The categorical setting for the general case

Fredrik Nordvall Forsberg

November 29, 2010

Abstract

This note will try to explain the categorical setting for the general case of inductive-inductive definitions, first the Warsaw-Nottingham version, and then Anton's recent suggestion. Finally, I will show that the two versions in fact are (trivially) isomorphic.

1 Common starting point

Before we start, recall the category Fam(Set):

Definition 1. The category Fam(Set) has

- objects: pairs $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$.
- morphisms $(A, B) \Rightarrow_{\mathrm{Fam(Set)}} (A', B')$: $f : A \to A'$, $g : (a : A) \to B(a) \to B(f(a))$.

The identity on (A, B) is $(\lambda x.x, \lambda x, y.y)$ and composition is given by $(f', g') \circ (f, g) :\equiv (f' \circ f, \lambda(x : A)(y : B(x)).g'(f(x), g(x, y)))$. ■

Our inductive-inductive sets are given to us as a functor

$$F : \mathrm{Fam(Set)} \to \mathrm{Set}$$

and an "operation"

$$G : (A : \mathrm{Set})(B : A \to \mathrm{Set})(c : F(A, B) \to A) \to F(A, B) \to \mathrm{Set}$$

(we will see in what sense G has to be functorial later). After two examples, let us fix such F and G.

Example 2 (Platforms and buildings)**.** The platforms and buildings are given by the following code in Agda:

```
mutual
  data Platform : Set where
    ground : Platform
    extension : (p : Platform) -> Building p -> Platform
```

```
  data Building : Platform -> Set where
    onTop : (p : Platform) -> Building p
    hangingUnder : (p : Platform) -> (b : Building p)
                                    -> Building (extension p b)
```

which corresponds to

$$\begin{aligned} F(A,B) &= 1 + \Sigma\ A\ B \\ G(A,B,c,x) &= 1 + \Sigma a : A, b : B(a).\ x = \mathrm{inr}(\langle a, b\rangle) \end{aligned}$$

■

Example 3 (Contexts and types)**.**

```
mutual
  data Context : Set where
      ε : Context
    _::_ : (Γ : Context) -> Type Γ -> Context

  data Type : Context -> Set where
    ι  : (Γ : Context) -> Type Γ
    Π : (Γ : Context) -> (A : Type Γ) -> (B : Type (Γ :: A)) -> Type Γ
```

is represented by the functors

$$\begin{aligned} F(A,B) &= 1 + \Sigma\ A\ B \\ G(A,B,c,x) &= 1 + \Sigma a : A, b : B(a), B(c(\mathrm{inr}(\langle a, b\rangle))).\ c(x) = a. \end{aligned}$$

■

2 Bialgebra version

We will now go from these functors to the sets defined by them. The important concept is that of a bialgebra:

Definition 4. Let $S, T : \mathbb{C} \to \mathbb{D}$ be functors. The category $\mathrm{BiAlg}(S,T)$ of (S,T)-*bialgebras* has

- objects: an object $X \in |\mathbb{C}|$ together with a morphism $\alpha : SX \to TX$.
- morphisms $(X, \alpha) \Rightarrow_{\mathrm{BiAlg}(S,T)} (Y, \beta)$: morphisms $f : X \to Y$ such that the following diagram commutes:

$$\begin{tikzcd} SX \arrow[r, "\alpha"] \arrow[d, "Sf"'] & TY \arrow[d, "Tf"] \\ SX \arrow[r, "\beta"] & TY \end{tikzcd}$$

Composition and identity morphisms are inherited from $\mathbb{C}$. ∎

Remark 5. We might want to come up with a new name, as "bialgebra" seems to be already taken. Wikipedia informs me that

> "in mathematics, a bialgebra over a field K is a structure (vector space) which is both a unital associative algebra and a coalgebra over K, such that these structures are compatible."

I will still use the terminology bialgebra for now.

Note 6. (T, S)-bialgebras are a generalisation of T-algebras, as an (T, ID)-bialgebra is exactly an T-algebra (where $\mathrm{ID} : \mathbb{C} \to \mathbb{C}$ is the identity functor).

