Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information and Computation 204 (2006) 437–468

Information and Computation

www.elsevier.com/locate/ic

Recursive coalgebras from comonads[☆],[☆☆]

Venanzio Capretta [a,1], Tarmo Uustalu [b,*,2], Varmo Vene [c,2]

[a] *Department of Mathematics and Statistics, University of Ottawa, 585 King Edward Ave., Ottawa, Ont., Canada K1N 6N5*

[b] *Institute of Cybernetics at Tallinn University of Technology, Akadeemia tee 21, EE-12618 Tallinn, Estonia*

[c] *Department of Computer Science, University of Tartu, J. Liivi 2, EE-50409 Tartu, Estonia*

Received 3 August 2004; revised 29 June 2005
Available online 9 March 2006

Abstract

The concept of recursive coalgebra of a functor was introduced in the 1970s by Osius in his work on categorical set theory to discuss the relationship between wellfounded induction and recursively specified functions. In this paper, we motivate the use of recursive coalgebras as a paradigm of structured recursion in programming semantics, list some basic facts about recursive coalgebras and, centrally, give new conditions for the recursiveness of a coalgebra based on comonads, comonad-coalgebras and distributive laws of functors over comonads. We also present an alternative construction using countable products instead of cofree comonads.

Keywords: Structured recursion; Recursive coalgebras; Wellfounded coalgebras; Comonads; Distributive laws

[☆] Research supported by the Estonian-French cooperation programme Parrot.

[☆☆] This paper is a significantly revised and expanded version of a conference paper that appeared in Proc. of CMCS'04, Electron. Notes in Theoret. Comput. Sci. 106 (2004) 43–61.

[*] Corresponding author.

E-mail addresses: venanzio@site.uottawa.ca (V. Capretta), tarmo@cs.ioc.ee (T. Uustalu), varmo@cs.ut.ee (V. Vene).

[1] Previously at INRIA Sophia Antipolis, 2004 route des Lucioles, BP 93, F-06902 Sophia Antipolis, France. Research activity partially supported by Marie Curie Individual Fellowship HPMF-CT-2001-01369.

[2] Research activity partially supported by the Estonian Science Foundation Grant No. 5567 and by EU FP5 project IST-2001-37592 (eVikings II).

0890-5401/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2005.08.005

1. Introduction

This paper is dedicated to the study of recursive functor-coalgebras. In the sense of Osius [30], a coalgebra (A, α) of a functor $F : \mathcal{C} \to \mathcal{C}$ is recursive iff, for any algebra (C, φ) of F, the morphism equation

$$f = \varphi \circ Ff \circ \alpha \tag{*}$$

has a unique solution in the unknown $f : A \to C$.

Our prime interest in recursive coalgebras comes from their application to programming semantics. Specifically, we see how Eq. (*) can be seen as the most abstract form of definition of a recursive function. In programming (we are chiefly thinking of typed functional programming), it is customary to wish to be able to take some function $\Phi : \mathcal{C}(A, C) \to \mathcal{C}(A, C)$ and read the equation

$$f = \Phi(f) \tag{**}$$

as a function definition. The problem is that, for an arbitrary Φ, the equation (**) is not guaranteed to make sense as a definition: it may have exactly one solution, but it can just as well have no solution or multiple solutions among which there is no canonical choice. But for more specific choices of Φ, the equation may indeed be predestined to have exactly one solution (or several solutions, but among them a canonical one) and in this case it is really meaningful to see it as a definition.

Eq. (*) is a structured instance of (**), that is, a pattern to define specific operators Φ by specifying components F, α, and φ. One of the ways to know that Φ properly defines a function is to know that (A, α) is recursive. The equation form (*) covers most useful situations in programming and examples of recursive coalgebras abound. To mention some:

(1) For any functor $F : \mathcal{C} \to \mathcal{C}$ with an initial algebra, $(\mu F, \mathsf{in}_F)$, the F-coalgebra $(\mu F, \mathsf{in}_F^{-1})$ is recursive (iteration). But so are also the $F(\mathsf{Id} \times K_{\mu F})$-algebra $(\mu F, F\langle\, \mathsf{id}_{\mu F}, \mathsf{id}_{\mu F} \,\rangle \circ \mathsf{in}_F^{-1})$ (primitive recursion), the $F(\mathsf{Id} \times F)$-coalgebra $(\mu F, F\langle\, \mathsf{id}_{\mu F}, \mathsf{in}_F^{-1} \,\rangle \circ \mathsf{in}_F^{-1})$ (iteration back one or two steps), etc. Recursive coalgebras cover a wide variety of structured recursion schemes for initial algebras.

(2) The set $\mathsf{List}Z$ of all lists over some linearly ordered set Z, together with the nil and cons functions, is the initial algebra of the functor $\mathsf{L}_Z = K_1 + K_Z \times \mathsf{Id} : \mathbf{Set} \to \mathbf{Set}$. Endowed with the analysis of every non-empty list into its head and tail, the set $\mathsf{List}Z$ is a recursive L_Z-coalgebra (a fact which justifies insertion sort) and so is every suffix-closed subset of $\mathsf{List}Z$. A recursive L_Z-coalgebra is also given by the set $\mathsf{List}Z$ equipped with the analysis of every non-empty list into its smallest element and the rest (this enables selection sort). Equipped with the analysis of every non-empty, non-singleton list into two halves, the set $\mathsf{List}Z$ is a recursive coalgebra of the functor $\mathsf{BLT}_Z = K_1 + K_Z + \mathsf{Id} \times \mathsf{Id}$ (the foundation of mergesort) whereas equipped with the analysis of every non-empty list into an element and the lists of smaller and larger elements, it becomes a recursive coalgebra of the functor $\mathsf{BT}_Z = K_1 + K_Z \times \mathsf{Id} \times \mathsf{Id}$ (used in quicksort), etc.

(3) A functor may well have recursive coalgebras without having an initial algebra. For example, a set with a relation on it carries a recursive coalgebra of the powerset functor iff the relation is wellfounded.

The examples above show it convincingly that the notion of recursive coalgebra "generalizes" initial algebras insofar as structured recursion is concerned. We believe that recursive coalgebras are a more useful tool in the study of structured recursion than initiality and that most results for structured recursion and initial algebras can be recast in a clearer way in this more general framework. In particular, the recursive coalgebras approach makes it plain that structured recursion is not necessarily tied to initial algebras, essentially the same phenomenon can occur also in situations when there is none around.

Contribution of the paper. In this paper, we motivate the use of recursive coalgebras as a paradigm of structured recursion in programming semantics, list some basic facts about recursive coalgebras and, centrally, give new conditions for the recursiveness of a coalgebra based on comonads, comonad-coalgebras and distributive laws of functors over comonads. We also present an alternative construction using countable products instead of cofree comonads. As a larger application example, we discuss a short proof of the dual of the Solution Theorem of Moss [28] and Aczel et al. [2].

Technically, the central results are a generalization of our results [36,37,39] on advanced structured recursion schemes for initial algebras and, modulo the duality, the dual results [5,8] on advanced structured corecursion schemes for final coalgebras. However, the important point with the generalization is not as much technical as it is conceptual. As we already stated, we believe that, as long as structured recursion is concerned, recursive coalgebras are a more basic concept than initial algebras, and developing an account of structured recursion proceeding from this basis yields a considerably more modular result. Accordingly, the main goal of the paper is to promote and defend this position. We believe that this is especially relevant for type-theoretically motivated functional languages without uniform fixed-point operators for all types, such as Cockett's Charity [9], Turner's total functional programming [35] or dependently typed languages like Epigram [25].

Some disclaimers are in order. First, we have consciously avoided trying to treat all aspects of recursive coalgebras in this paper, although we plan to study several of them in our future work (see Section 7). In particular, we do not discuss in any detail the relationship of recursiveness to wellfoundedness in the manner of Taylor [31–33] or to the condition of Adámek et al. [1] of existence of a coalgebra-to-algebra morphism to the initial algebra (see the paragraph Related work). For "nice" (**Set**-like) categories and "nice" functors (preserving monos and inverse image diagrams), recursiveness and wellfoundedness turn out to be equivalent, which yields a rich choice of tools to prove recursiveness. In this paper, we have deliberately chosen to focus on facts which do not depend on such assumptions of nicety, even if, in practical applications, they would most often be non-restrictive. We find it remarkable how much can be proved at this level of generality where recursiveness and wellfoundedness are not the same.

