Gamma \vdash \sigma\ \text{type} \qquad \Gamma, x : \sigma \vdash \tau(x)\ \text{type}}{\Gamma \vdash \Pi\, x{:}\sigma\,.\,\tau(x)\ \text{type}}$$

$$\frac{\Gamma : \text{Context} \qquad \sigma : \text{Type}(\Gamma)}{}$$

Constructor for Type referring to constructor for Context

$$\frac{\Gamma \text{ context} \qquad \Gamma \vdash \sigma \text{ type} \qquad \Gamma, x : \sigma \vdash \tau(x) \text{ type}}{\Gamma \vdash \Pi\, x : \sigma \,.\, \tau(x) \text{ type}}$$

$$\frac{\Gamma : \text{Context} \qquad \sigma : \text{Type}(\Gamma) \qquad \tau : \text{Type}(\Gamma \triangleright \sigma)}{}$$

Constructor for Type referring to constructor for Context

$$\frac{\Gamma\ \text{context} \qquad \Gamma \vdash \sigma\ \text{type} \qquad \Gamma, x : \sigma \vdash \tau(x)\ \text{type}}{\Gamma \vdash \Pi\, x{:}\sigma\,.\,\tau(x)\ \text{type}}$$

$$\frac{\Gamma : \text{Context} \qquad \sigma : \text{Type}(\Gamma) \qquad \tau : \text{Type}(\Gamma \triangleright \sigma)}{}$$

Constructor for Type referring to constructor for Context

$$\frac{\Gamma \text{ context} \qquad \Gamma \vdash \sigma \text{ type} \qquad \Gamma, x : \sigma \vdash \tau(x) \text{ type}}{\Gamma \vdash \Pi\, x{:}\sigma \,.\, \tau(x) \text{ type}}$$

$$\frac{\Gamma : \mathsf{Context} \qquad \sigma : \mathsf{Type}(\Gamma) \qquad \tau : \mathsf{Type}(\Gamma \triangleright \sigma)}{\Pi(\sigma, \tau) : \mathsf{Type}(\Gamma)}$$

But is there a theory?

- Previous work: axiomatisation.
- Now: initial algebra like semantics – less ugly details.

Describing inductive-inductive datatypes

Initial algebra semantics

- Let $F : \mathbb{C} \to \mathbb{C}$ be a functor. Recall that an F-algebra is a pair (X, f) where $X \in \mathbb{C}$ and $f : F(X) \to X$.

- A morphism $\alpha : (X, f) \to (Y, g)$ between F-algebras is a morphism $\alpha : X \to Y$ such that

$$\begin{tikzcd}
F(X) \arrow[r, "f"] \arrow[d, "F(\alpha)"'] & X \arrow[d, "\alpha"] \\
F(Y) \arrow[r, "g"'] & Y
\end{tikzcd}$$

- Model inductive data types as initial F-algebra for suitable endofunctor F. (F "represents" the data type by describing its constructors.)

- Example: $F(X) = \mathbf{1} + (X \times X)$, $[\texttt{empty}, \texttt{node}] : F(\texttt{BTree}) \to \texttt{BTree}$.

Induction-induction as initial algebras?

- In general, an inductive-inductive definition consists of
  - $A : \mathsf{Set}$,
  - $B : A \to \mathsf{Set}$,
  - a constructor $\mathsf{in}_A : \mathrm{Arg}_\mathrm{A}(A, B) \to A$ for A,
  - a constructor $\mathsf{in}_B : (x : \mathrm{Arg}_\mathrm{A}(A, B)) \to \mathrm{Arg}_\mathrm{B}(A, B, \mathsf{in}_A) \to B(\mathsf{in}_A(x))$ for B

  for some functors Arg_A, Arg_B (but from and to where?).

Induction-induction as initial algebras?

- In general, an inductive-inductive definition consists of
  - $A : \mathsf{Set}$,
  - $B : A \to \mathsf{Set}$,
  - a constructor $\mathsf{in}_A : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A$ for A,
  - a constructor $\mathsf{in}_B : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, \mathsf{in}_A) \to B(\mathsf{in}_A(x))$ for B

  for some functors $\mathrm{Arg}_{\mathrm{A}}$, $\mathrm{Arg}_{\mathrm{B}}$ (but from and to where?).

- First thought: an inductive-inductive def. is a family (A, B) of sets, so they should be represented by functors

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set}).$$

Induction-induction as initial algebras?

- In general, an inductive-inductive definition consists of
  - $A : \mathsf{Set}$,
  - $B : A \to \mathsf{Set}$,
  - a constructor $\mathsf{in}_A : \mathrm{Arg_A}(A, B) \to A$ for A,
  - a constructor $\mathsf{in}_B : (x : \mathrm{Arg_A}(A, B)) \to \mathrm{Arg_B}(A, B, \mathsf{in}_A) \to B(\mathsf{in}_A(x))$ for B

  for some functors $\mathrm{Arg_A}$, $\mathrm{Arg_B}$ (but from and to where?).

- First thought: an inductive-inductive def. is a family (A, B) of sets, so they should be represented by functors

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set}).$$

- Here, $\mathbf{Fam}(\mathbf{Set})$ category with objects (A, B) where $A : \mathsf{Set}$, $B : A \to \mathsf{Set}$.

- Morphism (A, B) to (A', B') is (f, g) where $f : A \to A'$, $g : (x : A) \to B(x) \to B'(f(x))$.

Induction-induction as initial algebras?

- In general, an inductive-inductive definition consists of
  - $A : \text{Set}$,
  - $B : A \to \text{Set}$,
  - a constructor $\text{in}_A : \mathrm{Arg}_\mathrm{A}(A, B) \to A$ for A,
  - a constructor $\text{in}_B : (x : \mathrm{Arg}_\mathrm{A}(A, B)) \to \mathrm{Arg}_\mathrm{B}(A, B, \text{in}_A) \to B(\text{in}_A(x))$ for B

  for some functors Arg_A, Arg_B (but from and to where?).

- First thought: an inductive-inductive def. is a family (A, B) of sets, so they should be represented by functors

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set}).$$

$$\begin{tikzcd}
\mathrm{Arg}_\mathrm{A} \arrow[d, bend right, "?"'] & \mathrm{Arg}_\mathrm{B} \arrow[d, bend left, "?"] \\
F_0 & F_1
\end{tikzcd}$$

- Here, $\mathbf{Fam}(\mathbf{Set})$ category with objects (A, B) where $A : \text{Set}$, $B : A \to \text{Set}$.

