M.Sc. Thesis in Computing Science
Faculty of Science

Intrinsically correct sorting using bialgebraic semantics

G.C. (Cass) Alexandru
student number: s1045749

2023-10-04

First Supervisor:
Dr. Jurriaan Rot

Co-supervisor:
Ruben Turkenburg, M.Sc.

Second reader:
Prof. dr. J.H. (Herman) Geuvers

Radboud University

Abstract

The paper "Sorting with bialgebras and distributive laws" (Hinze et al., 2012) uses the category-theoretical framework of bialgebraic semantics to define sorting algorithms. However, the correctness of the algorithms isn't verified. I use basic categorical semantics of dependent types to define an intrinsically verified algorithm using the same framework. Additionally, I show that the coalgebras of the codomain are well-founded, which resolves an issue of the original construction.

Contents

1 Introduction

Sorting algorithms are arguably some of the most studied creatures in computer science. They are, after Euclid's algorithm, one of the first algorithms learners of programming are introduced to. As such, one might be forgiven to think that everything concerning sorting has already been said. Yet sorting algorithms are a surprisingly useful test for ideas that may have much broader reach and be far more abstract in their general case. So, for example, an extremely concise formulation [13] of the Quicksort algorithm has long been used as a poster child for the Haskell [22] programming language, and declar-

ative programming in general. In this vein, this treatise is a case study of an approach to intrinsically correct algorithm design using category theory, applied to sorting. We begin by relating the origin of the particular category-theoretical construction we'll be using, "bialgebraic semantics", and how it can be applied to algorithm design.

1.1 Background

The syntax of formal languages can be modeled via algebras for a functor, a notion from category theory. Syntax is defined using a *signature*, an assignment of operator symbols to arities. An example of syntax is the grammar of regular expressions for an alphabet A, which has the signature $\Sigma = \{\emptyset : 0, \varepsilon : 0, (\text{literal}_a : 0)_{a \in A}, \text{concat} : 2, | : 2, * : 1\}$. Regular expression terms and their interpretation as languages form algebras for the functor: $R(X) = 1 + 1 + A + (X \times X) + (X \times X) + X$. The *initial algebra* for this functor is the set of closed terms.

The categorical dual to algebras, coalgebras, on the other hand, can be used to model the behavior of systems. Universal coalgebra [24] is an approach of modeling many kinds of transition systems via coalgebras. An example of transition systems are deterministic finite automata for an alphabet A. They can be described as coalgebras for the functor $(-)^A \times 2$ (where the $(- \times 2)$ encodes whether a state is accepting and $(-)^A$ encodes for each letter which state to transition to).

In the seminal work [25] by Turi and Plotkin, the two notions are combined to describe SOS (structural operational semantics) specifications as *distributive laws* of syntax over behavior functors. *Bialgebraic semantics* refers to the doubly unique morphism, by initiality and finality, from the initial algebra for the syntax functor to the final coalgebra for the behavior functor lifted to bialgebras using a distributive law. This same construction was used in [7] to define sorting algorithms.

To understand how this notion from semantics made it to algorithmics we need to introduce some background on categorical algorithm design. The analog of syntax and behavior functors in the field of functional programming language analysis are so-called "base functors" for inductive and coinductive datatypes [4]. In the category Set of sets and functions, the carrier μF of an initial algebra for a base functor F models an *inductive datatype*, which contains all finite trees "of shape" F. The carrier νF of a final coalgebra for F models a *coinductive datatype*, which contains both finite and infinite trees of shape F. The well-known cons-list for elements of type A e.g., can be modeled as the carrier of the initial algebra of the base functor $L(X) = 1 + A \times X$. In this context, bialgebraic semantics can be used to define two morphisms, one by initiality and the other by finality, from an inductive datatype μF to a coinductive datatype νB, given a distributive law $FB \Rightarrow BF$.

1.2 Problem Statement and Contributions

Our goal is to define an intrinsically verified sorting algorithm. This involves encoding the specification into the type of the program itself, i.e. the algorithm is a term in a dependently typed language. By interpreting types dependent on values of a type X as

objects in the slice category Set/X, we can use the techniques of categorical algorithm design, in this case bialgebraic semantics, to define it.

In defining an intrinsically verified algorithm we follow the mantra "local steps carry proofs of local properties, which extend to the desired global properties". For datastructures that should maintain some structural invariant, the respective mantra is "local invariants of components extend to the global invariant".

The prior work we're building on top of is the paper "Sorting with bialgebras and distributive laws"[7]. There, sorting algorithms are defined using a distributive law (the local step) of the list base functor L over an "ordered" list base functor O (the local components). However, it comes with two caveats: The first is that the output list is no longer guaranteed to be finite. The second is that the correctness of the algorithm is unverified. By defining a dependently-typed version of their algorithm we extend their framework, resulting in the following novel contributions:

- We arrive at an intrinsically verified sorting algorithm by verifying only a local step (the distributive law), and proving that local properties of the base functors extend to the requisite global properties in the carriers of their initial algebra/final coalgebra (Sections 4 and 5).
- After locally encoding the properties of sorting in its type, there is *exactly one such distributive law* so we are arguably getting proven correct sorting "for free" (Section 5).
- We define the base functor of the output type in such a way that its coalgebras are well-founded, which makes the carrier of its final coalgebra finite lists (Section 4). This ensures that the output list of the sorting algorithm is finite.

2 Overview

We begin with the fundamentals of the specification of sorting. A sorting algorithm is a function from a list of elements of some total order $(A, \leq)$ to again a list of such elements.

The first rule of sorting is: The output list must be *ordered*, i.e. all pairs of consecutive elements are related to each other via the ordering relation $\leq$.

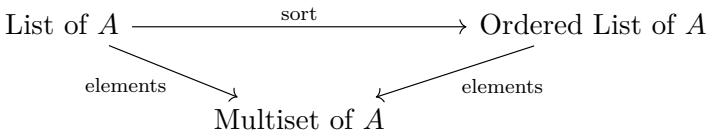
The second rule of sorting is: The output list is a permutation of the input list, i.e. sorting preserves elements. Note that this property relates the output to the input, which is not the case for the first property.

As our goal is to define an intrinsically verified algorithm using bialgebraic semantics, we need to break these two global properties down into local properties of the base functors L and O and the distributive law, $\sigma\colon LO \Rightarrow OL$, between them. The map we will obtain is of type $\mu L \to \nu O$, where μL is the carrier of the initial algebra for the base functor of the unordered list, and νO the carrier of the final coalgebra for the base functor of the ordered list. In the original construction used in [7], L and O are just aliases for the list base functor in Set, $(1 + A \times -)$. This means the resulting sorting

algorithm has type $A^* \to A^* + A^{\mathbb{N}}$, i.e. the output type is lists or infinite streams[1]. Both to encode the structural invariants needed for intrinsic verification and to ensure the final coalgebra of the ordered list base functor is well-founded, we need to work in a category with more structure than Set.

2.1 Element preservation

We start with examining the element preservation property, since the way we encode it gives the basis on top of which we define the orderedness property. It can be expressed globally as the requirement that the following diagram commutes:



This expresses that the mapping of a list to the multiset of its elements is invariant under sorting. A key insight of this thesis was to realize that this diagram is a morphism in the slice category $\mathsf{Set}/(\text{Multiset of } A)$, and that working in this slice category allows us to express all the properties we are interested in: element-preservation, orderedness, and wellfoundedness of the final coalgebra for the ordered list base functor.

We will write $\mathcal{M}(A)$ for "Multiset of A" in the following and use some notation and terminology introduced in Section 3.

To encode the element preservation property locally, that is as a property of the distributive law $LO \Rightarrow OL$, we need to express *element indexing* locally, i.e. that the base functors map to their elements, assuming their recursive arguments do. This means that, given an argument $(X, \text{element-index})$, its image under the base functor, which should have $(1 + A \times X)$ as its object component, should map to its respective elements in $\mathcal{M}(A)$.

Clearly, 1 should map to the empty multiset. The pair $(a, x) \colon A \times X$ should map to the insertion of a into the multiset index of x. We can express this as a definition sketch for the base functor L in $\mathsf{Set}/\mathcal{M}(A)$:

$$L(X, \text{element-index}) := (1 + A \times X, [\text{empty multiset}, \lambda(a, x).\, \text{insert}(a, \text{element-index}(x))])$$

2.2 Orderedness

Having established how we can locally express element preservation, it remains to express the orderedness property in addition to it in the base functor for ordered lists.

We can locally ensure orderedness by maintaining it as a structural invariant: we allow prepending an element to a list only if it is lesser than the head of the list to prepend to, or that list is the empty list. The final coalgebra of the base functor should then give us ordered lists (a fact which we verify in Section 4.3). We only *necessarily* need to index lists by their head, so that we can check that the element to prepend is smaller than

[1] See Theorem 3.8

the head of the list to prepend to. However, since in the construction so far we index by the entire multiset of the list's elements, and not only the head, we need to suitably extend the relation to that. The generalization of the requirement that the element to be prepended be smaller than the head of the list it is to be prepended to is that it be smaller than *all* the elements of that list, i.e. it is a lower bound to it. This is equivalent to the previous formulation due to the transitivity of $\leq$.

Combined with the element-indexing property, this adds up to the following definition sketch for the base functor O for ordered lists (the slice map is the same as the one in the definition sketch of L so we elide it here):

$$O(X, \text{element-index}) \coloneqq (1 + \{(a, x) \in A \times X \mid \text{is-lower-bound}(a, \text{element-index}(x))\}, \dots)$$

We proceed to work out the details of how exactly to define base functors that satisfy the definition sketches outlined above in a compositional way in Section 4. First though, we interleave a section on background to give the requisite preliminaries. A reader sufficiently familiar with the subject matter may choose to skip ahead to the Section 4 and refer back to Section 3 as needed.

3 Background, Definitions, Notation & Terminology

Over the course of this entire treatise we will be considering lists of elements from some set A on which there exists a linear (total) order $\leq$. We will make it clear when we are using this particular set A and when we are referring to some arbitrary set A.

We make a brief note on notation here: Cartesian products are denoted $X \times Y$ with projections $\pi_l \colon X \times Y \to X$, $\pi_r \colon X \times Y \to Y$, and tupling $\langle f, g \rangle \colon Z \to X \times Y$ for $f \colon Z \to X$ and $g \colon Z \to Y$.

Coproducts are denoted $X + Y$ with inclusions $\mathsf{inl} \colon X \to X + Y$, $\mathsf{inr} \colon Y \to X + Y$, and cotupling $[f, g] \colon X + Y \to Z$ for $f \colon X \to Z$ and $g \colon Y \to Z$.

Restrictions are denoted $f|_Z$ for $f \colon X \to Y$ and $Z \subseteq X$. Functions that are both restrictions and corestrictions are denoted $f \mid Z \to A$ for $f \colon X \to Y$, $Z \subseteq X$ and $A \subseteq Y$.

For categorical constructions that have both object- and morphism components, such as a functor $F \colon \mathcal{C} \to \mathcal{D}$, we use subscript 0 to refer specifically to their object- and subscript 1 to refer to their morphism component, e.g. $F_0 \colon \mathcal{C}_0 \to \mathcal{D}_0$, $F_1 \colon \mathcal{C}_1 \to \mathcal{D}_1$. We make occasional use of the *Structure Identity Principle* [1], in short, "isomorphic mathematical structures are structurally identical; i.e. have the same structural properties". This means that we may choose to replace objects by others which are isomorphic to them, without explicitly conjugating by the isomorphism.

Definition 3.1. We define the *multiset* on some set A as

$$\mathcal{M}(A) \coloneqq \{\varphi \colon A \to \mathbb{N} \mid \text{supp}(\varphi) \text{ is finite}\},$$

where $\text{supp}(\varphi) \coloneqq \{a \in A \mid \varphi(a) \neq 0\}$ is called the support of φ.

Notation 3.1.1. We write a multiset as a formal sum $m_1|a_1\rangle + \cdots + m_n|a_n\rangle$, where $m_i := \varphi(a_i) \in \mathbb{N}$ is called the multiplicity of the element a_i. The *join* of two multisets φ and ψ is defined as $\varphi \cup \psi := \lambda x : X.\, \varphi(x) + \psi(x)$. We denote the multiset with empty support with $\emptyset$. Multiset membership is defined as $x \in \varphi := \varphi(x) \neq 0$. We denote multisets with *nonempty* support with $\mathcal{M}_+(A)$ and note that $\mathcal{M}(A) \simeq 1 + \mathcal{M}_+(A)$. Singleton inclusion $\eta_A : A \to \mathcal{M}_+(A)$ is defined by $\eta_A(a) := 1|a\rangle$.