Second, programming semantics is interested in both recursive specification of functions and recursive specification of types. In this text, all our interest is in recursive specification of functions. Recursive types are in the picture only to the degree that structured recursion has to do with initial algebras.

Related work. Recursive coalgebras (also known as algebra-initial coalgebras), together with wellfounded coalgebras – a related concept where, instead of a recursion principle, the coalgebra has to obey an induction principle – were first introduced by Osius [30] in his work on categorical set theory. He considered wellfounded and recursive coalgebras of the powerset functor of the category of sets (or, more abstractly, of the powerobject functor of an elementary topos), and proved the general recursion theorem, that every wellfounded coalgebra of the powerset functor is recursive.

Taylor [31–33] took Osius's ideas further, showing that the general recursion theorem holds for any functor on **Set** preserving monos and inverse image diagrams. In a very new work [1], Adámek et al. showed that a coalgebra of a finitary **Set** functor preserving monos and inverse image diagrams is recursive iff it has a coalgebra-to-algebra morphism to the initial algebra.

Eppendahl [11,12] studied recursive coalgebras with the objective of obtaining an explanation to Freyd's [14,16,17] transposition of invariant objects.

The dual concept of a corecursive (also known as coalgebra-final, iterative) algebra was used by Escardó and Simpson [13] to provide a universal characterization of the closed euclidean interval. The newest work by Adámek and co-workers [27,4] on the free completely iterative monad (resp. the free iterative monad) is centered around a related, but stronger concept (resp. its finitary version considered also earlier by Nelson [29]).

Advanced structured recursion schemes for initial algebras (schemes reaching beyond the usual iteration and primitive recursion) have been studied by us and Pardo [36,37,39] and the dual schemes for final coalgebras by Bartels [5] and Cancila et al. [8].

In typed lambda calculi, initial algebras and final coalgebras model inductive and coinductive types. Inductive and coinductive types with (co)iteration and/or primitive (co)recursion are fairly standard constructions in such languages at least since the work of Hagino [20]. This is fine metatheoretically, but from a programming point of view, one would rather prefer direct support for less rigid primitives. On a more advanced level, Giménez [18,19] invented guarded-by-destructors recursion and guarded-by-constructors corecursion (in two versions – based on syntactic side-conditions and type-based), which are powerful structured (co)recursion schemes for inductive and coinductive types. Structured recursion from more general perspectives has been studied by, e.g., Bove and Capretta [6,7] and McBride and McKinna [24].

To functional programming, initial algebras and final coalgebras were first introduced by Malcolm [23]. The diagram of general structured recursion scheme was first introduced by Meijer et al. [26] who called it the hylo diagram. Usually, functional programmers assume settings where the diagram has always a canonical solution, no matter what the coalgebra and algebra are. But Doornbos and Backhouse [10] have asked the question under what conditions the hylo diagram has a unique solution.

Organization of the paper. In Section 2, we explain our motivation for studying recursive coalgebras and give the definition. In Section 3, we present a number of important basic facts about recursive coalgebras. In Section 4, which is the main section of the paper, we show how recursive coalgebras arise from comonads, comonad-coalgebras and distributive laws of functors over comonads. In 5, building on ideas from Bartels [5], we develop an alternative construction which uses countable products instead of cofree comonads. In Section 6, we discuss an extended example: the dual of the Solution Theorem of Moss [28] and Aczel et al. [2]. In Section 7, we conclude by pointing out some directions for future research.

2. Recursive coalgebras: motivation and definition

In functional programming, functions are commonly specified by recursive equations. Often, these equations have a nice and simple structure, although this structure may be hidden. As an example, consider a possible definition of the quicksort algorithm. Let Z be a set linearly ordered by $\leqslant$ and let $\mathsf{List}Z$ be the initial algebra of the functor L_Z defined by $\mathsf{L}_Z X = 1 + Z \times X$. Quicksort is defined as follows.

$$\begin{aligned}
&\mathsf{qsort}\colon \mathsf{List}Z \to \mathsf{List}Z\\
&\mathsf{qsort}\ [] = []\\
&\mathsf{qsort}\ (x : l) = \mathsf{qsort}(l_{\leqslant x}) +\!\!+ (x : \mathsf{qsort}(l_{>x})),
\end{aligned}$$

where $l_{\leqslant x} = [y \leftarrow l \mid y \leqslant x]$ and $l_{>x} = [y \leftarrow l \mid y > x]$.

This definition is clearly based on an equation of the form $qsort = \Phi(qsort)$ where $\Phi : \mathbf{Set}(\mathsf{List}Z, \mathsf{List}Z) \to \mathbf{Set}(\mathsf{List}Z, \mathsf{List}Z)$. With minimal effort, we can see that $\Phi(qsort)$ may be rewritten into an equivalent form $\mathsf{qconcat} \circ \mathsf{BT}_Z\, qsort \circ \mathsf{qpartition}$ where $\mathsf{BT}_Z\ X = 1 + Z \times X \times X$. The first morphism $\mathsf{qpartition}$ of the composition determines the arguments for the recursive calls; $(\mathsf{List}Z, \mathsf{qpartition})$ is a BT_Z-coalgebra:

$$\begin{aligned}
&\mathsf{qpartition}\colon \mathsf{List}Z \to 1 + Z \times \mathsf{List}Z \times \mathsf{List}Z\\
&\mathsf{qpartition}\ [] = \mathsf{inl}(*)\\
&\mathsf{qpartition}\ (x : l) = \mathsf{inr}(\langle x, l_{\leqslant x}, l_{>x}\rangle).
\end{aligned}$$

The second morphism $\mathsf{BT}_Z qsort\colon \mathsf{BT}_Z(\mathsf{List}Z) \to \mathsf{BT}_Z(\mathsf{List}Z)$ makes the recursive calls. The third morphism $\mathsf{qconcat}$ determines how the results of the recursive calls combine into the result of the main call; $(\mathsf{List}Z, \mathsf{qconcat})$ is a BT_Z-algebra:

$$\begin{aligned}
&\mathsf{qconcat}\colon 1 + Z \times \mathsf{List}Z \times \mathsf{List}Z \to \mathsf{List}Z\\
&\mathsf{qconcat}\ \mathsf{inl}(*) = []\\
&\mathsf{qconcat}\ \mathsf{inr}(\langle x, l_1, l_2\rangle) = l_1 +\!\!+ (x : l_2).
\end{aligned}$$

The definition of qsort is a description of qsort as the solution of the equation $qsort = \mathsf{qconcat} \circ \mathsf{BT}_Z qsort \circ \mathsf{qpartition}$ (here $qsort$ is the unknown of an equation while qsort is the solution), graphically represented by the diagram

$$\begin{tikzcd}
\mathsf{BT}_Z\,\mathsf{List}Z \arrow[d, "\mathsf{BT}_Z\, qsort"'] & \mathsf{List}Z \arrow[l, "\mathsf{qpartition}"'] \arrow[d, "qsort"] \\
\mathsf{BT}_Z\,\mathsf{List}Z \arrow[r, "\mathsf{qconcat}"'] & C
\end{tikzcd}$$

This description makes sense because the equation happens to have a unique solution. This follows from the arguments of the recursive calls being always strictly shorter than that of the main call – a property of the coalgebra $(\mathsf{List}Z, \mathsf{qpartition})$. The equation remains uniquely solvable also, if we replace $(\mathsf{List}Z, \mathsf{qconcat})$ with any other BT_Z-algebra (C,φ): we may say that $(\mathsf{List}Z, \mathsf{qpartition})$ is recursive.

An important feature here is that, to prepare the arguments for the recursive calls, we use a different structure than the initial L_Z-algebra structure of $\mathsf{List}Z$, which is basically its definition. Instead, $(\mathsf{List}Z, \mathsf{qpartition})$ is a recursive coalgebra of a different functor BT_Z, whose initial algebra is the type of binary trees with nodes labelled by elements of Z. So, the recursiveness of the coalgebra does not follow directly from ListZ being an initial algebra, but must be proved by other means. One of the main goals of this paper is to present results that allow one to deduce the recursiveness of a coalgebra from the recursiveness of simpler ones.

Abstracting away from the concrete data of the above example, we are led to the following definition ($\mathcal{C}$ denotes any category).