Note 7. Let $S, T : \mathbb{C} \to \mathbb{D}$. There is always a forgetful functor $V : \mathrm{BiAlg}(S, T) \to \mathbb{C}$ defined by $V(X, \alpha) = X$, $V(f, p) = f$ (where p is the proof that the diagram commutes – in a less type theoretical formalisation, there is of course no p).

The following (non-commuting!) diagram might be useful as a map to what follows:

$$\begin{tikzcd}
\mathrm{Set} & \mathrm{Fam(Set)} \arrow[l, bend right, "F"'] \arrow[l, bend left, "U"] & \mathrm{BiAlg}(F, U) \arrow[l, bend right, "\widehat{G}"'] \arrow[l, bend left, "V"] & \mathrm{BiAlg}(\widehat{G}, W) \arrow[l, bend right, "\overline{U}"'] \arrow[l, bend left, "W"] & E \arrow[l, hook']
\end{tikzcd}$$

Recall that we have a functor $F : \mathrm{Fam(Set)} \to \mathrm{Set}$. Let $U : \mathrm{Fam(Set)} \to \mathrm{Set}$ be the forgetful functor (i.e. $U(A, B) = A$, $U(f, g) = f$). Consider the category $\mathrm{BiAlg}(F, U)$. Concretely, it has as objects triples $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$, $c : F(A, B) \to A$ and morphisms $(A, B, c) \Rightarrow_{\mathrm{BiAlg}(F,U)} (A', B', c')$ are pairs $f : A \to A'$, $g : (a : A) \to B(a) \to B'(f(a))$ such that

$$\begin{tikzcd}
F(A, B) \arrow[r, "c"] \arrow[d, "F(f,g)"'] & A \arrow[d, "f"] \\
F(A', B') \arrow[r, "c'"] & A'
\end{tikzcd}$$

The idea now is to make

$$G : (A : \mathrm{Set})(B : A \to \mathrm{Set})(c : F(A, B) \to A) \to F(A, B) \to \mathrm{Set}$$

into a functor $\widehat{G} : \mathrm{BiAlg}(F, U) \to \mathrm{Fam(Set)}$ by defining

$$\widehat{G}(A, B, c) = (F(A, B), G(A, B, c))$$
$$\widehat{G}(f, g, p) = (F(f, g), G(f, g, p))$$

This requires G to be functorial in the sense that if $(f, g, p) : (A, B, c) \Rightarrow_{\mathrm{BiAlg}(F,U)} (A', B', c')$ – i.e. $f : A \to A'$, $g : (a : A) \to B(a) \to B'(f(a))$ and $p : f \circ c = c' \circ F(f, g)$ – then

$$G(f, g, p) : (x : F(A, B)) \to G(A, B, c, x) \to G(A', B', c', F(f, g)(x)).$$

Let $V : \mathrm{BiAlg}(F, U) \to \mathrm{Fam}(\mathrm{Set})$ be the forgetful functor, and consider the category $\mathrm{BiAlg}(\widehat{G}, V)$. It has objects quadruples $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$, $c : F(A, B) \to A$, $d : \widehat{G}(A, B, c) \Rightarrow_{\mathrm{Fam(Set)}} (A, B)$ and a morphism $(A, B, c, d) \Rightarrow_{\mathrm{BiAlg}(\widehat{G},V)} (A', B', c', d')$ is a morphism $(f, g, p) : (A, B, c) \Rightarrow_{\mathrm{BiAlg}(F,U)} (A', B', c')$ such that

$$\begin{tikzcd}
\widehat{G}(A, B, c) \arrow[r, "d"] \arrow[d, "\widehat{G}(f,g,p)"'] & (A, B) \arrow[d, "(f,g)"] \\
\widehat{G}((A', B', c') \arrow[r, "d'"] & (A', B')
\end{tikzcd}$$

More explicitly, $d : \widehat{G}(A, B, c) \Rightarrow_{\mathrm{Fam(Set)}} (A, B)$ means that $d = (d_0, d_1)$ where $d_0 : F(A, B) \to A$ and $d_1 : (x : F(A, B)) \to G(A, B, c) \to B(d_0(x))$. However, we would like that $d_0 = c$, so that $d_1 : (x : F(A, B)) \to G(A, B, c) \to B(c(x))$.