We refer to [9, Definition 4.1.1] as a reference for the above definition and notation.

3.1 Constructions in the Slice Category

Definition 3.2 (Slice category)**.** Given a category $\mathcal{C}$ and an object $C : \mathcal{C}$, the *slice category* $\mathcal{C}/C$ has:

- As objects pairs (X, g) where $X : \mathcal{C}$ and $g : X \to C$.
- As morphisms $(X_1, g_1) \to (X_2, g_2)$ $\mathcal{C}$-maps $f : X_1 \to X_2$ s.t. $g_2 \circ f = g_1$, i.e. the following diagram commutes:

$$\begin{array}{ccc} X_1 & \xrightarrow{\quad f \quad} & X_2 \\ & {\scriptstyle g_1}\searrow \quad \swarrow{\scriptstyle g_2} & \\ & C & \end{array}$$

Remark. We refer to the morphism components of slice objects as "slice maps", while we refer to the arrows between slice objects as "slice morphisms".

We may refer to a slice object (X, g) by just its slice map g.

Objects g of Set/X are equivalent to X-indexed families $\{A_x\}_{x \in X}$ of sets, where a particular set A_x corresponds to the inverse image of x under g. For example, vectors of element type A can be viewed equivalently as the family of length-indexed lists $\{A^n\}_{n \in \mathbb{N}}$ or as the object (A^*, length) in $\mathsf{Set}/\mathbb{N}$. This justifies our use of "indexed" when referring to slice objects.

We now introduce the base change functor and its left and adjoint. They will be used in Section 4.1. The reader need not be familiar with the concept of adjunction; throughout this treatise we only use one fact about left adjoints, namely that they preserve colimits [16]. The concept of pullback should however be familiar—should it not be we refer to e.g. [20].

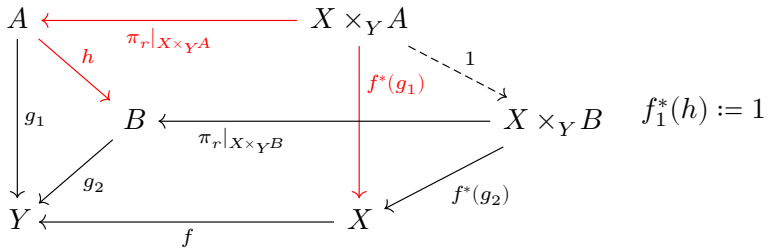
Definition 3.3 (Base change functor)**.** For $f : X \to Y$ there is an induced functor:

$$f^* : \mathsf{Set}/Y \to \mathsf{Set}/X$$

of slice categories. Its definition on objects (A, g) is given by the pullback of g along f:

$$\begin{array}{ccc} X \times_Y A & \xrightarrow{\pi_r|_{X \times_Y A}} & A \\ \downarrow{\scriptstyle \pi_l|_{X \times_Y A}} & & \downarrow{\scriptstyle g} \\ X & \xrightarrow{\quad f \quad} & Y \end{array} \qquad f_0^*(g) := \pi_l|_{X \times_Y A}$$

On morphisms, it is defined using the universal property of the pullback: A morphism $h\colon (A, g_1) \to (B, g_2)$ in Set/Y is mapped to the the unique morphism from the span $(X \times_Y A, h \circ \pi_r|_{X\times_Y A}, f^*(g_1))$ into the pullback $X \times_Y B$:



$$f_1^*(h) := 1$$

To obtain the unique morphism from $(X \times_Y A, h \circ \pi_r|_{X\times_Y A}, f^*(g_1))$ into the pullback $X \times_Y B$, we need to prove that it is a cone for the cospan (Y, f, g_2), i.e. $f \circ f^*(g_1) = g_2 \circ (h \circ \pi_r|_{X\times_Y A})$:

$$\begin{aligned} f \circ f^*(g_1) &= g_1 \circ \pi_r|_{X\times_Y A} && (X \times_Y A \text{ is the pullback for } f \text{ and } g_1) \\ &= g_2 \circ h \circ \pi_r|_{X\times_Y A} && (h\colon g_1 \to g_2 \text{ is a slice morphism, so } g_1 = g_2 \circ h) \end{aligned}$$

f^* preserves identities and composition by uniqueness.

Definition 3.4 (Dependent sum functor along f)**.** The dependent sum along $f\colon X \to Y$, $f_!\colon \mathsf{Set}/X \to \mathsf{Set}/Y$ is a functor, defined on objects as:

$$(X, h) \mapsto (X, f \circ h)$$

acting as postcomposition.

Lemma 3.1. *$f_!$ is left adjoint to f^**

Proof. We refer to [17]. □

Lemma 3.2. *f^* has a right adjoint f_*.*

Proof. We refer to [17]. □

Lemma 3.3. *The categorical product in Set/X of two objects $(A, g_1), (B, g_2)$ is given by the pullback $A \times_X B$:*

$$\begin{array}{ccc} A \times_X B & \xrightarrow{\pi_r|_{A\times_X B}} & B \\ \downarrow{\scriptstyle \pi_l|_{A\times_Y B}} & & \downarrow{\scriptstyle g_2} \\ A & \xrightarrow{g_1} & X \end{array}$$

With slice map equivalently $g_2 \circ \pi_r|_{A\times_X B}$ or $g_1 \circ \pi_l|_{X\times_Y A}$.

Proof. We refer to [21, Exercise 48]. □

Lemma 3.4. *Let* $(A, g) : \mathsf{Set}/X$. *Then the functor* $(g \times_X -) : \mathsf{Set}/X \to \mathsf{Set}/X$ *preserves colimits.*

Proof. This follows from the fact that $(g \times_X -)$ is equivalent to the composition of two functors: the dependent sum along g after the base change along g, i.e. $g \times_X - \simeq g_! \circ g^*$. Since these are both left adjoints their composition is likewise a left adjoint, and as such preserves colimits. □

3.2 Algebra and coalgebra

We briefly recall the definitions and some of the properties of (co)algebras. For details, see e.g. [11]. We also define some examples and prove some theorems that we will refer back to in Section 4.

3.2.1 Algebra

Given some endofunctor F on a category $\mathcal{C}$:

Definition 3.5. An F-algebra is an object X in $\mathcal{C}$ together with a map $g : FX \to X$. An F-algebra morphism from $g : FX \to X$ to $h : FY \to Y$ is a map $f : X \to Y$ s.t. the following diagram commutes:

$$\begin{array}{ccc} FX & \xrightarrow{Ff} & FY \\ \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle h} \\ X & \xrightarrow{f} & Y \end{array}$$

If there is an initial F-algebra $(\mu F, \mathrm{in})$, then for any algebra (X, g) the unique algebra morphism from "in" to g is called the *inductive extension* of g and denoted $(\!|g|\!)$:

$$\begin{array}{ccc} F\mu F & \overset{F(\!|g|\!)}{\dashrightarrow} & FX \\ \downarrow{\scriptstyle \mathrm{in}} & & \downarrow{\scriptstyle g} \\ \mu F & \overset{(\!|g|\!)}{\dashrightarrow} & X \end{array}$$

Definition 3.6. Consider the following sets:

$$\begin{aligned} A^n &:= \{\langle a_0, ..., a_{n-1}\rangle \mid a_i : A\} \\ A^+ &:= \sum_{n \in \mathbb{N}_{>0}} A^n \\ A^* &:= \sum_{n \in \mathbb{N}} A^n \end{aligned}$$

A^n is the type of lists of elements of type A of length n, A^+ nonempty lists and A^* all lists. We note that $A^* \simeq 1 + A^+$ and implicitly include 1 or A^+ in A^*. We define the function:

$$\begin{aligned} &\mathsf{cons} : A \times A^* \to A^+ \\ &\mathsf{cons}(a, \alpha = \langle a_0, ..., a_{n-1}\rangle) := \langle a, a_0, ..., a_{n-1}\rangle \end{aligned}$$

which prefixes a to α.

Theorem 3.5. $(A^*, [\langle\rangle, \mathsf{cons}])$ *is the initial algebra for the functor* $(1 + A \times -)$.

Proof. Let $(X, [b, n])$ be some $(1 + A \times -)$-algebra. Then we must show there is a unique algebra morphism f from $[\langle\rangle, \mathsf{cons}]$ to $[b, n]$, s.t. the following diagram commutes:

$$\begin{array}{ccc} 1 + A \times A^* & \xrightarrow{1+A\times f} & 1 + A \times X \\ {\scriptstyle [\langle\rangle,\mathsf{cons}]}\downarrow & & \downarrow{\scriptstyle [b,n]} \\ A^* & \xrightarrow[\exists f!]{} & X \end{array}$$

The commutativity of the above is, after chasing elements of type $1 + A \times A^*$ through it, equivalent to the equalities:

$$f(\langle\rangle) = b(\star)$$
$$f(\mathsf{cons}(a, r)) = n(a, f(r))$$

We take f to be the unique solution to this system of recursive equations. □

We use the initiality just proven to define a function which maps a list to the multiset of its elements:

Definition 3.7. Let $\mathsf{elt} := [\emptyset, \lambda(a, r).\, \eta_A(a) \cup r]$ be the algebra which maps 1 to the empty multiset and $A \times X$ to multiset insertion (singleton inclusion combined with union). Then elt uniquely extends to a $(1 + A \times -)$-algebra map $\mathsf{elts} := (\!|\mathsf{elt}|\!) \colon A^* \to \mathcal{M}(A)$ which maps a list to the multiset of its elements.

$$\begin{array}{ccc} 1 + A \times A^* & \xrightarrow{1+A\times(\!|\mathsf{elt}|\!)} & 1 + A \times \mathcal{M}(A) \\ {\scriptstyle [\langle\rangle,\mathsf{cons}]}\downarrow & & \downarrow{\scriptstyle \mathsf{elt}=[\emptyset,\cup\circ(\eta\times\mathrm{id})]} \\ A^* & \xrightarrow{\mathsf{elts}=(\!|\mathsf{elt}|\!)} & \mathcal{M}(A) \end{array}$$

Remark. "$\cup \circ (\eta_A \times \mathrm{id})$" is a pointfree way of writing $\lambda(a, r).\, \eta_A(a) \cup r$.

3.2.2 Coalgebra

Given some endofunctor B on a category $\mathcal{C}$:

Definition 3.8. A B-coalgebra is an object X in $\mathcal{C}$ together with a map $g \colon X \to BX$. A B-coalgebra morphism from $g \colon X \to BX$ to $h \colon Y \to BY$ is a map $f \colon X \to Y$ s.t. the following diagram commutes:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ BX & \xrightarrow{Bf} & BY \end{array}$$

If there is a final B-coalgebra $(\nu B, \mathrm{out})$, then for any algebra (X, g) the unique coalgebra morphism from g to "out" is called the *coinductive extension* of g and denoted $[\![g]\!]$:

$$\begin{array}{ccc} X & \overset{[\![g]\!]}{\dashrightarrow} & \nu B \\ \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle \mathrm{out}} \\ BX & \overset{B[\![g]\!]}{\dashrightarrow} & B\nu B \end{array}$$

Example 3.1. Consider the type of streams of elements of type A, $A^{\mathbb{N}}$ and the functions:

$$\begin{aligned} \mathsf{hd} &\colon A^{\mathbb{N}} \to A \\ \mathsf{hd}(a_0, a_1, \dots) &:= a_0 \\ \mathsf{tl} &\colon A^{\mathbb{N}} \to A^{\mathbb{N}} \\ \mathsf{tl}(a_i)_{i\in\mathbb{N}} &:= (a_{i+1})_{i\in\mathbb{N}} \end{aligned}$$

Theorem 3.6. $(A^{\mathbb{N}}, \langle \mathsf{hd}, \mathsf{tl}\rangle)$ *is the final coalgebra for the functor* $(A \times -)$.