- M

  $g : (x : A) \to B(x) \to B'(f(x))$.

Every endofunctor F on $\mathbf{Fam}(\mathbf{Set})$ can be split up into
$F_0 : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Set}$, $F_1 : (X : \mathbf{Fam}(\mathbf{Set})) \to F_0(X) \to \mathbf{Set}$.

Induction-induction as initial algebras?

- In general, an inductive-inductive definition consists of
  - $A : \text{Set}$,
  - $B : A \to \text{Set}$,
  - a constructor $\text{in}_A : \text{Arg}_\text{A}(A, B) \to A$ for A,
  - a constructor $\text{in}_B : (x : \text{Arg}_\text{A}(A, B)) \to \text{Arg}_\text{B}(A, B, \text{in}_A) \to B(\text{in}_A(x))$ for B

  for some functors Arg_A, Arg_B (but from and to where?).

- First thought: an inductive-inductive def. is a family (A, B) of sets, so they should be represented by functors

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set}).$$

- Here, $\mathbf{Fam}(\mathbf{Set})$ category with objects (A, B) where $A : \text{Set}$, $B : A \to \text{Set}$.

- Morphism (A, B) to (A', B') is (f, g) where $f : A \to A'$, $g : (x : A) \to B(x) \to B'(f(x))$.

Induction-induction as initial algebras? (cont.)

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set})$$

- An F-algebra $((A, B), (c, d))$ would have “constructors” $c : F_0(A, B) \to A$ and $d : (x : F_0(A, B)) \to F_1(A, B, x) \to B(c(x))$.

Induction-induction as initial algebras? (cont.)

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set})$$

- An F-algebra $((A, B), (c, d))$ would have "constructors" $c : F_0(A, B) \to A$ and $d : (x : F_0(A, B)) \to F_1(A, B, x) \to B(c(x))$.

- But then the constructor d for B cannot refer to the constructor c for A! [Necessary for the Π type example]

Induction-induction as initial algebras? (cont.)

$$F = (F_0, F_1) : \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Fam}(\mathbf{Set})$$

- An F-algebra $((A, B), (c, d))$ would have "constructors" $c : F_0(A, B) \to A$ and $d : (x : F_0(A, B)) \to F_1(A, B, x) \to B(c(x))$.
- But then the constructor d for B cannot refer to the constructor c for A! [Necessary for the Π type example]
- Instead, we would like

$$F_1' : ((A, B) : \mathbf{Fam}(\mathbf{Set}), c : F_0(A, B) \to A) \to F_0(A, B) \to \mathbf{Set}.$$

(what we have is

$$F_1 : ((A, B) : \mathbf{Fam}(\mathbf{Set})) \to F_0(A, B) \to \mathbf{Set}.)$$

Contexts and types described this way

$$\frac{}{\varepsilon : \mathsf{Context}} \qquad \frac{\Gamma : \mathsf{Context} \qquad \sigma : \mathsf{Type}(\Gamma)}{\Gamma \rhd \sigma : \mathsf{Context}}$$

$$\frac{\Gamma : \mathsf{Context}}{\iota_\Gamma : \mathsf{Type}(\Gamma)} \qquad \frac{\Gamma : \mathsf{Context} \qquad \sigma : \mathsf{Type}(\Gamma) \qquad \tau : \mathsf{Type}(\Gamma \rhd \sigma)}{\Pi(\sigma, \tau) : \mathsf{Type}(\Gamma)}$$

$$\begin{aligned} \mathsf{Arg}_{\mathsf{Context}}(A, B) &= \mathbf{1} + \Sigma\, \Gamma : A.\, B(\Gamma) \\ \mathsf{Arg}_{\mathsf{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\, \sigma : B(c(x)).\, \tau : B(c(\mathsf{inr}(c(x), \sigma)))) \ . \end{aligned}$$

Note ‘Γ : Context’ replaced by ‘$c(x)$’ for $x : \mathsf{Arg}_{\mathsf{Context}}(\mathsf{Context}, \mathsf{Type})$ in $\mathsf{Arg}_{\mathsf{Type}}$.

Can be combined into

$$\mathsf{Arg} : ((A, B) : \mathbf{Fam}(\mathbf{Set}), c : \mathsf{Arg}_{\mathrm{A}}(A, B) \to A) \to \mathbf{Fam}(\mathbf{Set})$$

by defining $\mathsf{Arg}(A, B, c) = (\mathsf{Arg}_{\mathrm{A}}(A, B), \mathsf{Arg}_{\mathrm{B}}(A, B, c))$.

Induction-induction as initial dialgebras

- What kind of structure is $((A, B) : \mathbf{Fam}(\mathbf{Set}), c : \mathsf{Arg}_{\mathrm{A}}(A, B) \to A)$?

Induction-induction as initial dialgebras

- What kind of structure is $((A, B) : \mathbf{Fam}(\mathbf{Set}), c : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A)$?
- Hagino introduced dialgebras in his PhD thesis:

Definition

Let $F, G : \mathbb{C} \to \mathbb{D}$ be functors. An (F, G)-dialgebra (X, f) consists of $X \in \mathbb{C}$ and $f : F(X) \to G(X)$. A morphism between dialgebras (X, f) and (Y, g) is a morphism $\alpha : X \to Y$ in $\mathbb{C}$ such that

$$\begin{tikzcd}
F(X) \arrow[r, "f"] \arrow[d, "F(\alpha)"'] & G(X) \arrow[d, "G(\alpha)"] \\
F(Y) \arrow[r, "g"'] & G(Y)
\end{tikzcd}$$

Induction-induction as initial dialgebras

Definition

Let $F, G : \mathbb{C} \to \mathbb{D}$ be functors. An (F, G)-dialgebra (X, f) consists of $X \in \mathbb{C}$ and $f : F(X) \to G(X)$. A morphism between dialgebras (X, f) and (Y, g) is a morphism $\alpha : X \to Y$ in $\mathbb{C}$ such that

$$\begin{tikzcd} F(X) \arrow[r, "f"] \arrow[d, "F(\alpha)"'] & G(X) \arrow[d, "G(\alpha)"] \\ F(Y) \arrow[r, "g"'] & G(Y) \end{tikzcd}$$

- $((A, B) : \textbf{Fam(Set)}, c : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A)$ is a $(\mathrm{Arg}_{\mathrm{A}}, U)$-dialgebra, where $U : \textbf{Fam(Set)} \to \textbf{Set}$ is the forgetful functor! [$U(A, B) = A$]

$$\mathbb{C} = \textbf{Fam(Set)}$$

$$\mathbb{D} = \textbf{Set}$$

$$F = \mathrm{Arg}_{\mathrm{A}} : \textbf{Fam(Set)} \to \textbf{Set}$$

$$G = U : \textbf{Fam(Set)} \to \textbf{Set} \qquad \text{(forgetful)}$$

Induction-induction as initial dialgebras (cont.)