Definition 1 (*Coalgebra-to-algebra morphism, recursive coalgebra*)**.** Let $F : \mathcal{C} \to \mathcal{C}$ be a functor. A *coalgebra-to-algebra morphism* from an F-coalgebra (A,α) to an F-algebra (C,φ) is a morphism $f : A \to C$ such that

$$\begin{tikzcd} FA \arrow[d, "Ff"'] & A \arrow[l, "\alpha"'] \arrow[d, "f"] \\ FC \arrow[r, "\varphi"] & C \end{tikzcd}$$

commutes.

An F-coalgebra (A,α) is *recursive* iff for every F-algebra (C,φ) there exists a unique coalgebra-to-algebra morphism from (A,α) to (C,φ), denoted $\mathsf{fix}_{F,\alpha}(\varphi)$.

So, in the quicksort example, $\mathsf{qsort} = \mathsf{fix}_{\mathsf{BT}_Z,\mathsf{qpartition}}(\mathsf{qconcat})$. The existence and uniqueness of the solution depends only on the recursiveness of the coalgebra $(\mathsf{List}Z, \mathsf{qpartition})$; so we can choose any other BT_Z algebra and we still get a unique solution. For example, let $\mathsf{Tree}_Z = \mu\mathsf{BT}_Z$ be the set of binary trees with nodes labelled by elements of Z; $(\mathsf{Tree}_Z, \mathsf{in}_{\mathsf{BT}_Z})$ is a BT_Z-algebra. The program $\mathsf{bst} = \mathsf{fix}_{\mathsf{BT}_Z,\mathsf{qpartition}}(\mathsf{in}_{\mathsf{BT}_Z})$ constructs a binary search tree associated with the input list.

Recursive coalgebras and (ordinary) coalgebra morphisms form a category $\mathbf{RecCoalg}_F$ which is trivially a full subcategory of $\mathbf{Coalg}_F$.

We note that, in the functional programming community, the coalgebra-to-algebra morphism condition is known as the *hylo diagram* [26]. The recursion scheme used – the *hylo scheme* – says that, if F has an initial algebra $(\mu F, \mathsf{in}_F)$ whose inverse is its final coalgebra (which is always true, if $\mathcal{C}$ is algebraically compact, as in the case where $\mathcal{C}$ gives a semantics, for lazy functional programming), then the postcomposition of the initial algebra morphism to a given algebra (C,φ) with the final coalgebra morphism from a given coalgebra (A,α), i.e., the morphism $\mathsf{It}_F(\varphi) \circ \mathsf{Coit}_F(\alpha) : A \to C$ (the hylomorphism), is a coalgebra-to-algebra morphism from (A,α) to (C,φ):

$$\begin{tikzcd}
FA \arrow[d, "F\mathsf{Coit}_F(\alpha)"'] & A \arrow[l, "\alpha"'] \arrow[d, "\mathsf{Coit}_F(\alpha)"] \\
F\mu F \arrow[d, "F\mathsf{It}_F(\varphi)"'] \arrow[r, shift right, "\mathsf{in}_F"'] & \mu F \arrow[l, shift right, "\mathsf{in}_F^{-1}"'] \arrow[d, "\mathsf{It}_F(\varphi)"] \\
FC \arrow[r, "\varphi"'] & C
\end{tikzcd}$$

The hylomorphism is not necessarily a unique solution of the hylo diagram, just a canonical one. In this paper, our interest is not in such situations, but rather in cases where some fixed coalgebra guarantees that there is a unique solution for any algebra.

For the powerset functor $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$, the notion of recursive coalgebra coincides with that of wellfounded relation. Indeed, any $\mathcal{P}$-coalgebra $\alpha : A \to \mathcal{P}A$ determines and is determined by a relation $\prec$ on A (we use the symbol $\prec$ to help intuition, but the relation need not be an order): $\alpha(a) = \{x \in A \mid x \prec a\}, x \prec a$ iff $x \in \alpha(a)$. A $\mathcal{P}$-coalgebra-to-algebra morphism from (A,α) to (C,φ) is a function $f : A \to C$ such that $f = \varphi \circ \mathcal{P}f \circ \alpha$. If $a \in A$, then $(\mathcal{P}f \circ \alpha)(a) = \{f(x) \mid x \prec a\}$, so the condition says that

$$f(a) = \varphi(\{f(x) \mid x \prec a\}).$$

We get that (A,α) is recursive iff, for any set C and function $\varphi : \mathcal{P}C \to C$, the equation above has a unique solution in $f : A \to C$. This happens exactly when the relation $\prec$ is wellfounded.

The wellfounded relation example is the main motivating example in [30,32,33] and is very beautiful. But it is important to realize that the programming uses of wellfounded recursion in its pure form are few. It is very uncommon in a function definition that one reduces a main call argument value to an unstructured set of recursive call argument values and assembles the main call result value from an unstructured set of recursive call result values.

3. Recursive coalgebras: basic constructions

The example of the powerset functor shows that it can be hard to determine whether a coalgebra of a given functor F is recursive: determining the wellfoundedness of a decidable relation on natural numbers is undecidable. So, instead of trying to solve the unsolvable, we will point out a few simple cases where some coalgebra is obviously recursive and then provide various constructions for producing new recursive coalgebras out of coalgebras already known to be recursive.

3.1. First observations

We start with the simplest interesting case when the functor F has an initial algebra. In this situation, we agree to write $(\mu F, \mathsf{in}_F)$ for the initial F-algebra and $\mathsf{It}_F(\varphi)$ for the unique algebra morphism from $(\mu F, \mathsf{in}_F)$ to a given F-algebra (C, φ) (the iteration given by (C, φ)). By Lambek's lemma [22], in_F is an isomorphism. We denote its inverse by in_F^{-1}.

Proposition 2. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor. If F has an initial algebra, then $(\mu F, \mathsf{in}_F^{-1})$ is a final recursive F-coalgebra.*

Proof. The F-coalgebra $(\mu F, \mathsf{in}_F^{-1})$ is certainly recursive, since the unique algebra morphism $\mathsf{It}_F(\,\varphi\,)$ from $(\mu F, \mathsf{in}_F)$ to an F-algebra (C, φ) is also a unique coalgebra-to-algebra morphism from $(\mu F, \mathsf{in}_F^{-1})$ to (C, φ).

To see that $(\mu F, \mathsf{in}_F^{-1})$ is final among the recursive F-coalgebras, let (A, α) be a recursive F-coalgebra. A coalgebra-to-algebra morphism from (A, α) to $(\mu F, \mathsf{in}_F)$ is the same thing as a coalgebra morphism from (A, α) to $(\mu F, \mathsf{in}_F^{-1})$. By recursiveness of (A, α) there exists a unique such coalgebra-to-algebra morphism, therefore a unique coalgebra morphism. □

Corollary 3. *If F has an initial algebra, then the unique coalgebra-to-algebra morphism from a recursive F-coalgebra (A, α) to an F-algebra (C, φ) factors as follows:*

$$\mathsf{fix}_{F,\alpha}(\varphi) = \mathsf{It}_F(\,\varphi\,) \circ \mathsf{fix}_{F,\alpha}(\mathsf{in}_F).$$

For example, the quicksort algorithm can be factored by first constructing a binary search tree: $\mathsf{qsort} = \mathsf{It}_{BT_Z}(\,\mathsf{qconcat}\,) \circ \mathsf{bst}$ where $\mathsf{bst} = \mathsf{fix}_{\mathsf{BT}_Z,\mathsf{qpartition}}(\mathsf{in}_{\mathsf{BT}_Z})$.

Proposition 4. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor and (A, α) a recursive F-coalgebra. If F has an initial algebra, then $m = \mathsf{fix}_{F,\alpha}(\mathsf{in}_F) : A \to \mu F$ is split mono (as a morphism, not necessarily as a coalgebra morphism) iff α is split mono.*

Proof. (if) Let the postinverse of $\alpha : A \to FA$ be $\alpha^- : FA \to A$. Then $m^- = \mathsf{It}_F(\,\alpha^-\,) : \mu F \to A$ is a postinverse of $m : A \to \mu F$:

$$\begin{tikzcd}
FA \arrow[d, "Fm"] \arrow[dd, bend right=60, "F\mathsf{id}_A"'] & A \arrow[l, "\alpha"'] \arrow[d, "m"'] \arrow[dd, bend left=60, "\mathsf{id}_A"] \\
F\mu F \arrow[r, "\mathsf{in}_F"] \arrow[d, "Fm^-"] & \mu F \arrow[d, "m^-"'] \\
FA \arrow[r, "\alpha^-"'] & A
\end{tikzcd}$$

Indeed, we have $m^- \circ m = m^- \circ \mathsf{in}_F \circ Fm \circ \alpha = \alpha^- \circ F(m^- \circ m) \circ \alpha$, but we also have $\mathsf{id}_A = \alpha^- \circ F\mathsf{id}_A \circ \alpha$, hence $m^- \circ m = \mathsf{fix}_{F,\alpha}(\alpha^-) = \mathsf{id}_A$.