Proof. Let $(X, \langle \mathsf{o}_X, \mathsf{tr}_X\rangle)$ be some $(A\times-)$-coalgebra. Then we must show there is a unique coalgebra morphism f from $\langle \mathsf{o}_X, \mathsf{tr}_X\rangle$ to $\langle \mathsf{hd}, \mathsf{tl}\rangle$, s.t. the following diagram commutes:

$$\begin{array}{ccc} X & \overset{\exists! f}{\dashrightarrow} & A^{\mathbb{N}} \\ \downarrow{\scriptstyle \langle \mathsf{o}_X, \mathsf{tr}_X\rangle} & & \downarrow{\scriptstyle \langle \mathsf{hd}, \mathsf{tl}\rangle} \\ A \times X & \overset{A\times f}{\dashrightarrow} & A \times A^{\mathbb{N}} \end{array}$$

The commutativity of the above is, after chasing an element $x \in X$ through it, equivalent to the equalities:

$$\begin{aligned} \mathsf{hd}(f(x)) &= \mathsf{o}_X(x) \\ \mathsf{tl}(f(x)) &= f(\mathsf{tr}_X(x)) \end{aligned}$$

We claim the unique solution to the above equations is f defined as:

$$f(x)(i) := \mathsf{o}_X(\mathsf{tr}_X^i(x))$$

We prove this by induction on i:

IB. $f(x)(0) = \mathsf{hd}(f(x)) = \mathsf{o}_X(x) = \mathsf{o}_X(\mathsf{tr}_X^0(x))$.

IS. $f(x)(i+1) = \mathsf{tl}(f(x))(i) = f(\mathsf{tr}_X(x))(i) \overset{\text{IH}}{=} \mathsf{o}_X(\mathsf{tr}_X^i(\mathsf{tr}_X(x))) = \mathsf{o}_X(\mathsf{tr}_X^{(i+1)}(x))$. □

The next final coalgebra we want to define is for the functor $(1 + A \times -)$. In this we follow roughly the structure of [9, Proposition 1.2.1]. The carrier of the final coalgebra will prove to be the type of either lists or streams. To define the morphism from some $(1 + A \times -)$-coalgebra (X, c) to the final coalgebra, we make a distinction on the number

of iterations c takes to reach 1. This number is in $\overline{\mathbb{N}} \coloneqq \mathbb{N} + \{\omega\}$, depending on whether 1 is reached or not. We define an iterable variant of c, $c^n \colon X \to 1 + X$ as:

$$\begin{aligned} c^0(x) &\coloneqq \mathsf{inr}(x) \\ c^{n+1}(x) &\coloneqq \begin{cases} \mathsf{inl}(\star) & c^n(x) = \mathsf{inl}(\star) \\ c(y) & c^n(x) = \mathsf{inr}(y) \end{cases} \end{aligned} \tag{1}$$

We introduce the following lemma:

Lemma 3.7. *Let (X, c) be a $(1 + A \times -)$-coalgebra. Consider the following sets:*

$$\begin{aligned} X_0 &\coloneqq \{x \in X \mid c(x) = \mathsf{inl}(\star)\} \\ X_n &\coloneqq \{x \in X \mid \forall i \leq n.\, c^i(x) \neq \mathsf{inl}(\star) \wedge c^{n+1}(x) = \mathsf{inl}(\star)\} \\ X_\omega &\coloneqq \{x \in X \mid \forall i.\, c^i(x) \neq \mathsf{inl}(\star)\} \\ X_* &\coloneqq \sum_{n \in \mathbb{N}} X_n \\ X_+ &\coloneqq \sum_{n \in \mathbb{N}_{>0}} X_n \end{aligned}$$

Then X can be split into the disjoint sets: $X \simeq X_0 + X_+ + X_\omega$ and c is equivalent, up to the isomorphism $1 + A \times X \simeq 1 + A \times X_* + A \times X_\omega$, to the sum of the following (co)restrictions:

$$c \simeq (c \mid X_0 \to 1) + \sum_{n \in \mathbb{N}_{>0}} (c \mid X_n \to A \times X_{n-1}) \;+\; (c \mid X_\omega \to A \times X_\omega)$$

As functions into a product, we may decompose $c|_{X_n}$ and $c|_{X_\omega}$ into $\langle \mathsf{o}_{X_n}, \mathsf{tr}_{X_n} \rangle$ and $\langle \mathsf{o}_{X_\omega}, \mathsf{tr}_{X_\omega} \rangle$, respectively.

Proof. We omit the proof that $X \simeq X_0 + X_+ + X_\omega$. We show the (co)restrictions are well-defined:

1. $c \mid X_0 \to 1$: By definition.
2. $c \mid X_n \to A \times X_{n-1}$: By reindexing.
3. $c \mid X_\omega \to A \times X_\omega$: By definition. □

We are now ready to define the final coalgebra for $(1 + A \times -)$:

Example 3.2. Consider the type of both streams and lists $A^\infty \coloneqq A^* + A^{\mathbb{N}}$ and the functions:

$$\begin{aligned} \mathsf{hd}_+ &\colon A^+ \to A \\ \mathsf{hd}_+(\mathsf{cons}(a, r)) &\coloneqq a \\ \mathsf{tl}_+ &\colon A^+ \to A^* \\ \mathsf{tl}_+(\mathsf{cons}(a, r)) &\coloneqq r \end{aligned}$$

Theorem 3.8. $((A^\infty \simeq 1 + A^+ + A^{\mathbb{N}}),\ \mathrm{id} + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle\ +\ \langle \mathsf{hd}, \mathsf{tl} \rangle)$ *is the final coalgebra for the functor* $(1 + A \times -)$.

Proof. Let (X, c) be some $(1 + A \times -)$-coalgebra. Then we must show there is a unique coalgebra morphism f from c to $\mathrm{id} + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle\ +\ \langle \mathsf{hd}, \mathsf{tl} \rangle$. We use Lemma 3.7 to rewrite c as the sum $c|_{X_0} + \sum_{n \in \mathbb{N}_{>0}} c|_{X_n}\ +\ c|_{X_\omega}$. To show that there exists a unique morphism f, it suffices to construct it as the direct product of a family of unique morphisms, namely $f = (f_0 \colon X_0 \to 1) + \sum_{n \in \mathbb{N}_{>0}} (f_n \colon X_n \to A^n)\ +\ (f_\omega \colon X_\omega \to A^{\mathbb{N}})$, such that the following diagrams commute:

$$\begin{array}{ccc}
X_0 & \xrightarrow{f_0} & 1 \\
\downarrow{\scriptstyle c|_{X_0}} & & \downarrow{\scriptstyle \mathrm{id}} \\
1 & \xrightarrow{\mathrm{id}} & 1
\end{array}
\qquad
\begin{array}{ccc}
X_n & \xrightarrow{f_n} & A^n \\
\downarrow{\scriptstyle \langle \mathsf{o}_{X_n}, \mathsf{tr}_{X_n} \rangle} & & \downarrow{\scriptstyle \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle} \\
A \times X_{n-1} & \xrightarrow{\mathrm{id} \times f_{n-1}} & A \times A^{n-1}
\end{array}
\qquad
\begin{array}{ccc}
X_\omega & \xrightarrow{f_\omega} & A^{\mathbb{N}} \\
\downarrow{\scriptstyle \langle \mathsf{o}_{X_\omega}, \mathsf{tr}_{X_\omega} \rangle} & & \downarrow{\scriptstyle \langle \mathsf{hd}, \mathsf{tl} \rangle} \\
A \times X_\omega & \xrightarrow{\mathrm{id} \times f_\omega} & A \times A^{\mathbb{N}}
\end{array}$$

f_ω is the same as the unique morphism of Theorem 3.6. We prove by induction on n that all the f_n exist and are unique:

IB. Clearly f_0 is unique as the arrow to the terminal object 1, and has type $X_0 \to 1$.

IS. f_n making the above diagram commute means that:

$$\begin{aligned} \mathsf{hd}_+(f_n(x)) &= \mathsf{o}_{X_n}(x) \\ \mathsf{tl}_+(f_n(x)) &= f_{n-1}(\mathsf{tr}_{X_n}(x)) \end{aligned}$$

We then define f_n as the unique solution of this set of equations. □

4 Base Functors for (un)ordered lists and their (co)initial (co)algebras

In this section we define base functors in $\mathsf{Set}/\mathcal{M}(A)$ that encode the local versions of the properties of element indexing and orderedness, and show that these local properties extend to the corresponding global properties in the initial algebra of unordered lists and the final coalgebra of ordered lists respectively.

In Section 4.1 we are concerned with defining base functors that encode the desired local properties. In Section 4.2 we show that for unordered lists, local element-indexing extends to global element-indexing in the initial algebra. In Section 4.3 we show that both element-indexing and orderedness extend to the respective global properties in the final coalgebra.

4.1 Base functors

Having introduced the requisite background, we can rephrase the definition sketches from Section 2.1 as:

$$L(X, g) = (1 + A \times X, [\emptyset, \cup \circ (\eta_A \times g)]) \tag{2}$$

$$O(X, g) = (1 + \{(a, x) \in A \times X \mid \forall e \in g(x).\, a \leq e\}, [\emptyset, \cup \circ (\eta_A \times g)]) \tag{3}$$

We recall that the base functor for lists in Set is $(1 + A \times -)$. We want to define the base functors in $\mathsf{Set}/\mathcal{M}(A)$ in a *compositional* way, i.e. lift the constituent components 1, +, A, and $\times$ individually from Set to $\mathsf{Set}/\mathcal{M}(A)$. This will allow for a compositional definition of the distributive law in Section 5.3 and allows us to use existing categorical constructions with known properties in defining the components.

We begin by lifting those that are the same for both functors from Set to $\mathsf{Set}/\mathcal{M}(A)$ s.t. they are indexed by their elements.

Definition 4.1 (Lifting 1 to $\mathsf{Set}/\mathcal{M}(A)$)**.** Let $\hat{1} \coloneqq (1, \emptyset)$, where $\emptyset \colon 1 \to \mathcal{M}(A)$ is the *generalized element* [18] $\emptyset \in \mathcal{M}(A)$.

Definition 4.2 (Lifting A to $\mathsf{Set}/\mathcal{M}(A)$)**.** Let $\hat{A} \coloneqq (A, \eta_A)$, where $\eta_X \coloneqq \lambda(x \colon X).\, 1|x\rangle$, i.e. the function that sends an element to the singleton multiset.

Definition 4.3. Let $+ \colon \mathsf{Set}^2 \to \mathsf{Set}$ be the functor that sends objects to their coproduct in Set. Then there is a functor $\oplus \colon (\mathsf{Set}/\mathcal{M}(A))^2 \to \mathsf{Set}/\mathcal{M}(A)$, defined on objects as:

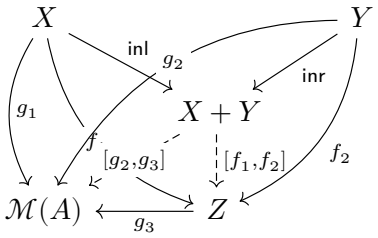
$$((X, g), (Y, h)) \mapsto ((X +_0 Y), [g, h])$$

And on morphisms as $(f_1, f_2) \mapsto f_1 +_1 f_2$.

The fact that it is a functor follows from the interaction laws of sum and cotupling.

Lemma 4.1. *$\oplus$ is the coproduct in $\mathsf{Set}/\mathcal{M}(A)$.*

Proof. Consider two objects $(X, g_1), (Y, g_2)$ in $\mathsf{Set}/\mathcal{M}(A)$. Consider some target object (Z, g_3) and slice morphisms $f_1 \colon (X, g_1) \to (Y, g_3)$, $f_2 \colon (Y, g_2) \to (Z, g_3)$. We claim that the unique morphism from the coproduct in Set, $[f_1, f_2] \colon X + Y \to Z$ is also the unique slice morphism of from $(X, g_1) \oplus (Y, g_2) = (X + Y, [g_1, g_2])$ to (Z, g_3). Uniqueness follows from its uniqueness in Set; thus it remains to show that it is a slice morphism, i.e. $g_3 \circ [f_1, f_2] = [g_1, g_2]$. This follows from cotupling laws and from the fact that f_1, f_2 are slice morphisms: $g_3 \circ [f_1, f_2] = [g_3 \circ f_1, g_3 \circ f_2] = [g_1, g_2]$.



□

It now remains to lift the product. In the definition sketch (2) we see that the slice maps of its arguments, (A, η_A) and (X, g), should be combined via $\cup$.

One construction which satisfies this is the tensor product obtained by lifting the monoidal category structure of $(\mathsf{Set}, \times, 1)$ to $\mathsf{Set}/\mathcal{M}(A)$ via the monoid $(\mathcal{M}(A), \cup, \emptyset)$:

Definition 4.4 ($\mathsf{Set}/\mathcal{M}(A)$ is a monoidal category)**.** $\mathsf{Set}/\mathcal{M}(A)$ inherits the monoidal structure of $(\mathsf{Set}, \times, 1)$ and $\mathcal{M}(A)$.

The tensor product $\otimes$ takes objects $g\colon X \to \mathcal{M}(A)$, $h\colon Y \to \mathcal{M}(A)$ to the morphism $X \times Y \xrightarrow{g \times h} \mathcal{M}(A) \times \mathcal{M}(A) \xrightarrow{\cup} \mathcal{M}(A)$, and the unit is given by $\hat{1}$.

