Graded Generalized Algebraic Data Types

Harley Eades

October 13, 2022

Abstract

Write abstract

1 Introduction

Write the intro.

2 The Fundamental Theory

Suppose $\mathcal{C}$ is a category and $(\mathcal{E}, \otimes, I)$ is a strict monoidal category.

Definition 2.1 (Graded F-Algebra). *For a functor $\mathsf{F} : \mathcal{E} \times \mathcal{C} \longrightarrow \mathcal{C}$, a graded F-algebra is a pair (A, h) that consists of a functor $\mathsf{A} : \mathcal{E} \longrightarrow \mathcal{C}$ and a family h of morphisms:*

$$h_{m,n} : \mathsf{F}(m, \mathsf{A}(n)) \longrightarrow \mathsf{A}(m \otimes n)$$

*A **homomorphism** between two graded F-algebras (A, h) and (B, h') consists of a morphism*

$$\alpha : (\mathsf{A}, h) \longrightarrow (\mathsf{B}, h')$$

is defined as a natural transformation $\alpha : \mathsf{A}_1 \longrightarrow \mathsf{A}_2$ such that:

$$\mathsf{F}(\mathsf{m}, \alpha_{\mathsf{n}}); h'_{m,n} = h_{m,n}; \alpha_{m \otimes n}$$

Definition 2.2. *If the category of graded* F*-algebras has an initial object, then we call this a* ***graded initial*** F*-algebra denoted by* $(\mu\mathsf{F}, \mathsf{in})$*. That is, for any other* F*-algebra* (A, h) *there must be a unique morphism* $\alpha : (\mu\mathsf{F}, \mathsf{in}) \longrightarrow (\mathsf{A}, h)$*, but this implies that for any object* n*,* $\alpha_n : \mu\mathsf{F}(n) \longrightarrow \mathsf{A}(n)$ *is unique and* $\mu\mathsf{F}(n)$ *is an initial object in* $\mathcal{C}$*.*

Lemma 2.3 (From Structures to Homomorphisms)**.** *Given any graded* F*-algebra* (A, h)*, the structure map* h *is also a homomorphism between* F*-algebras* $(\mathsf{F}(m, A(-)), \mathsf{F}(-, h_{m,-}))$ *and* $(\mathsf{A}(m \otimes -)), h_{-,m\otimes -})$*.*

Proof. This proof holds trivially by writing out the commutative square for the F-algebra homomorphism. □

Lemma 2.4 (Graded Lambek's Lemma)**.** *If* (A, h) *is a graded initial algebra of* F*, then for any object* m*,* $\mathsf{A}(m \otimes -) : \mathcal{E} \longrightarrow \mathcal{C}$ *is isomorphic to* $\mathsf{F}(m, A(-)) : \mathcal{E} \longrightarrow \mathcal{C}$ *via* h_m*.*

Proof. Suppose $h_{m,n} : \mathsf{F}(m, A(n)) \longrightarrow \mathsf{A}(m \otimes n)$ is an initial algebra structure for any m and n. Now define an algebra structure:

$$\mathsf{F}(m', h_{m,n}) : \mathsf{F}(m', \mathsf{F}(m, A(n))) \longrightarrow \mathsf{F}(m', \mathsf{A}(m \otimes n))$$

Then by initiality there exists an F-algebra homomorphism

$$i_m : \mathsf{A}(m \otimes -) \longrightarrow \mathsf{F}(m, \mathsf{A}(-))$$

such that:

$$\mathsf{F}(\mathsf{m}', \mathsf{i}_{\mathsf{m},\mathsf{n}}); \mathsf{F}(m', h_{m,n}) = h_{m',(m\otimes n)}; i_{m',(m\otimes n)}$$

We also know that $h_m : \mathsf{F}(m, A(-)) \longrightarrow \mathsf{A}(m \otimes -)$ is itself a graded F-algebra homomorphism (Lemma 2.3). Thus, since we know that $\mathsf{A}(m \otimes n)$ is an initial object by definition and assumption that A is a graded initial object, and hence, $i_{m,n}; h_{m,n} = \mathsf{id}_{m\otimes n}$.