- Thus, our ind.-ind. definitions should be represented by functors

$$\begin{aligned} \mathrm{Arg}_{\mathrm{A}} &: \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Set} \\ \mathrm{Arg} &: \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set}) \end{aligned}$$

  such that $U \circ \mathrm{Arg} = \mathrm{Arg}_{\mathrm{A}} \circ V$.

- Here, $V : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set})$ is the forgetful functor sending (A, f) to A.

- the condition just says that "the first component of Arg is $\mathrm{Arg}_{\mathrm{A}}$."

- We will often write $\mathrm{Arg} = (\mathrm{Arg}_{\mathrm{A}}, \mathrm{Arg}_{\mathrm{B}})$.

So what category do the ind.-ind. definitions live in?

- Given $\mathsf{Arg} = (\mathsf{Arg}_{\mathrm{A}}, \mathsf{Arg}_{\mathrm{B}}) : \mathsf{Dialg}(\mathsf{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set})$, one might think that the "algebras" we are looking for are in $\mathsf{Dialg}(\mathsf{Arg}, V)$.

So what category do the ind.-ind. definitions live in?

- Given $\mathrm{Arg} = (\mathrm{Arg}_{\mathrm{A}}, \mathrm{Arg}_{\mathrm{B}}) : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set})$, one might think that the "algebras" we are looking for are in $\mathrm{Dialg}(\mathrm{Arg}, V)$.

- Dialgebras with $\mathbb{C} = \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$, $\mathbb{D} = \mathbf{Fam}(\mathbf{Set})$,

$$F = \mathrm{Arg} : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set})$$
$$G = V : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set}) \qquad \text{(forgetful)}$$

  so $\mathrm{Dialg}(\mathrm{Arg}, V)$ has objects $(A, B, c, (d_0, d_1))$, where

  - $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$,
  - $c : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A$,
  - $(d_0, d_1) : \mathrm{Arg}(A, B, c) \to (A, B)$.

So what category do the ind.-ind. definitions live in?

- Given $\mathrm{Arg} = (\mathrm{Arg}_{\mathrm{A}}, \mathrm{Arg}_{\mathrm{B}}) : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set})$, one might think that the "algebras" we are looking for are in $\mathrm{Dialg}(\mathrm{Arg}, V)$.

- Dialgebras with $\mathbb{C} = \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$, $\mathbb{D} = \mathbf{Fam}(\mathbf{Set})$,

$$\begin{aligned} F &= \mathrm{Arg} : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set}) \\ G &= V : \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \to \mathbf{Fam}(\mathbf{Set}) \qquad \text{(forgetful)} \end{aligned}$$

  so $\mathrm{Dialg}(\mathrm{Arg}, V)$ has objects $(A, B, c, (d_0, d_1))$, where
  - $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$,
  - $c : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A$,
  - $(d_0, d_1) : \mathrm{Arg}(A, B, c) \to (A, B)$.

- The function $d_0 : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A$ looks like the constructor for A that we want, but

$$d_1 : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(d_0(x))$$

  does not look quite right – we need c and d_0 to be the same!

Making c and d_0 the same

$$d_1 : (x : \mathsf{Arg}_{\mathrm{A}}(A, B)) \to \mathsf{Arg}_{\mathrm{B}}(A, B, c, x) \to B(d_0(x))$$

- Use an equaliser in **CAT**.

Making c and d_0 the same

$$d_1 : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(d_0(x))$$

- Use an equaliser in **CAT**.

- Let $W : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ be the forgetful functor $[W(A, B, c, d) = (A, B, c)]$.

Making c and d_0 the same

$$d_1 : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(d_0(x))$$

- Use an equaliser in **CAT**.

- Let $W : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ be the forgetful functor $[W(A, B, c, d) = (A, B, c)]$.

- Define $(V, U) : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ by $(V, U)(A, B, c, (d_0, d_1)) := (V(A, B, c), U(d_0, d_1)) = (A, B, d_0)$.

Making c and d_0 the same

$$d_1 : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(d_0(x))$$

- Use an equaliser in **CAT**.

- Let $W : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ be the forgetful functor $[W(A, B, c, d) = (A, B, c)]$.

- Define $(V, U) : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ by $(V, U)(A, B, c, (d_0, d_1)) := (V(A, B, c), U(d_0, d_1)) = (A, B, d_0)$.

- Note:
$$\begin{aligned} U(d_0, d_1) : U(\mathrm{Arg}(A, B, c)) &\to U(V(A, B, c)) \\ = \mathrm{Arg}_{\mathrm{A}}(V(A, B, c)) &\to U(V(A, B, c)). \end{aligned}$$

Making c and d_0 the same

$$d_1 : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(d_0(x))$$

- Use an equaliser in **CAT**.

- Let $W : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ be the forgetful functor $[W(A, B, c, d) = (A, B, c)]$.

- Define $(V, U) : \mathrm{Dialg}(\mathrm{Arg}, V) \to \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U)$ by $(V, U)(A, B, c, (d_0, d_1)) \coloneqq (V(A, B, c), U(d_0, d_1)) = (A, B, d_0)$.

- Note:
$$\begin{aligned} U(d_0, d_1) : U(\mathrm{Arg}(A, B, c)) &\to U(V(A, B, c)) \\ = \mathrm{Arg}_{\mathrm{A}}(V(A, B, c)) &\to U(V(A, B, c)). \end{aligned}$$

- Take equaliser of W and (V, U), let's call it $\mathbb{E}_{(\mathrm{Arg}_{\mathrm{A}}, \mathrm{Arg}_{\mathrm{B}})}$ [subcategory with objects $(A, B, c, (d_0, d_1))$ such that $(A, B, c) = (A, B, d_0)$].