Remark. This is a known construction from topos theory, see [19, Example 2.2].

To show that the tensor product $- \otimes -$ is a functor $(\mathsf{Set}/\mathcal{M}(A))^2 \to \mathsf{Set}/\mathcal{M}(A)$, we factor it into the composition of two functors:

Definition 4.5 (Induced product functor)**.** Let $\times\colon \mathsf{Set}^2 \to \mathsf{Set}$ be the functor that sends objects to their cartesian product. Then the induced product $\boxtimes\colon (\mathsf{Set}/\mathcal{M}(A))^2 \to \mathsf{Set}/(\mathcal{M}(A) \times \mathcal{M}(A))$ is a functor, defined on objects as:

$$((X, g), (Y, h)) \mapsto ((X \times_0 Y), g \times_1 h)$$

and on morphisms as $(f_1, f_2) \mapsto f_1 \times_1 f_2$. The fact that it is a functor follows from the functoriality of $\times$.

Secondly, we recall the dependent sum functor, acting as postcomposition (Definition 3.4). Using these two functors, the tensor product can be defined as the composition:

$$\otimes = \cup_! \circ \boxtimes$$

We now have all the components we need to define the base functor for unordered lists:

Definition 4.6. Let $(X, g)\colon \mathsf{Set}/\mathcal{M}(A)$. We define a functor $L\colon \mathsf{Set}/\mathcal{M}(A) \to \mathsf{Set}/\mathcal{M}(A)$:

$$L(X, g) := (1, \emptyset) \oplus (A, \eta_A) \otimes (X, g)$$

Remark.

- L is a functor as a composition of functors: $L = (\hat{1} \oplus -) \circ (\hat{A} \otimes -)$.
- If we expand the definitions of the components we arrive at our original definition sketch: $L(X, g) = (1 + A \times X, [\emptyset, \cup \circ (\eta_A \times g)])$.

In the definition sketch of the *ordered* list base functor, the object component of the product construction needs some additional structure. We proceed to define it in the following section.

4.1.1 The ordered slice product

We now want to find a compositional definition for the product component of O, outlined in its definition sketch (3). Given an argument $(X, g)\colon \mathsf{Set}/\mathcal{M}(A)$, its object component should be the set $\{(a, x) \in A \times X \mid \forall e \in g(x).\, a \leq e\}$. To separate the two arguments of the product, we first lift the relation $\leq$ itself to $\mathcal{M}(A) \times \mathcal{M}(A)$.

Remark. In the following, we will abbreviate $\mathcal{M}(A) \times \mathcal{M}(A)$ as $\mathcal{M}(A)^2$.

Definition 4.7 (Extending the relation)**.** Consider the poset $(A, \leq)$. We extend $\leq$ to a relation on $\mathcal{M}(A)$:

$$\sqsubseteq := \{(l, m) \in \mathcal{M}(A)^2 \mid \forall x \in l, \forall x \in m.\, x \leq y\}$$

Notation 4.7.1. We may also represent the relation $\sqsubseteq$ by its inclusion in the binary product $\mathcal{M}(A)^2$: $\sqsubseteq \xrightarrow{\sqsubseteq} \mathcal{M}(A)^2$.

Remark. Note that $\sqsubseteq$ is no longer itself a poset, since $\forall a, b \in \mathcal{M}(A).\, a \sqsubseteq \emptyset \sqsubseteq b$, which breaks transitivity and antisymmetry. However, since the multiset on the left will in our case always be a singleton, this doesn't pose a problem.

Using our definition of $\widehat{A} := (A, \eta_A)$ the object component becomes $\{(a, x) \in A \times X \mid \eta_A(a) \sqsubseteq g(x)\}$, which we generalize for arbitrary arguments $(X, g), (Y, h)$ as:

$$\{(x, y) \in X \times Y \mid g(x) \sqsubseteq h(y)\} \tag{4}$$

This equation can be expressed as the pullback of $g \times h$ along $\sqsubseteq$ in Set:

$$\begin{array}{ccc} (X \times Y) \times_{\mathcal{M}(A)^2} \sqsubseteq & \xrightarrow{\pi_r} & \sqsubseteq \\ \downarrow{\scriptstyle \pi_l} & \lrcorner & \downarrow{\scriptstyle \sqsubseteq} \\ X \times Y & \xrightarrow{g \times h} & \mathcal{M}(A)^2 \end{array}$$

We verify that this pullback expresses equation (4) by performing a diagram chase:

$$\begin{array}{ccc} ((x, y), c \sqsubseteq d) & \xmapsto{\pi_r} & c \sqsubseteq d \\ \downarrow{\scriptstyle \pi_l} & \lrcorner & \downarrow{\scriptstyle \sqsubseteq} \\ (x, y) & \xmapsto{g \times h} & (g(x), h(y)) = (c, d) \end{array}$$

We express this in more detail in the following lemma:

Lemma 4.2. *The canonical representation of the pullback* $(X \times_{\mathcal{M}(A)^2} Y)$, $P = \{((x, y), (c, d)) \mid (x, y) \in X \times Y, (c, d) \in \sqsubseteq, (g(x), h(y)) = (c, d)\}$ *and the set* $Q = \{(x, y) \in X \times Y \mid g(x) \sqsubseteq h(y)\}$ *are isomorphic with* $\pi_l \mid P \to Q$ *and* $\langle \mathrm{id}, g \times h \rangle \mid Q \to P$ *the components of the isomorphism.*

Proof. We show that the (co)restrictions are well-defined. The fact that they are inverses of each other follows by equality substitution.

- $\Rightarrow$ We have $((x, y), (c, d)) \in P$, i.e. $(c, d) \in \sqsubseteq$ and $(g(x), h(y)) = (c, d)$. Therefore $\pi_l((x, y), (c, d)) = (x, y) \in Q$.
- $\Leftarrow$ We have $(x, y) \in Q$ i.e. $g(x) \sqsubseteq h(y)$. Then $\langle \mathrm{id}, f \times g \rangle (x, y) = ((x, y), (f(x), g(y))) \in P$. $\square$

Using this lemma, we use the set Q in place of the canonical representation P as the pullback $X \times_{\mathcal{M}(A)^2} Y$ in the following.

By Lemma 3.3, the pullback in Set of $g \times h$ along $\sqsubseteq$ is the categorical product in $\mathsf{Set}/\mathcal{M}(A)^2$ of $g \boxtimes h$ and $\sqsubseteq$. We note that:

$$((X, g) \boxtimes (Y, h)) \times_{\mathcal{M}(A)^2} \sqsubseteq = (\{(x, y) \in X \times Y \mid g(x) \sqsubseteq h(y)\}, g \times h)$$

Finally, we want to combine the slice maps of the arguments via $\cup$. We can again use the dependent sum along $\cup$, $\cup_! \colon \mathsf{Set}/\mathcal{M}(A)^2 \to \mathsf{Set}/\mathcal{M}(A)$ to do this. This finally leads us to the definition of the ordered product functor:

Definition 4.8. Let $(X, g), (Y, h) \colon \mathsf{Set}/\mathcal{M}(A)$. We define a functor $\rtimes \colon (\mathsf{Set}/\mathcal{M}(A))^2 \to \mathsf{Set}/\mathcal{M}(A)$:

$$g \rtimes h := \cup_!((g \boxtimes h) \times_{\mathcal{M}(A)^2} \sqsubseteq)$$

Remark. The functoriality of $\rtimes$ follows from the fact that it is a composition of functors: $\rtimes = \cup_! \circ (- \times_{\mathcal{M}(A)^2} \sqsubseteq) \circ \boxtimes$.

We can now define the base functor for ordered lists:

Definition 4.9. Let $(X, g) \colon \mathsf{Set}/\mathcal{M}(A)$. We define a functor $O \colon \mathsf{Set}/\mathcal{M}(A) \to \mathsf{Set}/\mathcal{M}(A)$:

$$O(X, g) := (1, \emptyset) \oplus (A, \eta_A) \rtimes (X, g)$$

Remark.

- O is a functor as a composition of functors: $O = (\hat{1} \oplus -) \circ (\hat{A} \rtimes -)$.
- If we expand the definitions of the components we arrive at our original definition sketch: $O(X, g) = (1 + \{(a, x) \in A \times X \mid \eta_A(a) \sqsubseteq g(x)\}, [\emptyset, \cup \circ (\eta_A \times g)])$.

4.1.2 Distributivity of the slice products

Finally we prove that both $\otimes$ and $\rtimes$ distribute over $\oplus$. We will use this property to define functions in and out of composites involving these functors. We make many backreferences to Section 3.1 here.

Lemma 4.3. *Let* $(A, g) \colon \mathsf{Set}/D$, $(B, h_1), (C, h_2) \colon \mathsf{Set}/E$. *Then*

$$g \boxtimes (h_1 \oplus h_2) \simeq g \boxtimes h_1 \oplus g \boxtimes h_1$$

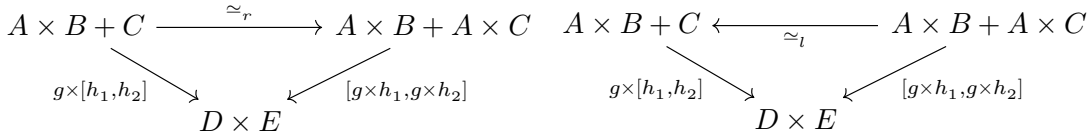
Proof. We assume that this distributivity follows from a general category-theoretical property but were regrettably unable to find one. Therefore we prove it manually. Since we are claiming this is an isomorphism in the slice category, the components of the isomorphism must be slice morphisms. If the components of a Set-isomorphism are slice morphisms then the Set-isomorphism extends to a slice isomorphism; this follows from the fact that the obvious forgetful functor $\mathcal{U} \colon \mathsf{Set}/X \to \mathsf{Set}$ is *conservative* [10, p. 7].

Thus the only proof obligation is to show that both components of the relevant Set-isomorphism are slice morphisms. We write out the Set-isomorphism $A \times (B + C) \simeq A \times B + A \times C$:

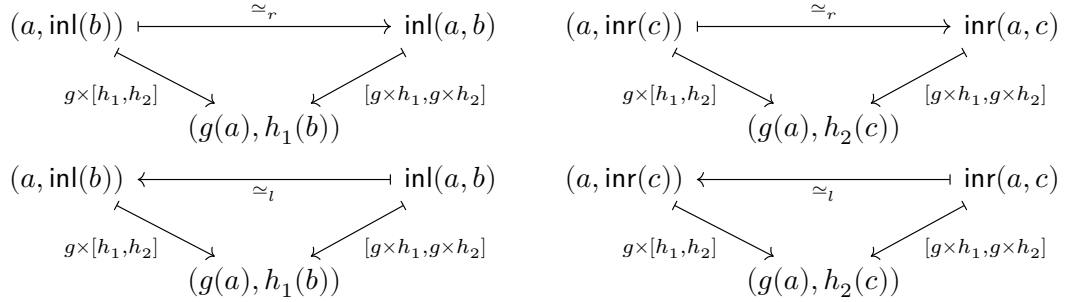
$$\begin{aligned}
&\simeq_r : A \times (B + C) \to A \times B + A \times C \\
&\simeq_r (a, \mathsf{inl}(b)) = \mathsf{inl}(a, b) \\
&\simeq_r (a, \mathsf{inr}(c)) = \mathsf{inr}(a, c)
\end{aligned}$$

$$\begin{aligned}
&\simeq_l : A \times B + A \times C \to A \times (B + C) \\
&\simeq_l (\mathsf{inl}(a, b)) = (a, \mathsf{inl}(b)) \\
&\simeq_l (\mathsf{inr}(a, c)) = (a, \mathsf{inr}(c))
\end{aligned}$$

We claim its components, $\simeq_r$ and $\simeq_l$, are slice morphisms:



This can be easily verified with suitable diagram chases:



□

Lemma 4.4. *Let $g, h_1, h_2 : \mathsf{Set}/\mathcal{M}(A)$. Then*

$$g \otimes (h_1 \oplus h_2) \simeq g \otimes h_1 \oplus g \otimes h_1$$

Proof. Writing out the definition of $\otimes$ we get:

$$\begin{aligned}
g \otimes (h_1 \oplus h_2) &:= \cup_!(g \boxtimes (h_1 \oplus h_2)) \\
&\simeq \cup_!((g \boxtimes h_1) \oplus (g \boxtimes h_2)) && \text{(Lemma 4.3)} \\
&\simeq \cup_!(g \boxtimes h_1) \oplus \cup_!(g \boxtimes h_2) \\
&\qquad (\cup_! \text{ preserves colimits as left adjoint (Lemma 3.1)}) \\
&=: g \otimes h_1 \oplus g \otimes h_2
\end{aligned}$$