For this end, consider the functor $\overline{U} : \mathrm{BiAlg}(\widehat{G}, V) \to \mathrm{BiAlg}(F, U)$ defined by

$$\begin{aligned}
\overline{U}(A, B, c, (d_0, d_1)) &:= (V(A, B, c), U(d_0, d_1)) = (A, B, d_0) \\
\overline{U}(f, g) &:= V(f, g)
\end{aligned}$$

This is well-typed, since $F \circ V = U \circ \widehat{G}$ by construction

$$F(V(A, B, c)) = F(A, B) = U(F(A, B), G(A, B, c)) = U(\widehat{G}(A, B, c))$$

and hence

$$\begin{aligned}
U(d_0, d_1) &: U(\widehat{G}(A, B, c)) \to U(V(A, B, c)) \\
&: F(V(A, B, c)) \to U(V(A, B, c)),
\end{aligned}$$

i.e. $(V(A, B, c), U(d_0, d_1))$ is an object of $\mathrm{BiAlg}(F, U)$. As usual, there is also a forgetful functor $W : \mathrm{BiAlg}(\widehat{G}, V) \to \mathrm{BiAlg}(F, u)$ with $W(A, B, c, d) = (A, B, c)$. Now consider the category E which is the equalizer of W and $\overline{U}$ in Cat. E is the subcategory of $\mathrm{BiAlg}(\widehat{G}, V)$ where for objects

$$W(A, B, c, (d_0, d_1)) = \overline{U}(A, B, c, (d_0, d_1)) \Leftrightarrow (A, B, c) = (A, B, d_0) \Leftrightarrow c = d_0.$$

Remark 8. From a type theoretic perspective, this requires an equality on the large type $A \to \mathrm{Set}$, since we need $B = B : A \to \mathrm{Set}$. The conclusion we need only talks about equality on the set $F(A, B) \to A$, though.

We can spell out the category E as the category with

- Objects: $A : \mathrm{Set}$, $B : A \to \mathrm{Set}$, $c : F(A, B) \to \mathrm{Set}$, $d : (x : F(A, B)) \to G(A, B, c, x) \to B(c(x))$.
- Morphisms $(A, B, c, d) \Rightarrow (A', B', c', d')$: $(f, g) : (A, B, c) \Rightarrow_{\mathrm{BiAlg}(F,U)} (A', B', c')$ such that

$$g(c(x), d(x, y)) = d'(F(f, g)(x), G(f, g)(x, y)).$$

[This equation is only well-typed since (f, g) is an BiAlg(F, U) morphism – the LHS has type $B(f(c(x))$, the RHS $B(c'(F(f, g)(x)))$, but $f(c(x)) = c'(F(f, g)(x)$ for BiAlg(F, U) morphisms.]

The intended interpretation of the inductive-inductive definition given by F, G is the initial object in E.

3 Anton's version

In Anton's version, we approach things slightly differently. The main difference is that we make F into a functor on Fam(Set) by choosing the empty family over $F(A, B)$. This allows us to use the following structure (with $\mathbb{C}$ instantiated to Fam(Set)):

Definition 9. Let $S : \mathbb{C} \to \mathbb{C}$ and $T : S\text{-Alg} \to \mathbb{C}$ be functors. The category $\overline{\text{Alg}}(S, T)$ of (S, T)-*algebras* has

- objects: an object $X \in |\mathbb{C}|$, a morphism $f : SX \to X$ and a morphism $g : T(X, f) \to X$.
- morphisms $(X, f, g) \Rightarrow_{\overline{\text{Alg}}(S,T)} (Y, f', g')$: morphisms $h : X \to Y$ such that the following diagram commutes:

$$\begin{tikzcd}
SX \arrow[r, "f"] \arrow[d, "Sh"'] \arrow[dr, phantom, "(p)"] & X \arrow[d, "h"] & T(X, f) \arrow[l, "g"'] \arrow[d, "T(h,p)"] \\
SY \arrow[r, "f'"'] & Y & T(Y, f') \arrow[l, "g'"]
\end{tikzcd}$$