(only if) Write $m^- : \mu F \to A$ for the postinverse of $m : A \to \mu F$. Then $\alpha^- = m^- \circ \mathsf{in}_F \circ Fm : FA \to A$ is a postinverse of $\alpha : A \to FA$, since $\alpha^- \circ \alpha = m^- \circ \mathsf{in}_F \circ Fm \circ \alpha = m^- \circ m = \mathsf{id}_A$. □

Next we discuss reductions of the question of recursiveness of one coalgebra to that of some other, related coalgebra.

Here is the first proposition describing a reduction of this sort.

Proposition 5. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A, α) a recursive F-coalgebra and (B, β) an F-coalgebra. If there are F-coalgebra morphisms $h : (A, \alpha) \to (B, \beta)$ and $k : (B, \beta) \to (FA, F\alpha)$ such that $\beta = Fh \circ k$, then (B, β) is also recursive.*

Proof. Consider an arbitrary F-algebra (C, φ). Let $g = \mathsf{fix}_{F,\alpha}(\varphi)$. The situation is summarized in the following diagram.

$$\begin{tikzcd}
 & FB \arrow[dl, bend right=60, "Fk"'] & B \arrow[l, "\beta"'] \arrow[dl, bend right=60, "k"'] \\
F^2A \arrow[d, "F^2g"'] & FA \arrow[l, "F\alpha"'] \arrow[u, "Fh"'] \arrow[d, "Fg"] & A \arrow[l, "\alpha"'] \arrow[u, "h"'] \arrow[d, "g"] \\
F^2C \arrow[r, "F\varphi"] & FC \arrow[r, "\varphi"] & C
\end{tikzcd}$$

Let $f = \varphi \circ Fg \circ k : B \to C$. We show that $\mathsf{fix}_{F,\beta}(\varphi) = f$. We have:

$$\begin{aligned} f &= \varphi \circ Fg \circ k = \varphi \circ F(\varphi \circ Fg \circ \alpha) \circ k \\ &= \varphi \circ F(\varphi \circ Fg \circ k) \circ \beta = \varphi \circ Ff \circ \beta. \end{aligned}$$

Hence f is a F-coalgebra-to-algebra morphism from (B, β) to (C, φ).

To see that f is unique, suppose that f' is another F-coalgebra-to-algebra morphism from (B, β) to (C, φ). Then we have:

$$f' \circ h = \varphi \circ Ff' \circ \beta \circ h = \varphi \circ F(f' \circ h) \circ \alpha.$$

This implies, by the unicity of $\mathsf{fix}_{F,\alpha}(\varphi)$, that $f' \circ h = \mathsf{fix}_{F,\alpha}(\varphi) = g$. Consequently, $f' = \varphi \circ Ff' \circ \beta = \varphi \circ F(f' \circ h) \circ k = \varphi \circ Fg \circ k = f$. □

A number of useful propositions follow from Proposition 5. First, recursive F-coalgebras are preserved by F.

Proposition 6. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor. If (A, α) is a recursive F-coalgebra, then $(FA, F\alpha)$ is also a recursive F-coalgebra.*

Proof. From Proposition 5 for $h = \alpha$ and $k = \mathsf{id}_{FA}$. □

The implication of Proposition 2 can be turned around.

Proposition 7. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor.*

(a) *If (A, α) is a recursive F-coalgebra and α is iso, then (A, α^{-1}) is an initial F-algebra.*

(b) *If (A, α) is a final recursive F-coalgebra, then α is iso both as a morphism and as a coalgebra morphism (and hence (A, α^{-1}) is an initial F-algebra).*

Proof. (a) The unique coalgebra-to-algebra morphism from (A, α) to an F-algebra (C, φ) is also a unique algebra morphism from (A, α^{-1}) to (C, φ): $\mathsf{It}_F(\varphi) = \mathsf{fix}_{F,\alpha}(\varphi)$.

(b) By Proposition 6, we have that $(FA, F\alpha)$ is a recursive F-coalgebra and it is trivial that α is a coalgebra morphism from (A, α) to $(FA, F\alpha)$. On the other hand, as (A, α) is a final recursive coalgebra, there exists a coalgebra morphism h from $(FA, F\alpha)$ to (A, α); i.e., we have the following situation:

$$\begin{tikzcd}
FA \arrow[d, "F\alpha"'] & A \arrow[l, "\alpha"'] \arrow[d, "\alpha"] \\
F^2A \arrow[d, "Fh"'] & FA \arrow[l, "F\alpha"'] \arrow[d, "h"] \\
FA & A \arrow[l, "\alpha"']
\end{tikzcd}$$

Now, as (A, α) is a final recursive coalgebra, there cannot be two distinct coalgebra morphisms from (A, α) to (A, α), hence $h \circ \alpha = \mathsf{id}_A$. From h being a coalgebra morphism, we further get also that $\alpha \circ h = F(h \circ \alpha) = \mathsf{id}_{FA}$. □

It is not true in general that a subcoalgebra of a recursive coalgebra is recursive. But the following weaker statement, holding just for *split* sub-coalgebras, is always true.

Proposition 8. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, let (A, α), (B, β) be F-coalgebras and $m : B \to A$ a split monic coalgebra morphism from (B, β) to (A, α).* (a) *If (A, α) is recursive, then (B, β) is also recursive.* (b) *If α is split mono, then so is β.*

Proof. Let $h : (A, \alpha) \to (B, \beta)$ be the postinverse of m: $h \circ m = \mathsf{id}_B$.

(a) We apply Proposition 5 with $k = \alpha \circ m : B \to FA$. We have to prove that k is a coalgebra morphism and that $\beta = Fh \circ k$. From k being a coalgebra morphism we have that $F\alpha \circ k = F\alpha \circ \alpha \circ m = F(\alpha \circ m) \circ \beta = Fk \circ \beta$. Furthermore, $\beta = \beta \circ h \circ m = Fh \circ \alpha \circ m = Fh \circ k$. By Proposition 5, (B, β) is recursive.

(b) Let α^- be the postinverse of α. Then $\beta^- = h \circ \alpha^- \circ Fm$ is a postinverse of β, since $\beta^- \circ \beta = h \circ \alpha^- \circ Fm \circ \beta = h \circ \alpha^- \circ \alpha \circ m = h \circ m = \mathsf{id}_B$. □

A formalization of recursion back one or two steps is given by the following proposition, with a relatively involved proof. But we will see in the following example that the seemingly meaningless diagram-chasing proof is in fact a justification of an optimizing program transformation. In the next section, we shall see that, under an extra assumption, this proposition is an instance of a more general theorem.

Proposition 9. *Let $\mathcal{C}$ be cartesian and $F : \mathcal{C} \to \mathcal{C}$ a functor. If (A, α) is a recursive F-coalgebra, then $(A, F\langle \mathsf{id}_A, \alpha \rangle \circ \alpha)$ is a recursive $F(\mathsf{Id} \times F)$-coalgebra.*

Proof. Consider an arbitrary $F(\mathsf{Id} \times F)$-algebra (C, φ). Let $\psi = \langle \varphi, F\mathsf{fst}_{C,FC} \rangle : F(C \times FC) \to C \times FC$, $g = \mathsf{fix}_{F,\alpha}(\psi) : A \to C \times FC$ and $f = \mathsf{fst}_{C,FC} \circ g : A \to C$. We show that $\mathsf{fix}_{F(\mathsf{Id} \times F), F\langle \mathsf{id}_A, \alpha \rangle \circ \alpha}(\varphi) = f$.