□

Lemma 4.5. *Let* $g, h_1, h_2 : \mathsf{Set}/\mathcal{M}(A)$. *Then*

$$g \rtimes (h_1 \oplus h_2) \simeq g \rtimes h_1 \oplus g \rtimes h_2$$

Proof. Writing out the definition of $\rtimes$ we get:

$$\begin{aligned}
g \rtimes (h_1 \oplus h_2) &\coloneqq \cup_!((g \boxtimes (h_1 \oplus h_2)) \times_{\mathcal{M}(A)^2} \sqsubseteq) \\
&\simeq \cup_!((g \boxtimes h_1 \oplus g \boxtimes h_2) \times_{\mathcal{M}(A)^2} \sqsubseteq) && \text{(Lemma 4.3)} \\
&\simeq \cup_!((g \boxtimes h_1) \times_{\mathcal{M}(A)^2} \sqsubseteq \oplus (g \boxtimes h_2) \times_{\mathcal{M}(A)^2} \sqsubseteq) && \text{(Lemma 3.4)} \\
&\simeq \cup_!((g \boxtimes h_1) \times_{\mathcal{M}(A)^2} \sqsubseteq) \oplus \cup_!((g \boxtimes h_2) \times_{\mathcal{M}(A)^2} \sqsubseteq) && \text{(Lemma 3.1)} \\
&=: g \rtimes h_1 \oplus g \rtimes h_2
\end{aligned}$$

□

4.1.3 The (un)ordered base slice functors

We we recall the definitions of L and O (Definitions 4.6 and 4.9):

$$\begin{aligned}
L(X, g) &\coloneqq (1, \emptyset) \oplus (A, \eta_A) \otimes (X, g) \\
O(X, g) &\coloneqq (1, \emptyset) \oplus (A, \eta_A) \rtimes (X, g)
\end{aligned}$$

Or, if we expand the definitions:

$$\begin{aligned}
L(X, g) &= (1 + A \times X, [\emptyset, \cup \circ (\eta_A \times g)]) \\
O(X, g) &= (1 + \{(a, x) \in A \times X \mid \eta_A(x) \sqsubseteq g(x)\}, [\emptyset, \cup \circ (\eta_A \times g)])
\end{aligned}$$

They conform to the definition sketches we laid out in Section 2 to locally encode the properties of element-indexing and orderedness. It now remains to prove that:

1. Element-indexed lists are the initial L-algebra.
2. Element-indexed ordered lists are the final O-coalgebra.

We proceed to do this in the following two sections.

4.2 Element-Indexed Lists are the Initial L-Algebra

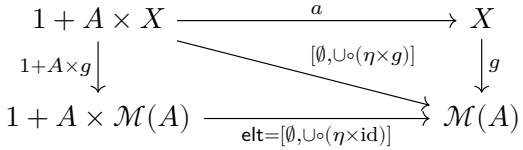
We need to show that local element-indexing extends to global element-indexing in the initial algebra, in order to get global element preservation as a property of the algorithm defined using a distributive law which locally preserves elements. We state that local element-indexing extends to global element-indexing in the initial algebra in the below theorem:

Theorem 4.6. $((A^*, \mathsf{elts}), [\langle\rangle, \mathsf{cons}])$ *is the initial algebra for the functor* $L = (\hat{1} \oplus \hat{A} \otimes -)$.

Proof. We recall that $\mathsf{elts} := (\!|\emptyset, \cup\circ(\eta_A \times \mathrm{id})|\!)$ (Definition 3.7). We introduce the following abbreviations:

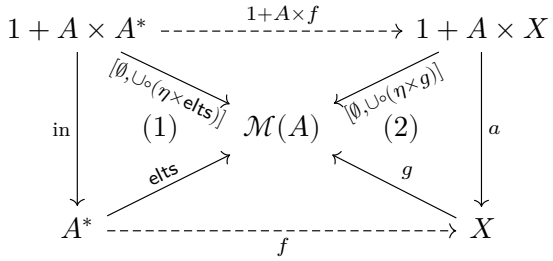
$$\begin{aligned}
\mathrm{in} &\colon 1 + A \times A^* \to A^* \\
\mathrm{in} &:= [\langle\rangle, \mathsf{cons}] \\
\mathsf{elt} &\colon 1 + A \times \mathcal{M}(A) \to \mathcal{M}(A) \\
\mathsf{elt} &:= [\emptyset, \cup \circ (\eta_A \times \mathrm{id})] \\
\mathsf{elts} &\colon A^* \to \mathcal{M}(A) \\
\mathsf{elts} &= (\!|\mathsf{elt}|\!)
\end{aligned}$$

The first thing to note is that L-algebras are equivalently $(1+A\times-)$-algebra-morphisms-to-elt. Namely, given an algebra $(\hat{1} \oplus \hat{A} \otimes (X, g)) \xrightarrow{a} (X, g)$, the slice map of its domain, $[\emptyset, \cup \circ (\eta_A \times g)]$ is equivalently $\mathsf{elt} \circ (1 + A \times g)$, and g is an $(1 + A \times -)$-algebra-morphism from a to elt:

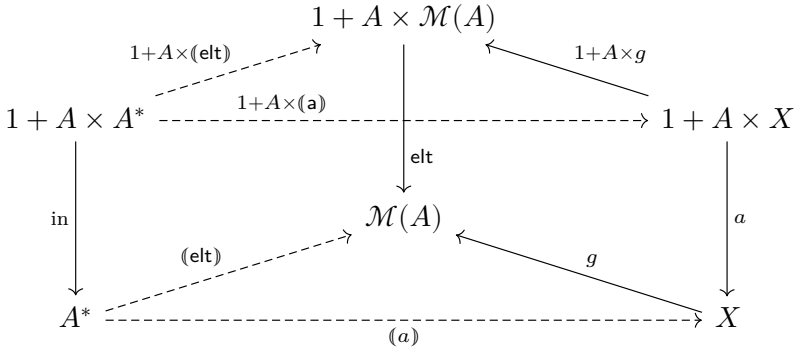


In fact, this illustrates that we are equivalently proving that $((A^*, \mathrm{in}), (\!|\mathsf{elt}|\!))$ is initial in the category $\mathsf{Alg}(1 + A \times -)/\mathsf{elt}$. The fact that $((A^*, \mathsf{elts}), \mathrm{in})$ is an L-algebra follows from the fact that $\mathsf{elts} = (\!|\mathsf{elt}|\!)$ is an $(1 + A \times -)$-algebra morphism to elt.

We still need to prove that $((A^*, \mathrm{in}), (\!|\mathsf{elt}|\!))$ is initial. Let $((X, g), a)$ be some L-algebra. Then there must be a unique algebra morphism from $((A^*, \mathsf{elts}), \mathrm{in})$ to $((X, g), a)$, i.e. the following diagram commutes (we already know that (1) and (2) do):



We define $f := (\!|a|\!)$ by initiality of (A^*, in). We can refactor our diagram to show that both $(\!|\mathsf{elt}|\!)$ and g are $(1 + A \times -)$-algebra morphisms to elt:



It remains to prove that $(\!|a|\!)$ is a slice morphism, i.e. $g \circ (\!|a|\!) = (\!|\mathsf{elt}|\!)$. But this follows from the uniqueness of $(\!|\mathsf{elt}|\!) \colon (A^*, \mathrm{in}) \to (\mathcal{M}(A), \mathsf{elt})$, since $g \circ (\!|a|\!)$ is a $(1 + A \times -)$-algebra morphism of the same type. □

4.3 Element-Indexed Ordered Lists are the Final O-Coalgebra

We need to show a total of three things about the final O-coalgebra:

1. The object component of its carrier is a subset of A^*, in particular not of A^∞.
2. It is indexed by its elements, i.e. its slice map is elts.
3. The object component of its carrier is ordered lists.

To point out why the first of these is not trivial, we recall Theorem 3.8, in which we showed that the final coalgebra of the Set-functor $(1 + A \times -)$ is both lists and streams $(A^* + A^{\mathbb{N}})$. The structure of our proof here will be similar to the one there. To define the morphism from some $(1 + A \times -)$-coalgebra $((X, g), c)$ to the final coalgebra, we need to make a distinction on the number of iterations c takes to reach 1. We will show that in this case, this is a natural number, i.e. 1 is always reached in a finite number of steps. This is the case because the slice map g acts as a *ranking* function for c.

Lemma 4.7. *Let $c \colon (X, g) \to \hat{1} \oplus \hat{A} \rtimes (X, g)$ be an O-coalgebra. We use the same iterable version of c, c^n, defined in* (1) *(Note that it doesn't need to be a slice morphism since it's only used internally). Consider the following sets:*

$$
\begin{aligned}
X_0 &\coloneqq \{x \in X \mid c(x) = \mathsf{inl}(\star)\} \\
X_n &\coloneqq \{x \in X \mid \forall i \leq n.\, c^i(x) \neq \mathsf{inl}(\star) \wedge c^{n+1}(x) = \mathsf{inl}(\star)\} \\
X_+ &\coloneqq \sum_{n \in \mathbb{N}_{>0}} X_n \\
X_* &\coloneqq X_0 + X_+
\end{aligned}
$$

Then $X \simeq X_*$.

Proof. We must show that from any state $x \in X$, the number of iterations c takes to reach 1 is a natural number, i.e. $\forall x \in X.\, \exists n.\, c^n(x) = \mathsf{inl}(\star)$. We do this by providing a *wellfoundedness* proof for c on $(\mathbb{N}, <)$ using $\mathsf{num} \circ g$ as a ranking function:

$$
\begin{aligned}
&\mathsf{num} \colon \mathcal{M}(A) \to \mathbb{N} \\
&\mathsf{num}(m_0|x_0\rangle + \ldots m_l|x_l\rangle) \coloneqq \sum_{i=0}^{l} m_i
\end{aligned}
$$

We know this sum is well-defined since $\mathcal{M}(A)$ has finite support. We make a case distinction on $c(x)$:

$c(x) = \mathsf{inl}(\star)$: Then $n = 0$.

$c(x) = \mathsf{inr}(a, r)$: Then c will be recursively called on r, therefore we must show that it is less than x by the ranking function. We know $g(x) = \eta_A(a) \cup g(r)$, thus $\mathsf{num}(g(r)) = \mathsf{num}(g(x)) - 1$, i.e. $(\mathsf{num} \circ g)(r) < (\mathsf{num} \circ g)(x)$. □

We now proceed to extend this factorization from X to c. This will enable us to define the map into the final coalgebra by induction.

Lemma 4.8. *Let $c\colon (X, g) \to \hat{1} \oplus \hat{A} \rtimes (X, g)$ be an O-coalgebra. Then c is equivalent, up to distributivity ($\hat{A} \rtimes X \simeq \hat{A} \rtimes X_0 \oplus \hat{A} \rtimes X_+$), to the direct product of the (co)restrictions:*

$$c \simeq (c \mid X_0 \to 1) + \sum_{n \in \mathbb{N}_{>0}} (c \mid X_n \to \hat{A} \rtimes X_{n-1})$$

Note that as functions into a product, we may decompose $c|_{X_n}$ into $\langle \mathsf{o}_{X_n}, \mathsf{tr}_{X_n} \rangle$.

Proof. We show the (co)restrictions are well-defined:

1. $c \mid X_0 \to 1$: By definition.
2. $c \mid X_n \to \hat{A} \rtimes X_{n-1}$: This means that from a state x that needs $n > 0$ iterations of c to reach 1, we end up in a state $c^1(x)$ that needs $n - 1$ iterations. The proof is by reindexing. □

We are now ready to define the final O-coalgebra:

Theorem 4.9. *Consider the following sets:*

$$\begin{aligned} S_n &:= (\{\sigma \in A^n \mid \forall i < n - 1.\, \sigma_i \leq \sigma_{i+1}\}, \mathsf{elts}) \\ S_+ &:= \oplus_{n \in \mathbb{N}_{>0}} S_n \\ S_* &:= \oplus_{n \in \mathbb{N}} S_n \end{aligned}$$

I.e. sets of ordered streams of element type A. Then $(S_* \simeq 1 + S_+, 1 + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle)$ is the final O-coalgebra.