Summary

$$\begin{tikzcd}
\mathbf{Set} & \mathbf{Fam}(\mathbf{Set}) \arrow[l, bend right, "\mathrm{Arg}_{\mathrm{A}}"'] \arrow[l, bend left, "U"] & \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \arrow[l, bend right, "\mathrm{Arg}"'] \arrow[l, bend left, "V"] & \mathrm{Dialg}(\mathrm{Arg}, V) \arrow[l, bend right, "(V{,}U)"'] \arrow[l, bend left, "W"] & \mathbb{E}_{\mathrm{Arg}} \arrow[l, hook']
\end{tikzcd}$$

Warning: the diagram is not commuting!

- The category $\mathbb{E}_{(\mathrm{Arg}_{\mathrm{A}}, \mathrm{Arg}_{\mathrm{B}})}$ has objects (A, B, c, d), where
  - $A : \mathrm{Set}$,
  - $B : A \to \mathrm{Set}$,
  - $c : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A$,
  - $d : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(c(x))$.

- Morphisms are **Fam**(**Set**)-morphisms making some diagrams commute.

Summary

$$\begin{tikzcd}
\mathbf{Set} & \mathbf{Fam}(\mathbf{Set}) \arrow[l, bend right, "\mathrm{Arg}_{\mathrm{A}}"'] \arrow[l, bend left, "U"] & \mathrm{Dialg}(\mathrm{Arg}_{\mathrm{A}}, U) \arrow[l, bend right, "\mathrm{Arg}"'] \arrow[l, bend left, "V"] & \mathrm{Dialg}(\mathrm{Arg}, V) \arrow[l, bend right, "(V{,}U)"'] \arrow[l, bend left, "W"] & \mathbb{E}_{\mathrm{Arg}} \arrow[l, hook']
\end{tikzcd}$$

Warning: the diagram is not commuting!

- The category $\mathbb{E}_{(\mathrm{Arg}_{\mathrm{A}},\mathrm{Arg}_{\mathrm{B}})}$ has objects (A, B, c, d), where
  - $A : \mathrm{Set}$,
  - $B : A \to \mathrm{Set}$,
  - $c : \mathrm{Arg}_{\mathrm{A}}(A, B) \to A$,
  - $d : (x : \mathrm{Arg}_{\mathrm{A}}(A, B)) \to \mathrm{Arg}_{\mathrm{B}}(A, B, c, x) \to B(c(x))$.

- Morphisms are **Fam**(**Set**)-morphisms making some diagrams commute.

- Intended interpretation initial object in $\mathbb{E}_{(\mathrm{Arg}_{\mathrm{A}},\mathrm{Arg}_{\mathrm{B}})}$.

Initiality and elimination rules

An example of iteration from initiality

- Back to the example.
- Suppose that we want a concatenation of contexts

$$+\!\!+ : \mathsf{Context} \to \mathsf{Context} \to \mathsf{Context}$$

- For example for more general formation rules such as

$$\frac{\sigma : \mathsf{Type}(\Gamma) \qquad \tau : \mathsf{Type}(\Delta)}{\sigma \times \tau : \mathsf{Type}(\Gamma +\!\!+ \Delta)}$$

Context concatenation

- Should satisfy equations

$$\begin{array}{cccll} \Delta & +\!\!\!+ & \varepsilon & = & \Delta \\ \Delta & +\!\!\!+ & (\Gamma \triangleright \sigma) & = & (\Delta +\!\!\!+ \Gamma) \triangleright (\mathsf{wk}_{\Gamma}(\sigma, \Delta)) \quad , \end{array}$$

Context concatenation

- Should satisfy equations

$$\begin{array}{ccccc} \Delta & ++ & \varepsilon & = & \Delta \\ \Delta & ++ & (\Gamma \triangleright \sigma) & = & (\Delta ++ \Gamma) \triangleright (\mathsf{wk}_\Gamma(\sigma, \Delta)) \end{array} \quad ,$$

- $\mathsf{wk} : (\Gamma : \mathsf{Context}) \to (\sigma : \mathsf{Type}(\Gamma)) \to (\Delta : \mathsf{Context}) \to \mathsf{Type}(\Delta ++ \Gamma)$ is a weakening operation.
    - That is, if $\sigma : \mathsf{Type}(\Gamma)$, then $\mathsf{wk}_\Gamma(\sigma, \Delta) : \mathsf{Type}(\Delta ++ \Gamma)$.

Context concatenation

- Should satisfy equations
$$\begin{array}{cccll} \Delta & +\!\!+ & \varepsilon & = & \Delta \\ \Delta & +\!\!+ & (\Gamma \triangleright \sigma) & = & (\Delta +\!\!+ \Gamma) \triangleright (\mathsf{wk}_\Gamma(\sigma, \Delta)) \quad , \end{array}$$

- $\mathsf{wk} : (\Gamma : \mathsf{Context}) \to (\sigma : \mathsf{Type}(\Gamma)) \to (\Delta : \mathsf{Context}) \to \mathsf{Type}(\Delta +\!\!+ \Gamma)$ is a weakening operation.
    - That is, if $\sigma : \mathsf{Type}(\Gamma)$, then $\mathsf{wk}_\Gamma(\sigma, \Delta) : \mathsf{Type}(\Delta +\!\!+ \Gamma)$.

- Should satisfy own equations
$$\begin{aligned} \mathsf{wk}_\Gamma(\iota_\Gamma, \Delta) &= \iota_{\Delta +\!\!+ \Gamma} \\ \mathsf{wk}_\Gamma(\Pi_\Gamma(\sigma, \tau), \Delta) &= \Pi_{\Delta +\!\!+ \Gamma}(\mathsf{wk}_\Gamma(\sigma, \Delta), \mathsf{wk}_{\Gamma \triangleright \sigma}(\tau, \Delta)) \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathsf{Arg}_{\mathsf{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathsf{Arg}_{\mathsf{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathsf{inr}(c(x), \sigma)))) \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathsf{Arg}_{\mathsf{Context}}(A, B) &= \mathbf{1} + \Sigma\, \Gamma : A.\, B(\Gamma) \\ \mathsf{Arg}_{\mathsf{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\, \sigma : B(c(x)).\, \tau : B(c(\mathsf{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathsf{Context} \to \mathsf{Context}$ and $B(f) = (\Delta : \mathsf{Context}) \to \mathsf{Type}(f(\Delta))$, an $(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathsf{Arg}_{\mathsf{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathsf{Arg}_{\mathsf{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathsf{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathsf{Context} \to \mathsf{Context}$ and $B(f) = (\Delta : \mathsf{Context}) \to \mathsf{Type}(f(\Delta))$, an $(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned} &\mathsf{in}_A : \mathsf{Arg}_{\mathsf{Context}}(A, B) \to A \\ &\mathsf{in}_A(\mathsf{inl}(\star)) \quad = \quad \{? : \mathsf{Context} \to \mathsf{Context}\} \\ &\mathsf{in}_A(\mathsf{inr}(\langle f, g \rangle)) \quad = \quad \{? : \mathsf{Context} \to \mathsf{Context}\} \quad , \end{aligned}$$