That f is a $F(\mathsf{Id} \times F)$-coalgebra-to-algebra morphism from $(A, F\langle \mathsf{id}_A, \alpha \rangle \circ \alpha)$ to (C, φ) is evident from the commutativity of the outer square in the diagram

$$\begin{tikzcd}
F(A \times FA) \arrow[dd, "F(g\times Fg)"] \arrow[ddd, bend right=60, "F(f\times Ff)"'] & & FA \arrow[ll, "F\langle \mathsf{id},\alpha \rangle"'] \arrow[dl, "F\langle \mathsf{id},\mathsf{id} \rangle"'] \arrow[dd, "Fg"] & A \arrow[l, "\alpha"'] \arrow[dd, "g"] \arrow[ddd, bend left=60, "f"] \\
 & F(A \times A) \arrow[ul, "F(\mathsf{id}\times\alpha)"] \arrow[d, "F(g\times g)"] & & \\
F((C \times FC) \times F(C \times FC)) \arrow[r, "F(\mathsf{id}\times\psi)"] \arrow[d, "F(\mathsf{fst}\times F\mathsf{fst})"] & F((C \times FC) \times (C \times FC)) \arrow[d, "F(\mathsf{fst}\times\mathsf{snd})"] & F(C \times FC) \arrow[l, "F\langle \mathsf{id},\mathsf{id} \rangle"'] \arrow[r, "\psi"] \arrow[d, equal] & C \times FC \arrow[d, "\mathsf{fst}"] \\
F(C \times FC) \arrow[r, equal] & F(C \times FC) \arrow[r, equal] & F(C \times FC) \arrow[r, "\varphi"] & C
\end{tikzcd}$$

To verify that f is unique, suppose that f' is another $F(\mathsf{Id} \times F)$-coalgebra-to-algebra morphism from $(A, F\langle \mathsf{id}_A, \alpha \rangle \circ \alpha)$ to (C, φ); so that $f' = \varphi \circ F(f' \times Ff') \circ F\langle \mathsf{id}, \alpha \rangle \circ \alpha = \varphi \circ F\langle f', Ff' \circ \alpha \rangle \circ \alpha$. We observe that

$$\begin{aligned}\langle f', Ff' \circ \alpha \rangle &= \langle \varphi \circ F\langle f', Ff' \circ \alpha \rangle \circ \alpha, F(\mathsf{fst}_{C,FC} \circ \langle f', Ff' \circ \alpha \rangle) \circ \alpha \rangle \\ &= \langle \varphi, F\mathsf{fst}_{C,FC} \rangle \circ F\langle f', Ff' \circ \alpha \rangle \circ \alpha = \psi \circ F\langle f', Ff' \circ \alpha \rangle \circ \alpha.\end{aligned}$$

This tells us that $\langle f', Ff' \circ \alpha \rangle = \mathsf{fix}_{F,\alpha}(\psi) = g$. As a consequence, $f' = \mathsf{fst}_{C,FC} \circ \langle f', Ff' \circ \alpha \rangle = \mathsf{fst}_{C,FC} \circ g = f$. □

As an application, we show how to construct the Fibonacci function. Its standard definition is the following:

$$\begin{aligned}&\mathsf{fib} : \mathbb{N} \to \mathbb{N} \\ &\mathsf{fib}\, 0 = 0 \\ &\mathsf{fib}\, 1 = 1 \\ &\mathsf{fib}\ (x+2) = \mathsf{fib}(x+1) + \mathsf{fib}(x).\end{aligned}$$

We apply the proposition to the recursive $(K_1 + \mathsf{Id})$-coalgebra $(\mathbb{N}, \mathsf{pred}) = (\mu(K_1 + \mathsf{Id}), \mathsf{in}^{-1}_{K_1+\mathsf{Id}})$ (Proposition 2). Let $\mathsf{fibpre} = (\mathsf{id}_1 + \langle \mathsf{id}_{\mathbb{N}}, \mathsf{pred} \rangle) \circ \mathsf{pred}$. By Proposition 9, $(\mathbb{N}, \mathsf{fibpre})$ is a recursive $(K_1 + \mathsf{Id} \times (K_1 + \mathsf{Id}))$-coalgebra. Let us have a look at what fibpre does on different values of its argument:

$$\begin{aligned}&\mathsf{fibpre} : \mathbb{N} \to 1 + \mathbb{N} \times (1 + \mathbb{N}) \\ &\mathsf{fibpre}(0) = \mathsf{inl}(*) \\ &\mathsf{fibpre}(1) = \mathsf{inr}(\langle 0, \mathsf{inl}(*) \rangle) \\ &\mathsf{fibpre}(x+2) = \mathsf{inr}(\langle x+1, \mathsf{inr}(x) \rangle).\end{aligned}$$

Now define the $(K_1 + \mathsf{Id} \times (K_1 + \mathsf{Id}))$-algebra $(\mathbb{N}, \mathsf{fibpost})$ as follows:

$$\begin{aligned}
&\mathsf{fibpost} : (1 + \mathbb{N} \times (1 + \mathbb{N})) \to \mathbb{N}\\
&\mathsf{fibpost}(\mathsf{inl}(*)) = 0\\
&\mathsf{fibpost}(\mathsf{inr}(\langle x, \mathsf{inl}(*)\rangle)) = 1\\
&\mathsf{fibpost}(\mathsf{inr}(\langle x, \mathsf{inr}(y)\rangle)) = x + y.
\end{aligned}$$

The Fibonacci function $\mathsf{fib} : \mathbb{N} \to \mathbb{N}$ is the coalgebra-to-algebra morphism $\mathsf{fib} = \mathsf{fix}_{(K_1+\mathsf{Id}\times(K_1+\mathsf{Id})),\mathsf{fibpre}}(\mathsf{fibpost})$ whose unique existence is guaranteed by Proposition 9.

If we look at the proof of Proposition 9, we realize that it gives a reduction of the recursion pattern of fib to standard iteration on the natural numbers. The decomposition of fib as $\mathsf{fst} \circ \mathsf{fix}_{(K_1+\mathsf{Id}),\mathsf{pred}}(\langle \mathsf{fibpost}, \mathsf{id}_1 + \mathsf{fst}\rangle)$ corresponds to the usual "tupling" optimization of the Fibonacci function where the recursion returns a pair of results: not solely the Fibonacci number corresponding to the argument value, but also that for the predecessor. The moral is that the diagram-chasing proof of the proposition is in fact a correctness proof for an optimizing program transformation.

It is clear that a similar proposition, giving an optimization transformation, can be given for any fixed depth of recurrence.

3.2. Simple transposition results

The following two transposition propositions appeared in Eppendahl [11,12].

Proposition 10. *Let $F, G : \mathcal{C} \to \mathcal{C}$ be functors and $\tau : F \dot{\to} G$ a natural transformation.*

(a) *If (A, α) is a F-coalgebra and (C, φ) is a G-algebra, then $f : A \to C$ is a G-coalgebra-to-algebra morphism from $(A, \tau_A \circ \alpha)$ to (C, φ) iff it is a F-coalgebra-to-algebra morphism from (A, α) to $(C, \varphi \circ \tau_C)$.*

(b) *If an F-coalgebra (A, α) is recursive, then the G-coalgebra $(A, \tau_A \circ \alpha)$ is recursive.*

Proof. (a) Immediate from the naturality of τ: $\varphi \circ Gf \circ \tau_A \circ \alpha = \varphi \circ \tau_C \circ Ff \circ \alpha$.

$$\begin{tikzcd}
GA \arrow[ddd, "Gf"'] & & A \arrow[ll, "\tau_A\circ\alpha"'] \arrow[dl, "\alpha"] \arrow[ddd, "f"] \\
& FA \arrow[ul, "\tau_A"] \arrow[d, "Ff"'] & \\
& FC \arrow[dl, "\tau_C"'] \arrow[dr, "\varphi\circ\tau_C"] & \\
GC \arrow[rr, "\varphi"'] & & C
\end{tikzcd}
\qquad
\begin{tikzcd}
FA \arrow[ddd, "Ff"'] \arrow[dr, "\tau_A"'] & & A \arrow[ll, "\alpha"'] \arrow[dl, "\tau_A\circ\alpha"] \arrow[ddd, "f"] \\
& GA \arrow[d, "Gf"'] & \\
& GC \arrow[dr, "\varphi"] & \\
FC \arrow[ur, "\tau_C"] \arrow[rr, "\varphi\circ\tau_C"'] & & C
\end{tikzcd}$$

(b) For any G-algebra (C, φ), the unique F-coalgebra-to-algebra morphism from (A, α) to $(C, \varphi \circ \tau_C)$ is also a unique G-coalgebra-to-algebra morphism from $(A, \tau_A \circ \alpha)$ to (C, φ). □

Proposition 11. *Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ be functors.*