Next we know that i is a graded F-algebra homomorphism which implies

$$\mathsf{F}(m, i_{n,I}); \mathsf{F}(m, h_{n,I}) = h_{m,n}; i_{m,n}$$

but again by initiality we know that

$$\mathsf{F}(m, i_{n,I}); \mathsf{F}(m, h_{n,I}) = \mathsf{id}_{\mathsf{F}(m,\mathsf{A}(n))}$$

Therefore, i is the inverse of h and we obtain our result. □

Definition 2.5 (Graded Folds)**.** *Suppose* $(\mu\mathsf{F}, \mathsf{in})$ *is a graded initial* F*-algebra. Then the unquie map between* $\mu\mathsf{F}$ *and any other graded* F*-algebra* (A, h) *is the **fold** for* $\mu\mathsf{F}$ *and is denoted by*

$$\mathsf{fold}(h) : \mu\mathsf{F} \longrightarrow \mathsf{A}$$

Furthermore, we know that the following must hold:

$$\mathsf{in}_{m,n};\mathsf{fold}(h)_{m*n} = \mathsf{F}(m, \mathsf{fold}(h)_n);h$$

Definition 2.6 (Graded Forgetful Limits)**.** *There is a forgetful functor from the category of graded* F*-algebras and the functor category* $[\mathcal{E}, \mathcal{C}]$ *and their natural transformations. This functor is defined as follows:*

$$\begin{aligned} \mathsf{U_F}(\mathsf{A}, h) &= \mathsf{A} \\ \mathsf{U_F}(\alpha) &= \alpha \end{aligned}$$

Given an object of $[\mathcal{E}, \mathcal{C}]$*, say* X*, then a* $\mathsf{U_F}$***-cone for*** X *comprises, for every graded* F*-albebra* (A, h)*, a natural transformation* $v_{(\mathsf{A},h)} : \mathsf{X} \longrightarrow \mathsf{A}$ *in* $[\mathcal{E}, \mathcal{C}]$ *such that, for every graded* F*-algebra homomorphism* $\alpha : \mathsf{A} \longrightarrow \mathsf{B}$*, we have* $v_{(\mathsf{B},g)} = \alpha;v_{(\mathsf{A},h)}$*. We denote these cones by* (X, v) *and call* X *its **vertex** and* $v_{(A,h)}$ *the **projection** from* X *to* A*.*

A $\mathsf{U_F}$***-cone morphism*** $g : (\mathsf{X}, v_1) \longrightarrow (\mathsf{Y}, v_2)$ *is a natural transformation* $g : \mathsf{X} \longrightarrow \mathsf{Y}$ *such that for any graded* F*-algebra* (A, h)*, we have* $g;v_2 = v_1$*. A* $\mathsf{U_F}$***-limit*** *is a* $\mathsf{U_F}$*-cone to which there is a unique* $\mathsf{U_F}$*-cone morphism, call the **mediating morphism**, from any other* $\mathsf{U_F}$*-cone.*

Lemma 2.7 (Forgetful Limits are Unique)**.** *If* (X, v) *is a* $\mathsf{U_F}$*-limit, then it is unique up to isomorphism.*

Proof. Suppose (Y, v') is another $\mathsf{U_F}$-limit. Thus, there is a unique $\mathsf{U_F}$-cone morphism $i : \mathsf{Y} \longrightarrow \mathsf{X}$ such that $i;v = v'$. But, there must also be a unique $\mathsf{U_F}$-cone morphism $j : \mathsf{X} \longrightarrow \mathsf{Y}$ such that $j;v' = v$. But, by substitution $i;j;v' = v'$ and $j;i;v = v$, but these in addition to the assumption that both i and j are unique imply that $i;j = \mathsf{id_Y}$ and $j;i = \mathsf{id_X}$, and thus i and j are inverses of each other. □

2.1 Interpretation

- A data type is seen as the carrier of the initial algebra of a higher-order functor with type $(|\mathcal{C}| \to \mathcal{C}) \to |\mathcal{C}| \to \mathcal{C}$.

- Constructors have return types of the form $\mathsf{G}(h\,a)$, but initial algebra semantics requires them to be of the form $\mathsf{G}(a)$. To get an equivalent data type with constructors whose return types have this form we use left Kan extensions.

References