$$\begin{aligned} &\mathsf{in}_B : (x : \mathsf{Arg}_{\mathsf{Context}}(A, B)) \to \mathsf{Arg}_{\mathsf{Type}}(A, B, \mathsf{in}_A, x) \to B(\mathsf{in}_A(x)) \\ &\mathsf{in}_B(\Delta, \mathsf{inl}(\star)) \quad = \quad \{?\} \\ &\mathsf{in}_B(\Delta, \mathsf{inr}(\langle g, h \rangle)) \quad = \quad \{?\} \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \end{aligned} \quad .$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned} &\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A \\ &\mathrm{in}_A(\mathrm{inl}(\star)) \quad = \quad \lambda\Delta.\ \{? : \mathrm{Context}\} \\ &\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) \quad = \quad \{? : \mathrm{Context} \to \mathrm{Context}\} \quad , \end{aligned}$$

$$\begin{aligned} &\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) \\ &\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) \quad = \quad \{?\} \\ &\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) \quad = \quad \{?\} \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned} &\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A \\ &\mathrm{in}_A(\mathrm{inl}(\star)) && = && \lambda\Delta.\, \Delta \\ &\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) && = && \{? : \mathrm{Context} \to \mathrm{Context}\} \quad , \end{aligned}$$

$$\begin{aligned} &\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) \\ &\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) && = && \{?\} \\ &\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) && = && \{?\} \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathsf{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathsf{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathsf{Context} \to \mathsf{Context}$ and $B(f) = (\Delta : \mathsf{Context}) \to \mathsf{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathsf{Context}}, \mathrm{Arg}_{\mathsf{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned} &\mathrm{in}_A : \mathrm{Arg}_{\mathsf{Context}}(A, B) \to A \\ &\mathrm{in}_A(\mathrm{inl}(\star)) &&= \lambda\Delta.\, \Delta \\ &\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) &&= \lambda\Delta.\, \{? : \mathsf{Context}\} \quad , \end{aligned}$$

$$\begin{aligned} &\mathrm{in}_B : (x : \mathrm{Arg}_{\mathsf{Context}}(A, B)) \to \mathrm{Arg}_{\mathsf{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) \\ &\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) &&= \{?\} \\ &\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) &&= \{?\} \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\, \Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\, \sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{array}{lcl} \mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A & & \\ \mathrm{in}_A(\mathrm{inl}(\star)) & = & \lambda\Delta.\, \Delta \\ \mathrm{in}_A(\mathrm{inr}(\langle f, g \rangle)) & = & \lambda\Delta.\, (\ \{?0 : \mathrm{Context}\} \triangleright \{?1 : \mathrm{Type}(?0)\}\ ) \quad , \end{array}$$

$$\begin{array}{lcl} \mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) & & \\ \mathrm{in}_B(\Delta, \mathrm{inl}(\star)) & = & \{?\} \\ \mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h \rangle)) & = & \{?\} \quad . \end{array}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned}\mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma)\\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma))))\ .\end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned}&\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A\\ &\mathrm{in}_A(\mathrm{inl}(\star)) &&= \quad \lambda\Delta.\,\Delta\\ &\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) &&= \quad \lambda\Delta.(\, f(\ \{?0 : \mathrm{Context}\}\ ) \triangleright\ \{?1 : \mathrm{Type}(f(?0))\}\end{aligned}$$

$$\begin{aligned}&\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x))\\ &\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) &&= \quad \{?\}\\ &\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) &&= \quad \{?\}\quad .\end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{array}{lcl} \multicolumn{3}{l}{\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A} \\ \mathrm{in}_A(\mathrm{inl}(\star)) & = & \lambda\Delta.\, \Delta \\ \mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) & = & \lambda\Delta.\, (f(\Delta) \triangleright \{?1 : \mathrm{Type}(f(\Delta))\}) \quad , \end{array}$$

$$\begin{array}{lcl} \multicolumn{3}{l}{\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x))} \\ \mathrm{in}_B(\Delta, \mathrm{inl}(\star)) & = & \{?\} \\ \mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) & = & \{?\} \quad . \end{array}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \ . \end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned} &\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A \\ &\mathrm{in}_A(\mathrm{inl}(\star)) &&= \quad \lambda\Delta.\,\Delta \\ &\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) &&= \quad \lambda\Delta.\,(f(\Delta) \triangleright g(\ \{?1 : \mathrm{Context}\}\ )) \quad , \end{aligned}$$

$$\begin{aligned} &\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) \\ &\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) &&= \quad \{?\} \\ &\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) &&= \quad \{?\} \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned} \mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\, \Gamma : A.\, B(\Gamma) \\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\, \sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \quad . \end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned} &\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A \\ &\mathrm{in}_A(\mathrm{inl}(\star)) \quad = \quad \lambda \Delta.\, \Delta \\ &\mathrm{in}_A(\mathrm{inr}(\langle f, g \rangle)) \quad = \quad \lambda \Delta.\, (f(\Delta) \triangleright g(\Delta)) \quad , \end{aligned}$$

$$\begin{aligned} &\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) \\ &\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) \quad = \quad \{?\} \\ &\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h \rangle)) \quad = \quad \{?\} \quad . \end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned}
\mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma) \\
\mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma)))) \quad .
\end{aligned}$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{aligned}
&\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A \\
&\mathrm{in}_A(\mathrm{inl}(\star)) &&= \quad \lambda\Delta.\, \Delta \\
&\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) &&= \quad \lambda\Delta.\, (f(\Delta) \triangleright g(\Delta)) \quad ,
\end{aligned}$$

$$\begin{aligned}
&\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x)) \\
&\mathrm{in}_B(\Delta, \mathrm{inl}(\star)) &&= \quad \lambda\Gamma.\, \iota_{\mathrm{in}_A(\Delta)(\Gamma)} \\
&\mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) &&= \quad \{?\} \quad .
\end{aligned}$$

Context concatenation (cont.)