Proof. In the following it is useful to specialize $\hat{A} \rtimes -$ to S_* in order to concretize the proof obligations. Thus, note that:

$$\hat{A} \rtimes S_* \ = (\{(a, \sigma) \in (A \times S_*) \mid \eta_A(a) \sqsubseteq \mathsf{elts}(\sigma)\}, [\emptyset, \cup \circ (\eta_A \times \mathsf{elts})])$$

1. We must first show that $(S_*, 1 + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle)$ is an O-coalgebra. This generates the following proof obligations:
   a) The (co)restriction of $\mathsf{tl}_+ : A^+ \to A^*$ to $S_+ \to S_*$ is well-defined, i.e. consecutive elements are still related by $\leq$. It is clear that this holds, since the tail operation performs reindexing, which preserves relations between consecutive elements.

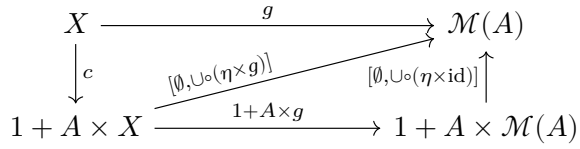
b) Arising from the $\sqsubseteq$ constraint of $(\widehat{A} \rtimes -)$:

$$\forall \sigma = \langle a_0, ..., a_{n-1} \rangle \in S_+.\ \eta_A(\mathsf{hd}_+(\sigma)) \sqsubseteq \mathsf{elts}(\mathsf{tl}_+(\sigma))$$

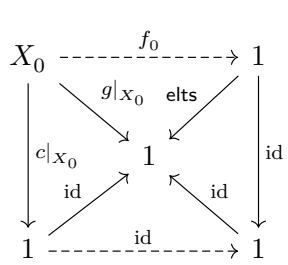
This means that the head of the list should be smaller than all the elements of the tail. This holds due to the transitivity of $\leq$ and the fact that all consecutive elements of σ are related by $\leq$. In the case that $n = 1$ it is vacuously true.

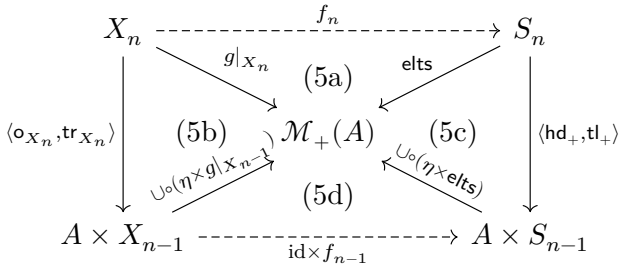
c) $1 + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle$ is a slice morphism, i.e. $(1 + \cup \circ (\eta_A \times \mathsf{elts})) \circ (1 + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle) = (1 + \mathsf{elts})$ holds. This factors into the separate obligations $1 \circ 1 = 1$ and $(\cup \circ (\eta_A \times \mathsf{elts}) \circ \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle)(\sigma) = \eta_A(\mathsf{hd}_+(\sigma)) \cup \mathsf{elts}(\mathsf{tl}_+(\sigma)) = \mathsf{elts}(\sigma)$. The latter follows from the definitions of the functions elts, hd_+ and tl_+.

2. We must show that $(S_*, 1 + \langle \mathsf{hd}_+, \mathsf{tl}_+ \rangle)$ is the *final* O-coalgebra. To this end, consider the following arbitrary $(\widehat{1} \oplus \widehat{A} \rtimes -)$-coalgebra:



Using Lemma 4.8, to show that there exists a unique morphism f, it suffices to construct it as the direct product of a family of unique morphisms, namely $f = (f_0 \colon X_0 \to 1) + \sum_{n \in \mathbb{N}_{>0}} (f_n \colon X_n \to S_n)$, such that the following diagrams commute:





We will write $g|_{X_n}$ as g_n in the following. To prove is that all the f_n exist, are well-defined and unique, and are slice morphisms, i.e. $\mathsf{elts} \circ f_n = g_n$ holds. We prove this by induction on n:

IB. Clearly f_0 is unique as the arrow to the terminal object 1. Also $\mathsf{elts}|_1 \circ f_0|_{X_0} = 1 \circ 1 = 1 = g|_{X_0}$.

IS. f_n making the outside of diagram (5) commute means that:

$$\mathsf{hd}_+(f_n(x)) = \mathsf{o}_{X_n}(x) \tag{6}$$

$$\mathsf{tl}_+(f_n(x)) = f_{n-1}(\mathsf{tr}_{X_n}(x)) \tag{7}$$

We take this as a definition of f_n in terms of f_{n-1}. We must show that:

i. $f_n \colon X_n \to S_n$ is well-typed, i.e.: Let $x \in X_n$, then $\forall i. < n-1. f_n(x)_i \leq f_n(x)_{i+1}$. This is only meaningful to prove for $n > 1$, so we will assume $n > 1$ in the following. First we write out what it means for $c|_{X_n}$ to map into $\hat{A} \rtimes X_{n-1}$:

$$\begin{aligned}
&\eta_A(\mathsf{o}_{X_n}(x)) \sqsubseteq g_{n-1}(\mathsf{tr}_{X_n}(x)) && \Leftrightarrow \\
&\eta_A(\mathsf{o}_{X_n}(x)) \sqsubseteq (\cup \circ (\eta_A \times g_{n-2}) \circ \langle \mathsf{o}_{X_{n-1}}, \mathsf{tr}_{X_{n-1}} \rangle)(\mathsf{tr}_{X_n}(x))) && \Leftrightarrow \text{(5b)} \\
&\eta_A(\mathsf{o}_{X_n}(x)) \sqsubseteq \eta_A(\mathsf{o}_{X_{n-1}}(\mathsf{tr}_{X_n}(x))) \cup g_{n-2}(\mathsf{tr}_{X_{n-1}}(\mathsf{tr}_{X_n}(x))) && \Leftrightarrow \\
&\quad \mathsf{o}_{X_n}(x) \leq \mathsf{o}_{X_{n-1}}(\mathsf{tr}_{X_n}(x)) && \wedge \\
&\eta_A(\mathsf{o}_{X_{n-1}}(x)) \sqsubseteq g_{n-2}(\mathsf{tr}_{X_{n-1}}(\mathsf{tr}_{X_n}(x)))
\end{aligned} \tag{8}$$

Our induction hypothesis is that f_{n-1} is well-typed, i.e.: IH: $\forall x \in X_{n-1}, \forall i < (n-1)-1. f_{n-1}(x)_i \leq f_{n-1}(x)_{i+1}$. Basically our proof obligation follows from using the induction hypothesis for i in the range 1–$(n-1)$, and equation (8) to relate the head to the second element, so for $i = 0$.

We prove that $\forall x \in X_n$, $\forall i. < n-1. f_n(x)_i \leq f_n(x)_{i+1}$ by a case distinction on i:

$i = 0$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{o}_{X_n}(x') \leq \mathsf{o}_{X_{n-1}}(\mathsf{tr}_{X_n}(x'))}\,(8)}{\mathsf{o}_{X_n}(x') \leq \mathsf{hd}_+(f_{n-1}(\mathsf{tr}_{X_n}(x')))}\,\text{Rewrite with (6)}}{\mathsf{hd}_+(f_n(x')) \leq \mathsf{hd}_+(\mathsf{tl}_+(f_n(x')))}\,\text{Rewrite with (6), (7)}}{f_n(x')_0 \leq f_n(x')_1}\,\text{Definition of } \mathsf{hd}_+/\mathsf{tl}_+}{\forall x \in X_n. f_n(x)_0 \leq f_n(x)_1}\,\forall \mathrm{I}(x')$$

$i \in [1, n-1]$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall i < (n-1)-1. f_{n-1}(\mathsf{tr}_{X_n}(x'))_i \leq f_{n-1}(\mathsf{tr}_{X_n}(x'))_{i+1}}\,\mathrm{IH}[x \backslash \mathsf{tr}_{X_n}(x')]}{\forall i. 0 < i < n-1 \Rightarrow f_{n-1}(\mathsf{tr}_{X_n}(x'))_{i-1} \leq f_{n-1}(\mathsf{tr}_{X_n}(x'))_i}\,\text{Reindexing}}{\forall i. 0 < i < n-1 \Rightarrow f_n(x')_i \leq f_n(x')_{i+1}}\,\text{Rewrite with (7)}}{\forall x \in X_n. \forall i. 0 < i < n-1 \Rightarrow f_n(x)_i \leq f_n(x)_{i+1}}\,\forall \mathrm{I}(x')$$

ii. $\mathsf{elts} \circ f_n = g_n$ holds.

By the inductive hypothesis we know $\forall x \in X_{n-1}. \mathsf{elts}(f_{n-1}(x)) = g_{n-1}(x)$.

We introduce an argument $x \in X_n$. Then to prove is:

$$\begin{aligned}
\mathsf{elts}(f_n(x)) &= \eta_A(\mathsf{hd}_+(f_n(x))) \cup \mathsf{elts}(\mathsf{tl}_+(f_n(x))) && \text{(5c)} \\
&= \eta_A(\mathsf{o}_{X_n}(x)) \cup \mathsf{elts}(f_{n-1}(\mathsf{tr}_{X_n}(x))) && \text{(6, 7)} \\
&= \eta_A(\mathsf{o}_{X_n}(x)) \cup g_{n-1}(\mathsf{tr}_{X_n}(x)) && \text{(IH)} \\
&= g_n(x) && \text{(5b)}
\end{aligned}$$

□

5 Verified Sorting with a Distributive Law

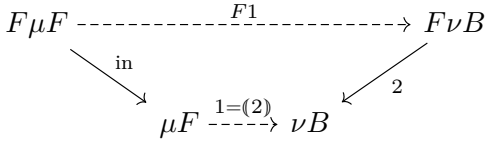
Having defined the base functors and shown that the local properties of element-indexing and orderedness they encode extend to the desired global properties in the carriers of the initial algebra for L and the final coalgebra for O, respectively, we must complete the final step of defining a morphism between them using bialgebraic semantics.

In Section 5.1 we give an abstract summary of bialgebraic semantics, which is used as a categorical algorithm design pattern in [7] (For an explanation that is specific to sorting, we refer the reader to [7]). In Section 5.2 we show how a distributive law between the composition of list base functors can be factored into simpler components using distributivity. Finally in Section 5.3 we show that there can be exactly one distributive law of type $LO \Rightarrow OL$, from which we get an intrinsically correct sorting algorithm.

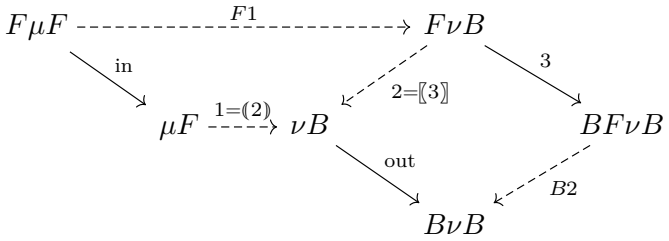
5.1 Bialgebraic Semantics

The sorting algorithms in [7] are defined using bialgebraic semantics as a categorical algorithm design pattern. This pattern can arise when one wants to define a function from the carrier of an initial algebra $(\mu F, \mathrm{in})$ for some functor F to the carrier of the final coalgebra $(\nu B, \mathrm{out})$ for a functor B. In the following we attempt to provide a step-by-step derivation of how one might arrive at this pattern.