Composition and identity morphisms are inherited from $\mathbb{C}$. ■

Given a functor $F : \text{Fam(Set)} \to \text{Set}$, we construct a functor $\widehat{F} : \text{Fam(Set)} \to \text{Fam(Set)}$ by defining

$$\begin{aligned}
\widehat{F}(A, B) &= (F(A, B), \lambda x.\ \mathbf{0}) \\
\widehat{F}(f, g) &= (F(f, g), \lambda x, y.\ y)
\end{aligned}$$

Now we want to make

$$G : (A : \text{Set})(B : A \to \text{Set})(c : F(A, B) \to A) \to F(A, B) \to \text{Set}$$

into a functor $\widehat{G} : \widehat{F}\text{-Alg} \to \text{Fam(Set)}$ by defining

$$\begin{aligned}
\widehat{G}(A, B, (c_0, c_1)) &= (F(A, B), G(A, B, c_0)) \\
\widehat{G}(f, g, p) &= (F(f, g), G(f, g, p))
\end{aligned}$$

where we assume that G to be functorial in the sense that if $(f, g, p) : (A, B, c_0, c_1) \Rightarrow_{\widehat{F}\text{-Alg}} (A', B', c'_0, c'_1)$ – i.e. $f : A \to A'$, $g : (a : A) \to B(a) \to B'(f(a))$ and $p : (f, g) \circ (c_0, c_1) = (c'_0, c'_1) \circ (F(f, g), \lambda x .!_{\mathbf{0}})$ – then

$$G(f, g, p) : (x : F(A, B)) \to G(A, B, c, x) \to G(A', B', c', F(f, g)(x)).$$

That $(f, g) \circ (c_0, c_1) = (c'_0, c'_1) \circ (F(f, g), \lambda x, y.\ y)$ means that

- $f \circ c_0 = F(f, g) \circ c'_0$, and
- for all $x : F(A, B)$, $y : \mathbf{0}$, we have $g(c_0(x), c_1(x, y)) = c'_1(F(f, g)(x), y)$.

However, since $c_1 : (x : F(A, B)) \to \mathbf{0} \to B(c_0(x))$ and $c'_1 : (x' : F(A', B')) \to \mathbf{0} \to B'(c'_0(x'))$, we have $c_1(x) =!_{B(c_0(x))}$ and $c'_1(F(f, g)(x)) =!_{B'(c'_0(F(F,g)(x)))}$ for all x by extensionality. This reduces the second equation to

$$g(c_0(x), !_{B(c_0(x))}(y)) =!_{B'(c'_0(F(F,g)(x)))}(y),$$

but since $\lambda y.\ g(c_0(x), !_{B(c_0(x))}(y)) : \mathbf{0} \to B'(f(c_0(x)))$ and $f \circ c_0 = F(f, g) \circ c'_0$ by the first equation, the second equation must hold by the uniqueness of $!_{B'(c'_0(F(f,g)(x)))}$. Hence, p only needs to be a proof that $f \circ c_0 = F(f, g) \circ c'_0$.

Now consider the category $\overline{\text{Alg}}(\widehat{F}, \widehat{G})$. An object consists of

- $A : \text{Set}$, $B : A \to \text{Set}$,
- $c_0 : F(A, B) \to A$, $c_1 : (x : F(A, B) \to \mathbf{0} \to B(c_0(x))$,
- $d_0 : F(A, B) \to A$, $d_1 : (x : F(A, B)) \to G(A, B, c_0) \to B(d_0(x))$.