- Recall:

$$\begin{aligned}\mathrm{Arg}_{\mathrm{Context}}(A, B) &= \mathbf{1} + \Sigma\,\Gamma : A.\, B(\Gamma)\\ \mathrm{Arg}_{\mathrm{Type}}(A, B, c, x) &= \mathbf{1} + (\Sigma\,\sigma : B(c(x)).\, \tau : B(c(\mathrm{inr}(c(x), \sigma))))\end{aligned} \quad .$$

- Want to give (A, B), where $A = \mathrm{Context} \to \mathrm{Context}$ and $B(f) = (\Delta : \mathrm{Context}) \to \mathrm{Type}(f(\Delta))$, an $(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})$ structure such that initiality gives us $+\!\!+$ and wk.

$$\begin{array}{lcl}\multicolumn{3}{l}{\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A}\\ \mathrm{in}_A(\mathrm{inl}(\star)) &=& \lambda\Delta.\, \Delta\\ \mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) &=& \lambda\Delta.\, (f(\Delta) \triangleright g(\Delta)) \quad ,\end{array}$$

$$\begin{array}{lcl}\multicolumn{3}{l}{\mathrm{in}_B : (x : \mathrm{Arg}_{\mathrm{Context}}(A, B)) \to \mathrm{Arg}_{\mathrm{Type}}(A, B, \mathrm{in}_A, x) \to B(\mathrm{in}_A(x))}\\ \mathrm{in}_B(\Delta, \mathrm{inl}(\star)) &=& \lambda\Gamma.\, \iota_{\mathrm{in}_A(\Delta)(\Gamma)}\\ \mathrm{in}_B(\Delta, \mathrm{inr}(\langle g, h\rangle)) &=& \lambda\Gamma.\, \Pi_{\mathrm{in}_A(\Delta)(\Gamma)}(g(\Gamma), h(\Gamma)) \quad .\end{array}$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$+\!\!+(\varepsilon) =$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$+\!\!+(\varepsilon) = \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star)))$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathrm{wk}) : (\mathrm{Context}, \mathrm{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})(\mathrm{Context}, \mathrm{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathrm{Arg}_{\mathrm{Context}},\mathrm{Arg}_{\mathrm{Type}})(+\!\!+,\mathrm{wk})}"'] & (\mathrm{Context}, \mathrm{Type}) \arrow[d, "{(+\!\!+,\mathrm{wk})}"] \\
(\mathrm{Arg}_{\mathrm{Conte}} \arrow[r, "{(\mathrm{in}_A,\mathrm{in}_B)}"'] & (A, B)
\end{tikzcd}$$

$$\boxed{\mathrm{Arg}_{\mathrm{Context}}(f, g) = \mathrm{id} + \Sigma(f, g)}$$

- We better check it satisfies the specification:

$$+\!\!+(\varepsilon) = \mathrm{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathrm{wk})(\mathrm{inl}(\star)))$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})(\mathrm{Context}, \mathrm{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathrm{Arg}_{\mathrm{Context}},\mathrm{Arg}_{\mathrm{Type}})(+\!\!+,\mathrm{wk})}"'] & (\mathrm{Context}, \mathrm{Type}) \arrow[d, "{(+\!\!+,\mathrm{wk})}"] \\
(\mathrm{Arg}_{\mathrm{Conte} \arrow[r, "{(\mathrm{in}_A,\mathrm{in}_B)}"'] & (A, B)
\end{tikzcd}$$

$$\boxed{\mathrm{Arg}_{\mathrm{Context}}(f, g) = \mathrm{id} + \Sigma(f, g)}$$

- We better check it satisfies the specification:

$$+\!\!+(\varepsilon) = \mathrm{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathrm{wk})(\mathrm{inl}(\star))) = \mathrm{in}_A(\mathrm{inl}(\star))$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathrm{Arg}_{\mathsf{Context}}, \mathrm{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathrm{Arg}_{\mathsf{Context}},\mathrm{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathrm{Arg}_{\mathsf{Conte} & B)
\end{tikzcd}$$

$$\begin{aligned}
&\mathsf{in}_A : \mathrm{Arg}_{\mathsf{Context}}(A, B) \to A \\
&\mathsf{in}_A(\mathsf{inl}(\star)) && = && \lambda\Delta.\, \Delta \\
&\mathsf{in}_A(\mathsf{inr}(\langle f, g\rangle)) && = && \lambda\Delta.\, (f(\Delta) \triangleright g(\Delta))
\end{aligned}$$

- We better ch

$$+\!\!+(\varepsilon) = \mathsf{in}_A(\mathrm{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star))$$

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Conte} & B)
\end{tikzcd}$$

$$\begin{aligned}
&\mathsf{in}_A : \mathsf{Arg}_{\mathsf{Context}}(A, B) \to A \\
&\mathsf{in}_A(\mathsf{inl}(\star)) \quad = \quad \lambda\Delta.\,\Delta \\
&\mathsf{in}_A(\mathsf{inr}(\langle f, g \rangle)) \quad = \quad \lambda\Delta.\,(f(\Delta) \triangleright g(\Delta))
\end{aligned}$$

- We better ch

$$+\!\!+(\varepsilon) = \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\,\Delta$$

Context concatenation (cont.)

- Initiality gives a morphism $(\mathbin{+\!\!+}, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(\mathbin{+\!\!+},\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(\mathbin{+\!\!+},\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$\mathbin{+\!\!+}(\varepsilon) = \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(\mathbin{+\!\!+}, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\,\Delta$$

- Thus $\Delta \mathbin{+\!\!+} \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$+\!\!+(\varepsilon) = \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\, \Delta$$

$$+\!\!+\,(\Gamma \triangleright \sigma) =$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(\mathbin{+\!\!+}, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(\mathbin{+\!\!+},\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(\mathbin{+\!\!+},\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$\mathbin{+\!\!+}(\varepsilon) = \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(\mathbin{+\!\!+}, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\, \Delta$$

$$\mathbin{+\!\!+}\ (\Gamma \triangleright \sigma) = \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(\mathbin{+\!\!+}, \mathsf{wk})(\mathsf{inr}(\langle \Gamma, \sigma \rangle)))$$