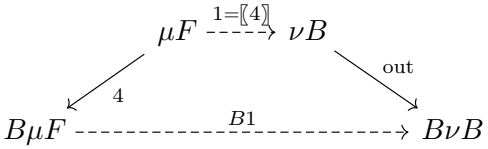
By having an initial algebra as domain and a final coalgebra as codomain, we get two ways to define such a function: As the inductive extension of an F-algebra with carrier νB, or the coinductive extension of a B-coalgebra with carrier μF. We start with the first approach. Let us first add the F-algebra "in" and the yet-to-be-defined F-algebra 2 with carrier νB to the picture:



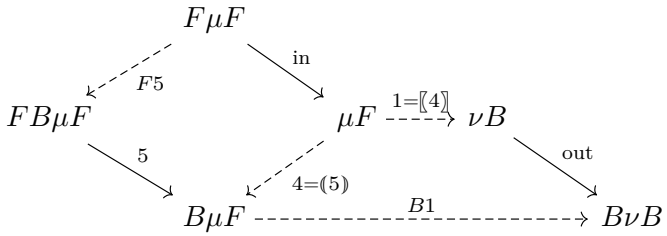
Now, 2 is an arrow *into* νB and as such we can define it as the coinductive extension of a B-coalgebra with carrier $F\nu B$:



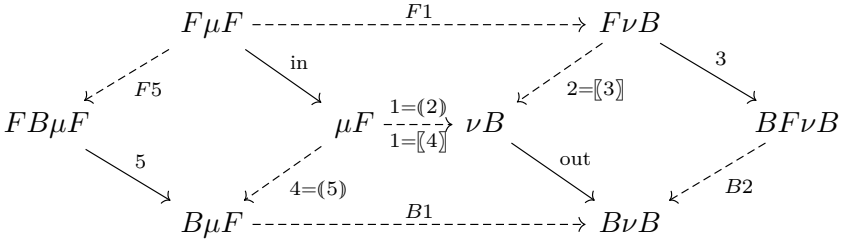
It remains to define this B-coalgebra $3 \colon F\nu B \to BF\nu B$, but first we look at the second possibility of defining the morphism 1 as the coinductive extension of a B-coalgebra with carrier μF:



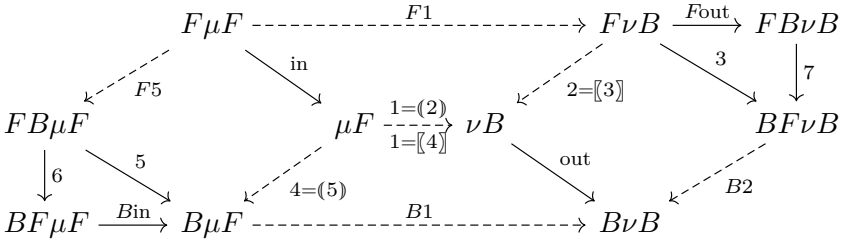
Now, 4 is an arrow *out of* μF and as such we can define it as the inductive extension of an F-algebra with carrier $B\mu F$:



Now, if we combine the partial diagrams of the two approaches, we get the following diagram, with 3 and 5 still left to define:



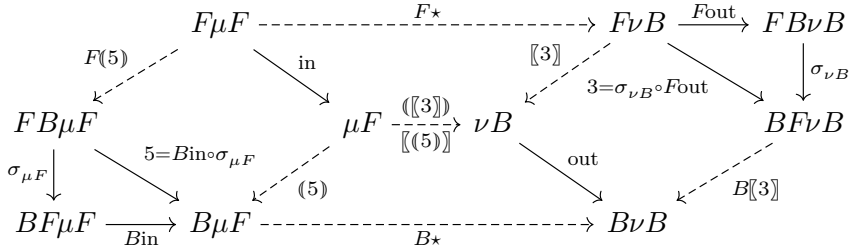
We can get part of the way to defining them, in the case of 3 by working forwards from its domain using "out" lifted by F, and in the case of 5 by working backwards from its codomain using "in" lifted by B:



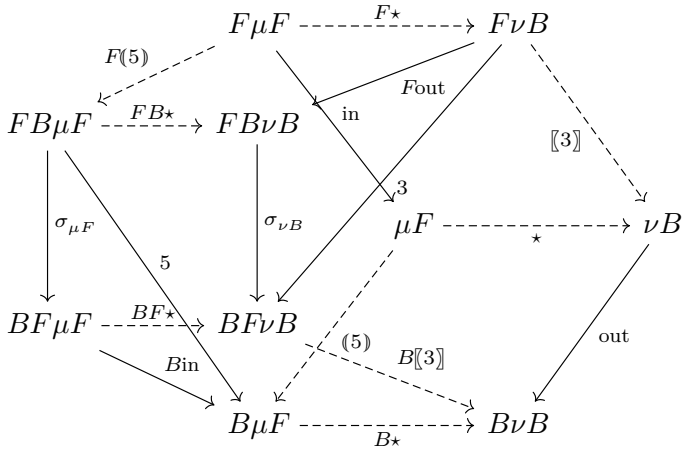
Now, notice that 6 and 7 have the type of components of a single natural transformation $\sigma \colon FB \Rightarrow BF$, at μF and νB, respectively. Since the morphisms 1 and 2 will now be equal[2], we replace them in their lifted occurrences with $\star$. Putting it all together we get

[2]For an explanation why, we refer to the works we cite at the end of this section.

the following diagram:



The above diagram looks like, in fact it *is* a 2D version of a 3D diagram, with some roundabout connections missing. The diagram is the following:



Remark. We refer to [11] and [3, Section 3.2] for a more complete treatment of bialgebraic semantics. For a formalization in Agda, see my contribution [2] to the `agda-categories` library [8].

5.2 Factoring the distributive law using distributivity

In this section only, we will shadow our definitions of L and O as functors in $\mathsf{Set}/\mathcal{M}(A)$ with L and O as aliases for the list base functor on Set, $(1 + A \times -)$, which is how they are defined in [7]. We will also shadow our definitions of $\times_O$ and $\times_L$ with aliases for the product component of O and L, which is just $\times$ in Set.

In [7] the distributive law $\delta\colon LO \Rightarrow OL$ at X is defined as a morphism:

$$
\begin{aligned}
&\delta_X : 1 + A \times_L (1 + A \times_O X) \to 1 + A \times_O (1 + A \times_L X)\\
&\delta_X\,(\mathsf{inl}(\star)) &&:= \mathsf{inl}(\star)\\
&\delta_X\,(\mathsf{inr}(a, \mathsf{inl}(\star))) &&:= \mathsf{inr}(a, \mathsf{inl}(\star))\\
&\delta_X\,(\mathsf{inr}(a, \mathsf{inr}(b, x))) &&:= \begin{cases} \mathsf{inr}(a, \mathsf{inr}(b, x)) & a \le b \\ \mathsf{inr}(b, \mathsf{inr}(a, x)) & b \le a \end{cases}
\end{aligned}
$$

We find this monolithic definition a bit hard to take in all at once. Using distributivity of $\times$ over $+$ we rewrite the type of δ_X to:

$$\delta_X \colon 1 + A \times_L 1 + A \times_L (A \times_O X) \to 1 + A \times_O 1 + A \times_O (A \times_L X)$$

We can now factor δ_X into the parallel composition of three morphisms:

$$\begin{array}{ccccc} 1 & + & A \times_L 1 & + & A \times_L (A \times_O X) \\ \downarrow{\scriptstyle \delta_{1_X}} & + & \downarrow{\scriptstyle \delta_{s_X}} & + & \downarrow{\scriptstyle \delta_{p_X}} \\ 1 & + & A \times_O 1 & + & A \times_O (A \times_L X) \end{array}$$

Such that $\delta_X \simeq \delta_{1_X} + \delta_{s_X} + \delta_{p_X}$. They are defined as:

$$\begin{aligned} \delta_{1_X} &:= \mathrm{id}_1 \\ \delta_{s_X} &:= \mathrm{id}_{A \times 1} \\ \delta_{p_X}(a, (b, x)) &:= \begin{cases} (a, (b, x)) & a \leq b \\ (b, (a, x)) & b \leq a \end{cases} \end{aligned}$$

We argue that this is a much more concise presentation. More importantly, in the following section, where the components carry proofs, these are much easier to write for the components individually than they would be for the non-factorized law.

Via bialgebraic semantics, as outlined in the previous section, given $\delta \colon LO \Rightarrow OL$, we get a map of type $\mu L \to \nu O$. As L and O are aliases for $(1 + A \times -)$, we get a map between the carriers of the initial and final algebras of that functor, thus, by Theorems 3.5 and 3.8, a map of type $A^* \to A^* + A^{\mathbb{N}}$.

In the next section we see how our base functor definitions from Section 4.1 let us define a map in the same way, which will however have a type that guarantees it to be a sorting algorithm.

5.3 Sliced Distributive Law

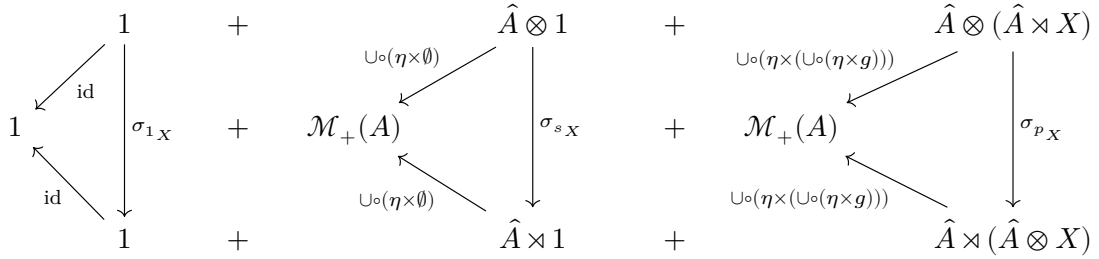
In Sections 4.2 and 4.3 we showed that the local properties of element-indexing and orderedness extend to the carriers of the (co)initial (co)algebras for the (un)ordered list slice base functors. We are now ready to use the base functors in which we encoded these properties to define a distributive law which will extend via bialgebraic semantics to a slice morphism between the carriers of the initial algebra and final coalgebra.

As it turns out, the constraints imposed by the base functors ensure that there is exactly one inhabitant of the type $LO \Rightarrow OL$, which we show in the following theorem:

Theorem 5.1. *There is a unique distributive law $\sigma \colon LO \Rightarrow OL$, where L and O are the endofunctors on $\mathsf{Set}/\mathcal{M}(A)$ defined in 4.6 and 4.9.*

Proof. Let (X, g) be an object in $\mathsf{Set}/\mathcal{M}(A)$. Since we are defining a natural transformation, this is a family of slice morphisms: $\sigma_X \colon \hat{1} \oplus \hat{A} \otimes (\hat{1} \oplus \hat{A} \rtimes X) \to \hat{1} \oplus \hat{A} \rtimes (\hat{1} \oplus \hat{A} \otimes X)$.

Using distributivity of $\otimes$ and $\rtimes$ over $\oplus$ (Lemmata 4.4 and 4.5), we can factor σ_X into the parallel composition of three morphisms:



We define and check that each component is a slice morphism and show that it is the *unique* inhabitant of its respective type:

σ_{1_X}: There is exactly one (slice) morphism of type $1 \to 1$ namely id_1.

σ_{s_X}: We first note that $\cup \circ (\eta_A \times \emptyset) = \eta_A \circ \pi_l$ and use the latter as the slice map in the following. We know that $(\pi_r \circ \sigma_{s_X})(x) \colon 1$ so $(\pi_r \circ \sigma_{s_X})(x) = \star$. Secondly, σ_{s_X} being a slice morphism requires that $\eta_A \circ \pi_l \circ \sigma_{s_X} = \eta_A \circ \pi_l$ i.e. $\pi_l \circ \sigma_{s_X} = \pi_l$. Thus, $\sigma_{s_X}(a, \star) = (a, \star)$. It remains to show that $\eta_A(a) \sqsubseteq \emptyset$; this holds by definition of $\sqsubseteq$.

σ_{p_X}: We assign variables to the components of the result tuple to avoid wrangling nested projections. Let $(c, (d, e)) \coloneqq \sigma_{p_X}(a, (b, x))$. We know the equality $\{a, b\} \cup g(x) = \{c, d\} \cup g(e)$ must hold. We don't know anything about the type X, therefore we cannot map x to anything other than itself, so the equality $e = x$ must hold. This results in the remaining constraint $\{a, b\} = \{c, d\}$. There are now two possible things that $\sigma_{p_X}(a, (b, x))$ could be equal to while satisfying the initial equality: $(a, (b, x))$ or $(b, (a, x))$.

This is where the $\sqsubseteq$ constraint comes into play. We have $\eta_A(c) \overset{!}{\sqsubseteq} \eta_A(d) \cup g(x)$. i.e. $c \leq d \wedge \forall l \in g(x).\ c \leq e$. We know that $\eta_A(b) \sqsubseteq g(x)$, i.e. $\forall l \in g(x).\ b \leq l$. Furthermore, due to the *linearity* of $\leq$ we know that either $a \leq b$ or $b \leq a$.

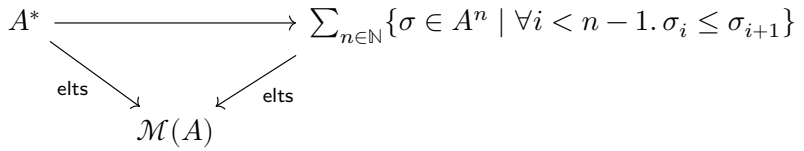
- $a \leq b$ Then $\sigma_{p_X}(a, (b, x))$ must equal $(a, (b, x))$. Namely, $\eta_A(a) \sqsubseteq \eta_A(b) \cup g(x)$ since $a \leq b$, and from $\forall l \in g(x).\ b \leq l$ we have $\forall l \in g(x).\ a \leq l$ by transitivity of $\leq$. Also it cannot equal $(b, (a, x))$, since $\eta_A(b) \sqsubseteq \eta_A(a) \cup g(x))$ doesn't hold, e.g. for $\leq = \leq_{\mathbb{N}}$, $a = 1, b = 2$ and g, x arbitrary, $\eta_A(2) \not\sqsubseteq \eta_A(1) \cup g(x)$.
- $b \leq a$ Then $\sigma_{p_X}(a, (b, x))$ must equal $(b, (a, x))$. The proof is the same as the above, with a and b switched.