(a) *If (A,α) is an GF-coalgebra and (C,φ) is a FG-algebra, then there is a bijection between FG-coalgebra-to-algebra morphisms from $(FA,F\alpha)$ to (C,φ) and GF-coalgebra-to-algebra morphisms from (A,α) to $(GC,G\varphi)$.*

(b) *If (A,α) is a recursive GF-coalgebra, then $(FA,F\alpha)$ is a recursive FG-coalgebra.*

Proof. (a) For a GF-coalgebra-to-algebra morphism f from (A,α) to $(GC,G\varphi)$, set $f^\star = \varphi \circ Ff : FA \to C$. For an FG-coalgebra-to-algebra morphism g from $(FA,F\alpha)$ to (C,φ), set $g^\dagger = Gg \circ \alpha : A \to GC$. Now $f^\star$ is an FG-coalgebra-to-algebra morphism from $(FA,F\alpha)$ to (C,φ) since $f^\star = \varphi \circ Ff = \varphi \circ F(G(\varphi \circ Ff) \circ \alpha) = \varphi \circ F(Gf^\star \circ \alpha)$ and similarly $g^\dagger$ is a GF-coalgebra-to-algebra morphism from (A,α) to $(GC,G\varphi)$. Further, $(f^\star)^\dagger = Gf^\star \circ \alpha = G(\varphi \circ Ff) \circ \alpha = f$ and similarly $(g^\dagger)^\star = g$.

$$\begin{tikzcd}
FGFA \arrow[d, "FGf^\star"', "FGg"] & FA \arrow[l, "F\alpha"'] \arrow[d, "f^\star"', "g"] \arrow[dl, "Ff"] \\
FGC \arrow[r, "\varphi"'] & C
\end{tikzcd}
\qquad
\begin{tikzcd}
GFA \arrow[d, "GGf^\star"', "GFg^\dagger"] \arrow[dr, "Gg"] & A \arrow[l, "\alpha"'] \arrow[d, "f"', "g^\dagger"] \\
GFGC \arrow[r, "G\varphi"'] & GC
\end{tikzcd}$$

(b) If (C,φ) is a FG-algebra, then the unique GF-coalgebra-to-algebra morphism from (A,α) to $(GC,G\varphi)$ gives, by part (a), a unique FG-coalgebra-to-algebra morphism from $(FA,F\alpha)$ to (C,φ). □

The following proposition builds on Propositions 10 and 11.

Proposition 12. *Let $F : \mathcal{C} \to \mathcal{C}$, $G : \mathcal{D} \to \mathcal{D}$ be functors, $L : \mathcal{C} \to \mathcal{D}$ a functor with a right adjoint, and $\tau : LF \stackrel{\cdot}{\to} GL$ a natural transformation. If (A,α) is a recursive F-coalgebra, then $(LA,\tau_A \circ L\alpha)$ is a recursive G-coalgebra.*

Proof. Let R be the right adjoint of L and $\eta : \mathsf{Id} \stackrel{\cdot}{\to} RL$ and $\varepsilon : LR \stackrel{\cdot}{\to} \mathsf{Id}$ the unit resp. counit of the adjunction. Let $\lambda(\cdot)$ denote the natural bijection between the homsets $\mathcal{D}(L-,=)$ and $\mathcal{C}(-,R=)$. Now, let $\beta = \lambda(\tau_A \circ L\alpha) = R(\tau_A \circ L\alpha) \circ \eta_A = R\tau_A \circ \eta_{FA} \circ \alpha = (R\tau \circ \eta F)_A \circ \alpha : A \to RGLA$.

According to Proposition 10, the RGL-coalgebra (A,β) is recursive. But then by Proposition 11, the LRG-coalgebra $(LA,L\beta)$ is recursive. By Proposition 10 once more, the G-coalgebra $(LA,\varepsilon_{GLA} \circ L\beta)$ is recursive. But $\varepsilon_{GLA} \circ L\beta = \lambda^{-1}(\beta) = \lambda^{-1}(\lambda(\tau_A \circ L\alpha)) = \tau_A \circ L\alpha$. □

3.3. Variations of recursiveness

We conclude this section by briefly looking at two useful strengthenings of the notion of recursiveness, which we call strong recursiveness and parametric recursiveness.[3]

Strong recursiveness relates to recursiveness for coalgebras as allowing strong iteration (iteration with parameters) relates to allowing iteration (i.e., initiality) for algebras.

[3] Parametric recursiveness was called very recursiveness in the conference version of this paper.

Recall that a strength of a functor F is a natural transformation σ with components $\sigma_{A,B} : A \times FB \to F(A \times B)$ satisfying the equations

$$\begin{aligned} Fr_A \circ \sigma_{1,A} &= r_{FA} \\ Fa_{A,B,C} \circ \sigma_{A\times B,C} &= \sigma_{A,B\times C} \circ (\mathsf{id}_A \times \sigma_{B,C}) \circ a_{A,B,FC}, \end{aligned}$$

where r and a are the canonical natural isomorphisms with components $r_A : 1 \times A \to A$, $a_{A,B,C} : (A \times B) \times C \to A \times (B \times C)$.

Definition 13 (*Strongly recursive coalgebra*). Let $\mathcal{C}$ be cartesian and $F : \mathcal{C} \to \mathcal{C}$ a functor with a strength σ. An F-coalgebra (A, φ) is *strongly recursive* (or recursive with parameters) iff, for any object Γ of $\mathcal{C}$ and F-algebra (C, φ), there is a unique morphism $f : \Gamma \times A \to C$, denoted $\mathsf{sfix}_{F,\Gamma,\alpha}(\varphi)$, satisfying

$$\begin{tikzcd}
F(\Gamma \times A) \arrow[d, "Ff"'] & \Gamma \times FA \arrow[l, "\sigma_{\Gamma,A}"'] & \Gamma \times A \arrow[l, "\mathsf{id}_\Gamma \times \alpha"'] \arrow[d, "f"] \\
FC \arrow[rr, "\varphi"] & & C
\end{tikzcd}$$

It is immediate that an F-coalgebra (A, α) is strongly recursive iff, for any object Γ, the F-coalgebra $(\Gamma \times A, \sigma_{\Gamma,A} \circ (\mathsf{id}_\Gamma \times \alpha))$ is recursive.

A strongly recursive F-coalgebra (A, α) is also a recursive F-coalgebra: for an F-algebra (C, φ), $\mathsf{fix}_{F,\alpha}(\varphi) = \mathsf{sfix}_{F,1,\alpha}(\varphi) \circ \langle\, !_A, \mathsf{id}_A \,\rangle$. For the converse to hold, it is sufficient that $\mathcal{C}$ is cartesian closed: if (A, α) is a recursive F-coalgebra, then, for any object Γ, by Proposition 12 for $\mathcal{D} = \mathcal{C}$, $G = F$, $L = K_\Gamma \times \mathsf{Id}$, $\tau = \sigma_\Gamma$, the F-coalgebra $(\Gamma \times A, \sigma_{\Gamma,A} \circ (\mathsf{id}_\Gamma \times \alpha))$ is recursive.

An object A is the carrier of a final strongly recursive F-coalgebra iff it is the carrier of a strongly initial F-algebra.

Parametric recursiveness is roughly in the same position w.r.t. recursiveness for coalgebras as allowing primitive recursion is w.r.t. initiality for algebras. The new work of Adámek, Milius and Velebil [27,4] on the free completely iterative (resp. iterative) monad of a functor (elaborating on their original approach in [2,3]) is centered around the dual concept (resp. a finitary version of it).

Definition 14 (*Parametrically recursive coalgebra*). Let $\mathcal{C}$ be cartesian and $F : \mathcal{C} \to \mathcal{C}$ a functor. An F-coalgebra (A, α) is *parametrically recursive* iff, for any $(K_A \times F)$-algebra (C, φ), there is a unique morphism $f : A \to C$, denoted $\mathsf{pfix}_{A,\alpha}(\varphi)$, satisfying

$$\begin{tikzcd}
A \times FA \arrow[d, "\mathsf{id}_A \times Ff"'] & A \arrow[l, "\langle \mathsf{id}_A, \alpha \rangle"'] \arrow[d, "f"] \\
A \times FC \arrow[r, "\varphi"] & C
\end{tikzcd}$$

An F-coalgebra (A, α) is parametrically recursive iff the $(K_A \times F)$-coalgebra $(A, \langle \mathsf{id}_A, \alpha \rangle)$ is recursive. A parametrically recursive F-coalgebra (A, α) is necessarily recursive: for an F-algebra (C, φ), $\mathsf{fix}_{F,\alpha}(\varphi) = \mathsf{pfix}_{F,\alpha}(\varphi \circ \mathsf{snd}_{A,FC})$.