- Thus $\Delta \mathbin{+\!\!+} \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathrm{wk}) : (\mathrm{Context}, \mathrm{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})(\mathrm{Context}, \mathrm{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathrm{Arg}_{\mathrm{Context}},\mathrm{Arg}_{\mathrm{Type}})(+\!\!+,\mathrm{wk})}"'] & (\mathrm{Context}, \mathrm{Type}) \arrow[d, "{(+\!\!+,\mathrm{wk})}"] \\
(\mathrm{Arg}_{\mathrm{Conte}} \arrow[r, "{(\mathrm{in}_A,\mathrm{in}_B)}"'] & (A, B)
\end{tikzcd}$$

$$\boxed{\mathrm{Arg}_{\mathrm{Context}}(f, g) = \mathrm{id} + \Sigma(f, g)}$$

- We better check it satisfies the specification:

$$+\!\!+(\varepsilon) = \mathrm{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathrm{wk})(\mathrm{inl}(\star))) = \mathrm{in}_A(\mathrm{inl}(\star)) = \lambda\Delta.\, \Delta$$

$$+\!\!+\,(\Gamma \triangleright \sigma) = \mathrm{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathrm{wk})(\mathrm{inr}(\langle \Gamma, \sigma \rangle)))$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathrm{Arg}_{\mathrm{Context}},\mathrm{Arg}_{\mathrm{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathrm{Arg}_{\mathrm{Conte}} \arrow[r, "{(\mathrm{in}_A,\mathrm{in}_B)}"'] & (A, B)
\end{tikzcd}$$

$$\boxed{\mathrm{Arg}_{\mathrm{Context}}(f, g) = \mathsf{id} + \Sigma(f, g)}$$

- We better check it satisfies the specification:

$$\begin{aligned}
+\!\!+(\varepsilon) &= \mathsf{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\, \Delta \\
+\!\!+\,(\Gamma \triangleright \sigma) &= \mathsf{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inr}(\langle \Gamma, \sigma \rangle))) \\
&= \mathsf{in}_A(\mathsf{inr}(\langle +\!\!+(\Gamma), \mathsf{wk}(\Gamma, \sigma) \rangle))
\end{aligned}$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathrm{wk}) : (\mathrm{Context}, \mathrm{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathrm{Arg}_{\mathrm{Context}}, \mathrm{Arg}_{\mathrm{Type}})(\mathrm{Context}, \mathrm{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathrm{Arg}_{\mathrm{Context}},\mathrm{Arg}_{\mathrm{Type}})(+\!\!+,\mathrm{wk})}"'] & (\mathrm{Context}, \mathrm{Type}) \arrow[d, "{(+\!\!+,\mathrm{wk})}"] \\
(\mathrm{Arg}_{\mathrm{Conte}} & B)
\end{tikzcd}$$

$$\begin{array}{lcl}
\mathrm{in}_A : \mathrm{Arg}_{\mathrm{Context}}(A, B) \to A & & \\
\mathrm{in}_A(\mathrm{inl}(\star)) & = & \lambda\Delta.\, \Delta \\
\mathrm{in}_A(\mathrm{inr}(\langle f, g\rangle)) & = & \lambda\Delta.\, (f(\Delta) \triangleright g(\Delta))
\end{array}$$

- We better ch

$$+\!\!+(\varepsilon) = \mathrm{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathrm{wk})(\mathrm{inl}(\star))) = \mathrm{in}_A(\mathrm{inl}(\star)) = \lambda\Delta.\, \Delta$$

$$\begin{aligned}
+\!\!+\,(\Gamma \triangleright \sigma) &= \mathrm{in}_A(\mathrm{Arg}_{\mathrm{Context}}(+\!\!+, \mathrm{wk})(\mathrm{inr}(\langle \Gamma, \sigma\rangle))) \\
&= \mathrm{in}_A(\mathrm{inr}(\langle +\!\!+(\Gamma), \mathrm{wk}(\Gamma, \sigma)\rangle))
\end{aligned}$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Conte} & B)
\end{tikzcd}$$

$$\begin{aligned}
&\mathsf{in}_A : \mathsf{Arg}_{\mathsf{Context}}(A, B) \to A \\
&\mathsf{in}_A(\mathsf{inl}(\star)) \quad = \quad \lambda\Delta.\, \Delta \\
&\mathsf{in}_A(\mathsf{inr}(\langle f, g\rangle)) \quad = \quad \lambda\Delta.\, (f(\Delta) \triangleright g(\Delta))
\end{aligned}$$

- We better ch

$$\begin{aligned}
+\!\!+(\varepsilon) &= \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\, \Delta \\
+\!\!+\ (\Gamma \triangleright \sigma) &= \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inr}(\langle \Gamma, \sigma\rangle))) \\
&= \mathsf{in}_A(\mathsf{inr}(\langle +\!\!+(\Gamma), \mathsf{wk}(\Gamma, \sigma)\rangle)) \\
&= \lambda\Delta.\, +\!\!+\ (\Gamma, \Delta) \triangleright \mathsf{wk}(\Gamma, \sigma, \Delta)\ .
\end{aligned}$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "({[\varepsilon,\triangleright]},{[\iota,\Pi]})"] \arrow[d, "(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "(+\!\!+,\mathsf{wk})"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "(\mathsf{in}_A,\mathsf{in}_B)"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$\begin{aligned}
+\!\!+(\varepsilon) &= \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\,\Delta \\
+\!\!+\,(\Gamma \triangleright \sigma) &= \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inr}(\langle \Gamma, \sigma \rangle))) \\
&= \mathsf{in}_A(\mathsf{inr}(\langle +\!\!+(\Gamma), \mathsf{wk}(\Gamma, \sigma) \rangle)) \\
&= \lambda\Delta.\, +\!\!+\,(\Gamma, \Delta) \triangleright \mathsf{wk}(\Gamma, \sigma, \Delta) \quad .
\end{aligned}$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$ and $\Delta +\!\!+ (\Gamma \triangleright \sigma) = (\Delta +\!\!+ \Gamma) \triangleright \mathsf{wk}_\Gamma(\sigma, \Delta)$.

Context concatenation (cont.)