However, we would like $c_0 = d_0$ so that d_1 gets the right type. Consider the forgetful functor $W : \overline{\text{Alg}}(\widehat{F}, \widehat{G}) \to \widehat{F}\text{-Alg}$ defined by

$$\begin{aligned} W(A, B, c_0, c_1, d_0, d_1) &= (A, B, c_0, c_1) \\ W(f, g, p, q) &= (f, g, p) \end{aligned}$$

and the functor $\overline{U} : \overline{\text{Alg}}(\widehat{F}, \widehat{G}) \to \widehat{F}\text{-Alg}$ defined by

$$\begin{aligned} \overline{U}(A, B, c_0, c_1, d_0, d_1) &= (A, B, d_0, \lambda x.\ !_{B(d_0(x))}) \\ \overline{U}(f, g, p, q) &= (f, g, \text{fst}(q).) \end{aligned}$$

Consider the equaliser E' of W and $\overline{U}$ in Cat. It is the subcategory of $\overline{\text{Alg}}(\widehat{F}, \widehat{G})$ where $W(A, B, c, d) = \overline{U}(A, B, c, d)$, i.e. $d_0 = c_0$. (We already know that $c_1 = \lambda x.\ !_{B(d_0(x))}$ by extensionality.) Thus, E' (is isomorphic to a category which) has objects

- $A : \text{Set}$, $B : A \to \text{Set}$,
- $c_0 : F(A, B) \to A$,
- $d_1 : (x : F(A, B)) \to G(A, B, c_0, x) \to B(c_0(x))$

and a morphism $(A, B, c_0, d_1) \Rightarrow (A', B', c'_0, d'_0)$ consists of a morphism $(f, g) : (A, B) \Rightarrow_{\text{Fam(Set)}} (A', B')$, a proof $p : f \circ c_0 = c'_0 \circ F(f, g)$ and a proof $q : (f, g) \circ (c_0, d_0) = (c'_0, d'_0) \circ (F(f, g), G(f, g, p))$.

The intended interpretation of the inductive-inductive set is the initial object in this category.

Remark 10. Another option is to have $G(A, B, c) : A \to \text{Set}$ and define $\widehat{G}(A, B, c_0, c_1) = (A, G(A, B, c_0))$. We then need to ensure that $d_0 = \text{id}$ instead of $d_0 = c_0$. I cannot immediately see how to achieve this by using e.g. equalizers.

4 Relating the two approaches

Let E be the category from Section 2 and E' the category from Section 3. They both have objects quadruples

- $A : \text{Set}$, $B : A \to \text{Set}$,
- $c : F(A, B) \to A$,
- $d : (x : F(A, B)) \to G(A, B, c, x) \to B(c(x))$

[this is not strictly true, as E' is only isomorphic to such a category – every object in E' also has a redundant copy of $\lambda x.!_{B(c(x))}$, for example. Thus, E and E' are not equal on the nose, but only isomorphic – but why would we expect anything else?]

Are the morphisms also the same? Let us consider morphisms from (A, B, c, d) to (A', B', c', d') in both categories. In E, they are $(f, g) : (A, B, c) \Rightarrow_{\text{BiAlg}(F,U)} (A', B', c')$ such that

$$g(c(x), d(x, y)) = d'(F(f, g)(x), G(f, g)(x, y)).$$

Spelt out, this means

- $(f, g) : (A, B) \Rightarrow_{\text{Fam(Set)}} (A', B')$, such that
- $p : f \circ c = c' \circ F(f, g)$, and
- $g(c(x), d(x, y)) = d'(F(f, g)(x), G(f, g, p)(x, y))$.

In E', we have morphisms $(f, g) : (A, B) \Rightarrow_{\text{Fam(Set)}} (A', B')$, a proof $p : f \circ c = c' \circ F(f, g)$ and a proof $q : (f, g) \circ (c, d) = (c', d') \circ (F(f, g), G(f, g, p))$. The first two conditions obviously correspond to the first two conditions for E. Spelling out the last condition, this means

- $f \circ c = c' \circ F(f, g)$ (again), and
- $g(c(x), d(x, y)) = d'(F(f, g)(x), G(f, g, p)(x, y))$,

which matches the last condition for E. Hence E and E' are isomorphic, and deciding which one to work with is a matter of taste.