Thus, the only way to define σ_{p_X} such that it is well-typed and σ is natural, is the following:

$$\sigma_{p_X}(a, (b, x)) = \begin{cases} (a, (b, x)) & a \leq b \\ (b, (a, x)) & b \leq a \end{cases}$$

σ is natural as the sum of *parametric* functions, i.e. functions that do not inspect the index X. □

Using bialgebraic semantics, by Theorems 4.9 and 4.6, we obtain a slice morphism of type:



This morphism has our desired global properties since:

- The orderedness of the output follows from the structure of the carrier of the final O-coalgebra.
- The property of element preservation follows from the fact that it is a slice morphism and the slice map of the source and target are both elts.

Thus we *uniquely* obtain the desired intrinsically verified sorting algorithm, including the guarantee that the output list is finite, while using the same framework for algorithm design as [7], bialgebraic semantics.

6 Related Work

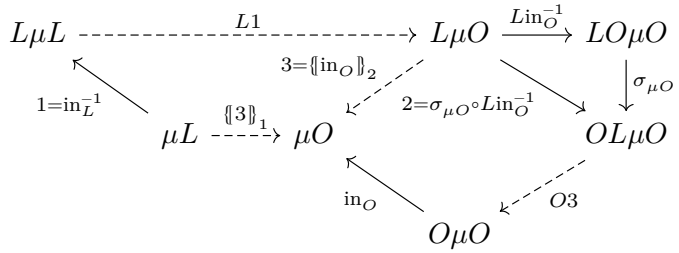
Our work essentially builds on top of [7] to construct an intrinsically verified sorting algorithm using categorical semantics of dependent types, while using the same framework, bialgebraic semantics, to define it. Their work seems to be the only one in the literature so far that applies bialgebraic semantics, which originated in the field of programming language semantics [25], to algorithm design.

The way orderedness is encoded locally in the base functor in [14, Section 4] bears similarity to our work (they make no reference to categorical semantics but we translate to our terminology here). However, their approach indexes lists by their heads, which suffices for orderedness, but, as we have seen, for element preservation one needs to index lists by the whole multiset of their elements. On the topic of element-preservation, [6] define multiset equivalence in Agda and as an example application extrinsically verify that a tree-sort preserves elements. As regards intrinsic verification, [12] define an intrinsically verified heapsort as a *metamorphism*, using a technique from [15]. Their algorithm has the shortcoming of having a coinductive type as codomain, i.e. the finiteness isn't captured in the type, a similar shortcoming to that which we have at least *theoretically* overcome in our work; however their work is actually formalized in Agda whereas ours isn't (at least yet: see Future Work).

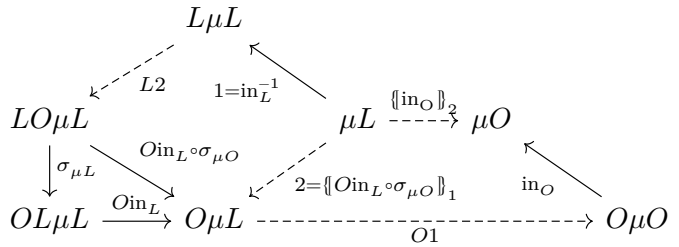
7 Future Work

In this work we managed to locally encode the properties arising from the specification of sorting, namely element-indexing and orderedness, in base functors for the input and output lists. We proved that there is a unique distributive law between them, the bialgebraic semantics of which gives us intrinsically correct sorting. A side effect of making the coalgebras for the output list base functor element-indexed was that they are necessarily well-founded. The main shortcoming of our work is that we haven't implemented our construction in a dependently typed language (e.g. Agda), thus leaving our claim that such a modular definition of an intrinsically correct algorithm reduces its verification burden untested. The main obstruction to a formalization is that we used finality to define morphisms into a final coalgebra. We proved that all such coalgebras are well-founded by using the elements index as a ranking function and constructed the unique map into the final coalgebra by splitting the domain into different sets based on the number of steps required until termination, but a direct translation of this into Agda seems to us unergonomic and unidiomatic for programming.

Ideally we would want to replace both algebra morphisms defined by initiality, as well as coalgebra morphisms by finality by coalgebra-to-algebra morphisms from *recursive coalgebras* [5]. To illustrate what we mean, we introduce the ad-hoc notation $\{\!|a|\!\}_c$ for the unique coalgebra-to-algebra morphism from a recursive F-coalgebra c to some target F-algebra a. The recursive-coalgebra version of the definition of the algorithm by initiality looks like this:



And the diagram for the definition by finality like this:



Since it is a known result that the inverse maps of initial algebras are recursive coalgebras [5], The proof obligations arising from the above diagrams would be that the two maps $\sigma_{\mu O} \circ L\mathrm{in}_O^{-1}$ and $\{\!|O\mathrm{in}_L \circ \sigma_{\mu O}|\!\}_{\mathrm{in}_L^{-1}}$ respectively are recursive coalgebras.

We conjecture that this is the case. It is worth investigating whether it would hold in the original Set-construction used in [7] or whether our encoding in $\mathsf{Set}/\mathcal{M}(A)$ is necessary. In this vein we would also like to further investigate our observation, made in Section 4.2, that L-algebras are equivalently $(1 + A \times -)$-algebra-morphisms to elt, since another consequence of it is that O-coalgebras are (omitting the orderedness constraint) $(1 + A \times -)$-coalgebra-to-algebra-morphisms to elt.

Should the conjecture hold, our next step would be to define such an intrinsically verified sorting algorithm in Agda. Prior work has been done on "good", i.e. ergonomic for proofs in practice, definitions of datatypes with orderedness-invariants [14], as well as on using proof-relevant membership relations on multisets for element preservation [6].

Should an adjusted "bialgebraic semantics" pattern for defining algorithms prove itself amenable to use in a dependently-typed programming language, we plan to investigate what other algorithms besides sorting could be expressed (and verified) in it.

8 Acknowledgments

I would like to thank my supervisor **Jurriaan Rot** for his great supervision. As regards content, he realized that the elements map for O-coalgebras could be used as a ranking function to prove that O-coalgebras are well-founded. He also had the idea that recursive coalgebras could be used to obtain a finite codomain. He outlined this idea in [23]; we recapitulate it in Future Work.

I would like to thank **Bálint Kocsis** in general for insight into categorical semantics of dependent types. In particular he explained to me in what way a pullback could be used in the elaboration of the definition sketch I postulate in Section 2.2. He also pointed me to the decomposition of the pullback into the composition of the existential after the base change functor, which I use in Section 4.1.2.

I would like to thank **Ruben Turkenburg** for advice on notation and terminology, as well as pointing out the fact that left adjoints preserve colimits, which I also use in Section 4.1.2.

Finally I would like to thank `varkor` for pointing out to me [26] that the definition sketch I postulate in Section 2.1 is satisfied by a known categorical construction, the induced monoidal structure of a category sliced over a monoid object, which I use in Section 4.1.

References

[1] Peter Aczel. "On Voevodsky's Univalence Axiom". In: *Mathematical Logic: Proof Theory, Constructive Mathematics* 8.4 (July 25, 2012), p. 2967. URL: `https://ems.press/journals/owr/articles/11450` (visited on 09/04/2023).

[2] Cass Alexandru. *mu-Bialgebras by cxandru · Pull Request #362 · agda/agda-categories*. GitHub. URL: `https://github.com/agda/agda-categories/pull/362` (visited on 09/20/2023).

[3] Falk Bartels. "On Generalised Coinduction and Probabilistic Specification Formats: Distributive laws in coalgebraic modelling". Issue: 6 Series: IPA Dissertation Series. PhD thesis. Vrije Universiteit Amsterdam, 2004. URL: https://ir.cwi.nl/pub/17104.

[4] Richard S. Bird and Oege de Moor. *Algebra of programming*. Prentice Hall International series in computer science. Prentice Hall, 1997.

[5] Venanzio Capretta, Tarmo Uustalu, and Varmo Vene. "Recursive coalgebras from comonads". In: *Inf. Comput.* 204.4 (2006), pp. 437–468.

[6] Nils Anders Danielsson. "Bag Equivalence via a Proof-Relevant Membership Relation". In: *Interactive Theorem Proving*. Lecture Notes in Computer Science. Springer, 2012, pp. 149–165.

[7] Ralf Hinze, Daniel W. H. James, Thomas Harper, Nicolas Wu, and José Pedro Magalhães. "Sorting with bialgebras and distributive laws". In: *Proceedings of the 8th ACM SIGPLAN workshop on Generic programming, WGP@ICFP 2012, Copenhagen, Denmark, September 9-15, 2012*. ACM, 2012, pp. 69–80.

[8] Jason Z. S. Hu and Jacques Carette. "Formalizing category theory in Agda". In: *CPP '21: 10th ACM SIGPLAN International Conference on Certified Programs and Proofs, Virtual Event, Denmark, January 17-19, 2021*. ACM, 2021, pp. 327–342.

[9] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Vol. 59. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016.

[10] P.T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium: Volume 1*. Oxford Logic Guides. Clarendon Press, 2002.

[11] Bartek Klin. "Bialgebras for structural operational semantics: An introduction". In: *Theor. Comput. Sci.* 412.38 (2011), pp. 5043–5069.

[12] Hsiang-Shang Ko. "Programming Metamorphic Algorithms: An Experiment in Type-Driven Algorithm Design". In: *Art Sci. Eng. Program.* 5.2 (2021), p. 7.

[13] Miran Lipovaca. "Recursion". In: *Learn You a Haskell for Great Good!* no starch press, 2011. URL: http://learnyouahaskell.com/recursion (visited on 09/11/2023).

[14] Conor Thomas McBride. "How to keep your neighbours in order". In: *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming*. ICFP'14: ACM SIGPLAN International Conference on Functional Programming. ACM, Aug. 19, 2014, pp. 297–309. URL: https://dl.acm.org/doi/10.1145/2628136.2628163 (visited on 09/20/2023).

[15] Keisuke Nakano. "Metamorphism in jigsaw". In: *J. Funct. Program.* 23.2 (2013), pp. 161–173.

[16] nLab authors. *adjoints preserve (co-)limits*. Sept. 2023. URL: https://ncatlab.org/nlab/revision/adjoints+preserve+%28co-%29limits/10.

[17] nLab authors. *base change*. Sept. 2023. URL: https://ncatlab.org/nlab/revision/base+change/39.

[18] nLab authors. *generalized element*. Sept. 2023. URL: https://ncatlab.org/nlab/revision/generalized+element/29.

[19] nLab authors. *monoidal topos*. Sept. 2023. URL: https://ncatlab.org/nlab/revision/monoidal+topos/8.

[20] nLab authors. *pullback*. Sept. 2023. URL: https://ncatlab.org/nlab/revision/pullback/50.

[21] Jaap van Oosten. *Basic Category Theory*. BRICS Lecture Series LS-95-01. BRICS, Computer Science Department, University of Aarhus, 1995. URL: http://www.staff.science.uu.nl/%E2%88%BCooste110/www/syllabi/catsmoeder.pdf.

[22] *Haskell 98 Language and Libraries – The Revised Report*. Cambridge University Press, 2003.

[23] Jurriaan Rot. "Distributive laws and recursive coalgebras". Aug. 26, 2019. URL: jurriaan.creativecode.org/wp-content/uploads/2023/09/corecursive_algebras_note.pdf (visited on 09/21/2023).

[24] Jan J. M. M. Rutten. "Universal coalgebra: a theory of systems". In: *Theor. Comput. Sci.* 249.1 (2000), pp. 3–80.

[25] Daniele Turi and Gordon D. Plotkin. "Towards a Mathematical Operational Semantics". In: *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*. IEEE Computer Society, 1997, pp. 280–291.

[26] varkor. *Answer to "Is this kind of functor Set/M×Set/M → Set/M, with M a monoid, a known construction?"* MathOverflow. June 23, 2023. URL: https://mathoverflow.net/a/449441/502814 (visited on 08/14/2023).