But not every recursive coalgebra is parametrically recursive, not even for a cartesian closed category and a strong functor. A beautiful example of a recursive, but not parametrically recursive, coalgebra for **Set** (which is cartesian closed and has every endofunctor strong) appears in the new work by Adámek et al. [1], which was prompted by the conference version of the present paper.

The example is the following: Let R be the quotient of the squaring functor defined by:

$$\begin{aligned} RX &= \{(x, y) \in X \times X \mid x \neq y\} \cup \{d\}, \\ Rf\ (x, y) &= \begin{cases} (f(x), f(y)) & \text{if } f(x) \neq f(y), \\ d & \text{otherwise,} \end{cases} \\ Rf\ (d) &= d. \end{aligned}$$

Now set $A = \{0,1\}$ and define $\alpha : A \to RA$ to be the constant function of value $(0,1)$. The R-coalgebra (A, α) is recursive, since for any R-algebra (C, φ), there is exactly one R-coalgebra-to-algebra morphism between them, namely the constant function of value $\varphi(d)$. At the same time it is not parametrically recursive: Consider any C and $\varphi : A \times RC \to C$ with the property that for several pairs $(x_0, x_1) \in RC$ we have $\varphi(i, (x_0, x_1)) = x_i$ for both $i \in A$. Every such pair gives a $(K_A \times R)$-coalgebra-to-algebra morphism f from $(A, \langle \mathsf{id}_A, \alpha \rangle)$ to (C, φ) to such that $f(i) = x_i$.

Morally, this counterexample is made possible by R not preserving inverse images. For functors on **Set** which preserve monos and inverse images, any recursive coalgebra is also parametrically recursive [1].

The concept of parametrically recursive coalgebras and its dual are elegant and useful because of the following fact whose dual is central in [27].

Proposition 15. *For any object X, an object DX is the carrier of a cofree parametrically recursive F-coalgebra over X iff DX is the carrier of an initial $(K_X \times F)$-algebra.*

With 'parametrically recursive' replaced with 'recursive', this equivalence is valid in the degenerate case $X = 1$ (an object A carries a final recursive F-coalgebra iff it carries an initial F-algebra), but not generally, unless additional assumptions are made.

4. Recursive coalgebras from (cofree) comonads

We shall now proceed to more powerful sufficient conditions for a coalgebra being recursive. These are based on comonads, comonad-algebras and distributive laws of a functor over a comonad. We recall the definitions.

Definition 16 (*Comonad*)**.** A *comonad* on a category $\mathcal{C}$ is a functor $D : \mathcal{C} \to \mathcal{C}$ together with natural transformations $\varepsilon : D \dot{\to} \mathsf{Id}$ (counit) and $\delta : D \dot{\to} D^2$ (comultiplication) satisfying, for any object X,

$$\begin{tikzcd}
DX \arrow[r, "\delta_X"] \arrow[d, "\delta_X"'] \arrow[dr, equal] & D^2X \arrow[d, "\varepsilon_{DX}"] \\
D^2X \arrow[r, "D\varepsilon_X"] & DX
\end{tikzcd}
\qquad
\begin{tikzcd}
DX \arrow[r, "\delta_X"] \arrow[d, "\delta_X"'] & D^2X \arrow[d, "\delta_{DX}"] \\
D^2X \arrow[r, "D\delta_X"] & D^3X
\end{tikzcd}$$

Definition 17 (*Coalgebra of a comonad*). A *coalgebra of a comonad* (D, ε, δ) on $\mathcal{C}$ is a coalgebra (A, ι) of the functor D satisfying

$$\begin{tikzcd}
A \arrow[r, "\iota"] \arrow[dr, equal] & DA \arrow[d, "\varepsilon_A"] \\
& A
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "\iota"] \arrow[d, "\iota"'] & DA \arrow[d, "\delta_A"] \\
DA \arrow[r, "D\iota"] & D^2A
\end{tikzcd}$$

Definition 18 (*Distributive law over a comonad*). A *distributive law of a functor* $F : \mathcal{C} \to \mathcal{C}$ over a *comonad* (D, ε, δ) on $\mathcal{C}$ is a natural transformation $\kappa : FD \dot{\to} DF$ satisfying, for any object X,

$$\begin{tikzcd}
FDX \arrow[r, "\kappa_X"] \arrow[d, "F\varepsilon_X"'] & DFX \arrow[d, "\varepsilon_{FX}"] \\
FX \arrow[r, equal] & FX
\end{tikzcd}
\qquad
\begin{tikzcd}
FDX \arrow[rr, "\kappa_X"] \arrow[d, "F\delta_X"'] & & DFX \arrow[d, "\delta_{FX}"] \\
FD^2X \arrow[r, "\kappa_{DX}"] & DFDX \arrow[r, "D\kappa_X"] & D^2FX
\end{tikzcd}$$

We present four theorems, each saying that a coalgebra constructed in a certain fashion from a coalgebra known to be recursive is recursive as well. The first of these (Theorem 19) uses a general comonad. The other three concern special cases: the second (Theorem 21) is about product comonads while the third and fourth (Theorems 23 and 26) are about cofree comonads. Theorem 26 is special in using no data relating to the cofree comonad (this comonad only has to exist), instead it engages data relating to the inducing functor.

4.1. Recursive coalgebras from general comonads

The most central for us is the first theorem, which uses a general comonad.

Theorem 19 (Main theorem 1). *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A, α) a recursive F-coalgebra, $\mathbf{D} = (D, \varepsilon, \delta)$ a comonad on $\mathcal{C}$ and (A, ι) a $\mathbf{D}$-coalgebra. If κ is a distributive law of F over $\mathbf{D}$ satisfying*

$$\begin{tikzcd}
FA \arrow[d, "F\iota"'] & & A \arrow[ll, "\alpha"'] \arrow[d, "\iota"] & (*) \\
FDA \arrow[r, "\kappa_A"] & DFA & DA \arrow[l, "D\alpha"'] &
\end{tikzcd}$$

then $(A, F\iota \circ \alpha)$ is a recursive FD-coalgebra (and, consequently, by Proposition 10, *$(A, D\alpha \circ \iota)$ is a recursive DF-coalgebra).*

It might make sense to define that the data (A, α, ι) form, say, a *dicoalgebra* of $(F, \boldsymbol{D}, \kappa)$ iff they meet the condition (*) and then to develop a theory of functor-comonad-dicoalgebras (cf. the functor-functor-bialgebras of Turi and Plotkin [34] or the monad-functor-bialgebras of [5,8]), but we have chosen not to specifically pursue this line here, as we will not need many properties of dicoalgebras.

Proof. Consider any FD-algebra (C, φ). Let $\psi = D\varphi \circ \kappa_{DC} \circ F\delta_C : FDC \to DC$, $g = \mathsf{fix}_{F,\alpha}(\psi) : A \to DC$ and $f = \varepsilon_C \circ g : A \to C$. We show that (i) f is a FD-coalgebra-to-algebra morphism from $(A, F\iota \circ \alpha)$ to (C, φ), and (ii) it is the only one, i.e., $\mathsf{fix}_{FD,F\iota\circ\alpha}(\varphi) = f$.

Proof of (i): We first notice that $Dg \circ \iota = \mathsf{fix}_{F,\alpha}(D\psi \circ \kappa_{DC}) = \delta_C \circ g$. This is witnessed by the commutativity of the outer squares in the following two diagrams.

$$\begin{tikzcd}
FA \arrow[d, "F\iota"'] & & A \arrow[ll, "\alpha"'] \arrow[d, "\iota"] \\
FDA \arrow[r, "\kappa_A"] \arrow[d, "FDg"'] & DFA \arrow[d, "DFg"'] & DA \arrow[l, "D\alpha"'] \arrow[d, "Dg"] \\
FD^2C \arrow[r, "\kappa_{DC}"] & DFDC \arrow[r, "D\psi"] & D^2C
\end{tikzcd}$$

$$\begin{tikzcd}
FA \arrow[d, "Fg"'] & & & & A \arrow[llll, "\alpha"'] \arrow[d, "g"] \\
FDC \arrow[r, "F\delta_C"'] \arrow[rrrr, bend left, "\psi"] \arrow[dd, "F\delta_C"'] & FD^2C \arrow[rr, "\kappa_{DC}"'] \arrow[d, "F\delta_{DC}"'] & & DFDC \arrow[r, "D\varphi"'] \arrow[dd, "\delta_{FDC}"] & DC \arrow[dd, "\delta_C"] \\
& FD^3C \arrow[dr, "\kappa_{D^2C}"] & & & \\
FD^2C \arrow[r, "\kappa_{DC}"] \arrow[ur, "FD\delta_C"] & DFDC \arrow[r, "DF\delta_C"] \arrow[rrr, bend right, "D\psi"'] & DFD^2C \arrow[r, "D\kappa_{DC}"] & D^2FDC \arrow[r, "D^2\varphi"] & D^2C
\end{tikzcd}$$

In the first diagram of these diagrams, the top square commutes by (*), the bottom left square commutes by naturality of κ and the bottom right square commutes by g being a coalgebra-to-algebra morphism from (A, α) to (DC, ψ).