- Initiality gives a morphism $(+\!\!+, \mathsf{wk}) : (\mathsf{Context}, \mathsf{Type}) \to (A, B)$ s. t.

$$\begin{tikzcd}
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(\mathsf{Context}, \mathsf{Type}, [\varepsilon, \triangleright]) \arrow[r, "{([\varepsilon,\triangleright],[\iota,\Pi])}"] \arrow[d, "{(\mathsf{Arg}_{\mathsf{Context}},\mathsf{Arg}_{\mathsf{Type}})(+\!\!+,\mathsf{wk})}"'] & (\mathsf{Context}, \mathsf{Type}) \arrow[d, "{(+\!\!+,\mathsf{wk})}"] \\
(\mathsf{Arg}_{\mathsf{Context}}, \mathsf{Arg}_{\mathsf{Type}})(A, B, \mathsf{in}_A) \arrow[r, "{(\mathsf{in}_A,\mathsf{in}_B)}"'] & (A, B)
\end{tikzcd}$$

- We better check it satisfies the specification:

$$\begin{aligned}
+\!\!+(\varepsilon) &= \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inl}(\star))) = \mathsf{in}_A(\mathsf{inl}(\star)) = \lambda\Delta.\, \Delta \\
+\!\!+\,(\Gamma \triangleright \sigma) &= \mathsf{in}_A(\mathsf{Arg}_{\mathsf{Context}}(+\!\!+, \mathsf{wk})(\mathsf{inr}(\langle \Gamma, \sigma \rangle))) \\
&= \mathsf{in}_A(\mathsf{inr}(\langle +\!\!+(\Gamma), \mathsf{wk}(\Gamma, \sigma) \rangle)) \\
&= \lambda\Delta.\, +\!\!+\,(\Gamma, \Delta) \triangleright \mathsf{wk}(\Gamma, \sigma, \Delta) \quad .
\end{aligned}$$

- Thus $\Delta +\!\!+ \varepsilon = \Delta$ and $\Delta +\!\!+ (\Gamma \triangleright \sigma) = (\Delta +\!\!+ \Gamma) \triangleright \mathsf{wk}_\Gamma(\sigma, \Delta)$.

- The equations for wk hold in the same way.

What about dependent functions?

- Traditional presentations of type theory include elimination rules (eliminator terms) instead of defining functions using initiality.
- Get dependent functions

$$\begin{aligned}&\mathsf{elim}_{\mathsf{Arg}_{\mathrm{A}}}(\ldots) : (x : A) \to P(x)\\&\mathsf{elim}_{\mathsf{Arg}_{\mathrm{B}}}(\ldots) : (x : A) \to (y : B(x)) \to Q(x, y, \mathsf{elim}_{\mathsf{Arg}_{\mathrm{A}}}(\ldots, x))\end{aligned}$$

  by defining them for elements of the form $c(x)$, $d(x, y)$ with access to inductive hypothesis / structurally recursive calls.

What about dependent functions?

- Traditional presentations of type theory include elimination rules (eliminator terms) instead of defining functions using initiality.

- Get dependent functions

$$\begin{aligned}&\mathsf{elim}_{\mathsf{Arg}_{\mathrm{A}}}(\ldots) : (x : A) \to P(x)\\ &\mathsf{elim}_{\mathsf{Arg}_{\mathrm{B}}}(\ldots) : (x : A) \to (y : B(x)) \to Q(x, y, \mathsf{elim}_{\mathsf{Arg}_{\mathrm{A}}}(\ldots, x))\end{aligned}$$

  by defining them for elements of the form $c(x)$, $d(x, y)$ with access to inductive hypothesis / structurally recursive calls.

- In detail:

$$\frac{\begin{array}{c}P : A \to \mathsf{Set}\\ Q : (x : A) \to B(x) \to P(x) \to \mathsf{Set}\\ \mathsf{step}_c : (x : \mathsf{Arg}_{\mathrm{A}}(A, B)) \to \Box_{\mathsf{Arg}_{\mathrm{A}}}(P, Q, x) \to P(c(x))\\ \mathsf{step}_d : (x : \mathsf{Arg}_{\mathrm{A}}(A, B)) \to (y : \mathsf{Arg}_{\mathrm{B}}(A, B, c, x)) \to (\widetilde{x} : \Box_{\mathsf{Arg}_{\mathrm{A}}}(P, Q, x))\\ \qquad \to \Box_{\mathsf{Arg}_{\mathrm{B}}}(P, Q, c, \mathsf{step}_c, x, y, \widetilde{x}) \to Q(c(x), d(x, y), \mathsf{step}_c(x, \widetilde{x}))\end{array}}{\begin{array}{c}\mathsf{elim}_{\mathsf{Arg}_{\mathrm{A}}}(P, Q, \mathsf{step}_c, \mathsf{step}_d) : (x : A) \to P(x)\\ \mathsf{elim}_{\mathsf{Arg}_{\mathrm{B}}}(P, Q, \mathsf{step}_c, \mathsf{step}_d) : (x : A) \to (y : B(x)) \to Q(x, y, \mathsf{elim}_{\mathsf{Arg}_{\mathrm{A}}}(P, Q, \mathsf{step}_c, \mathsf{step}_d, x))\end{array}}$$

Elimination rules vs. initiality

- One could think that eliminators are strictly stronger than initiality, since they allow one to define dependent functions (and do proof by induction).

Elimination rules vs. initiality

- One could think that eliminators are strictly stronger than initiality, since they allow one to define dependent functions (and do proof by induction).

- However, this is not the case!

Proposition

An initial object in $\mathbb{E}_{Arg}$ has an eliminator.

Elimination rules vs. initiality

- One could think that eliminators are strictly stronger than initiality, since they allow one to define dependent functions (and do proof by induction).
- However, this is not the case!

Proposition

An initial object in $\mathbb{E}_{Arg}$ has an eliminator.

- The proof is a generalisation of the proof of the analog result for initial algebras.

Initiality vs. elimination rules

- By considering constant families $P(x) = Y$, $Q(v, x, y) = Z(y)$, we get

Proposition

Every object with an eliminator is weakly initial in $\mathbb{E}_{Arg}$.

Equivalence for strictly positive functors

- For strictly positive functors (as codified in our previous axiomatisation), we can do induction on their build-up to prove the uniqueness of the initiality arrow.

Theorem

For an inductive-inductive definition given by a strictly positive functor $(\mathit{Arg}_{\mathrm{A}}, \mathit{Arg}_{\mathrm{B}})$*, the elimination rules hold if and only if* $\mathbb{E}_{(\mathit{Arg}_{\mathrm{A}}, \mathit{Arg}_{\mathrm{B}})}$ *has an initial object.*

Summary

- **Inductive-inductive definitions:** $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$ defined mutually dependent, both defined inductively.

- **Initial-algebra-like semantics**, but using **dialgebras** instead of ordinary algebras.

- Equivalence between **initiality** and **elimination rules** for strictly positive functors.

Summary

- **Inductive** ed
  mutually

- **Initial-al** of
  ordinary a

- Equivalen tly
  positive f

Thanks!