In the second diagram, the inner shapes commute by the following reasons (from top to bottom and from left to right): g being a coalgebra-to-algebra morphism from (A, α) to (DC, ψ), the third equation for comonads, the second equation for distributive laws, the naturality of δ, and the naturality of κ.

Now the desired equality $f = \varphi \circ F(Df \circ \iota) \circ \alpha$ is witnessed by the commutativity of the outer square in the diagram

$$\begin{tikzcd}
FDA \arrow[d, "FDg"] \arrow[dd, bend right=60, "FDf"'] & FA \arrow[l, "F\iota"'] \arrow[d, "Fg"] & & & A \arrow[llll, "\alpha"'] \arrow[d, "g"'] \arrow[dd, bend left=60, "f"] \\
FD^2C \arrow[d, "FD\varepsilon_C"] & FDC \arrow[l, "F\delta_C"] \arrow[r, "F\delta_C"'] \arrow[d, equal] \arrow[rrr, bend left, "\psi"] & FD^2C \arrow[r, "\kappa_{DC}"'] \arrow[d, "F\varepsilon_{DC}"] & DFDC \arrow[r, "D\varphi"'] \arrow[d, "\varepsilon_{FDC}"] & DC \arrow[d, "\varepsilon_C"'] \\
FDC \arrow[r, equal] & FDC \arrow[r, equal] & FDC \arrow[r, equal] & FDC \arrow[r, "\varphi"] & C
\end{tikzcd}$$

where the square at the top left corner expresses what we just proved. The other inner shapes commute by the following reasons: g being a coalgebra-to-algebra morphism from (A,α) to (DC,ψ), the second equation for comonads, the first equation for comonads, first equation for distributive laws, and the naturality of ε.

Proof of (ii): Suppose f' is a FD-coalgebra-to-algebra morphism from $(A, F\iota \circ \alpha)$ to (C,φ). We observe that the commuting outer square in the following diagram proves that $g = \mathsf{fix}_{F,\alpha}(\psi) = Df' \circ \iota$.

$$\begin{tikzcd}
FA \arrow[dr, "F\iota"] \arrow[dd, "F\iota"'] & & & A \arrow[lll, "\alpha"'] \arrow[d, "\iota"] \\
& FDA \arrow[r, "\kappa_A"] \arrow[d, "FD\iota"'] & DFA \arrow[d, "DF\iota"] & DA \arrow[l, "D\alpha"'] \arrow[dd, "Df'"] \\
FDA \arrow[r, "F\delta_A"] \arrow[d, "FDf'"'] & FD^2A \arrow[r, "\kappa_{DA}"] \arrow[d, "FD^2f'"'] & DFDA \arrow[d, "DFDf'"] & \\
FDC \arrow[r, "F\delta_C"] \arrow[rrr, bend right, "\psi"'] & FD^2C \arrow[r, "\kappa_{DC}"] & DFDC \arrow[r, "D\varphi"] & DC
\end{tikzcd}$$

where the inner shapes commute by the following reasons: the second equation of comonad-coalgebras, (*), the naturality of κ, f' being a coalgebra-to-algebra morphism from $(A, F\iota \circ \alpha)$ to (C,φ), the naturality of δ.

From here, by the first equation of comonad-coalgebras and the naturality of ε, it follows that $f' = f' \circ \varepsilon_A \circ \iota = \varepsilon_C \circ Df' \circ \iota = \varepsilon_C \circ g = f$. □

Notice that all assumptions of the theorem (incl. all equations of a comonad, a comonad-coalgebra and a distributive law) are used in the proof. In the rest of the paper we will omit the justification of the commutativity of a diagram whenever it follows immediately from the assumptions.

Theorem 19 provides a powerful generalization of the central theorem of Uustalu et al. [39], which was on structured recursion schemes for initial algebras derivable from comonads (cf. also the dual result stated in [5,8]). Indeed, the following corollary shows that the theorem of [39] is just a special case of Theorem 19.

Corollary 20. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor with an initial algebra and $\boldsymbol{D} = (D, \varepsilon, \delta)$ a comonad on $\mathcal{C}$. Let κ be a distributive law of F over $\boldsymbol{D}$; define a map $\iota : \mu F \to D\mu F$ by $\iota = \mathsf{It}_F(D\mathsf{in}_F \circ \kappa_{\mu F})$. Then $(\mu F, F\iota \circ \mathsf{in}_F^{-1})$ is a recursive FD-coalgebra.*

Proof. The commutativity of the outer square of the following diagram says that $\varepsilon_{\mu F} \circ \iota = \mathsf{It}_F(\,\mathsf{in}_F\,)$. But $\mathsf{It}_F(\,\mathsf{in}_F\,) = \mathsf{id}_{\mu F}$.

$$\begin{tikzcd}
F\mu F \arrow[rr, "\mathsf{in}_F"] \arrow[d, "F\iota"'] & & \mu F \arrow[d, "\iota"] \\
FD\mu F \arrow[r, "\kappa_{\mu F}"] \arrow[d, "F\varepsilon_{\mu F}"'] & DF\mu F \arrow[r, "D\mathsf{in}_F"] \arrow[d, "\varepsilon_{F\mu F}"'] & D\mu F \arrow[d, "\varepsilon_{\mu F}"] \\
F\mu F \arrow[r, equal] & F\mu F \arrow[r, "\mathsf{in}_F"] & \mu F
\end{tikzcd}$$

The commutativity of the outer squares of the two following diagrams proves that $\delta_{\mu F} \circ \iota = \mathsf{It}_F(\,D^2\mathsf{in}_F \circ D\kappa_{\mu F} \circ \kappa_{D\mu F}\,) = D\iota \circ \iota$.

$$\begin{tikzcd}
F\mu F \arrow[rrr, "\mathsf{in}_F"] \arrow[d, "F\iota"'] & & & \mu F \arrow[d, "\iota"] \\
FD\mu F \arrow[rr, "\kappa_{\mu F}"] \arrow[d, "F\delta_{\mu F}"'] & & DF\mu F \arrow[r, "D\mathsf{in}_F"] \arrow[d, "\delta_{F\mu F}"'] & D\mu F \arrow[d, "\delta_{\mu F}"] \\
FD^2\mu F \arrow[r, "\kappa_{D\mu F}"] & DFD\mu F \arrow[r, "D\kappa_{\mu F}"] & D^2F\mu F \arrow[r, "D^2\mathsf{in}_F"] & D^2\mu F
\end{tikzcd}$$

$$\begin{tikzcd}
F\mu F \arrow[rrr, "\mathsf{in}_F"] \arrow[d, "F\iota"'] & & & \mu F \arrow[d, "\iota"] \\
FD\mu F \arrow[r, "\kappa_{\mu F}"] \arrow[d, "FD\iota"'] & DF\mu F \arrow[rr, "D\mathsf{in}_F"] \arrow[d, "DF\iota"'] & & D\mu F \arrow[d, "D\iota"] \\
FD^2\mu F \arrow[r, "\kappa_{D\mu F}"] & DFD\mu F \arrow[r, "D\kappa_{\mu F}"] & D^2F\mu F \arrow[r, "D^2\mathsf{in}_F"] & D^2\mu F
\end{tikzcd}$$

Hence $(\mu F, \iota)$ is a $\boldsymbol{D}$-coalgebra. It is immediate that it relates appropriately to the recursive F-coalgebra $(\mu F, \mathsf{in}_F^{-1})$ via κ. Hence, by Theorem 19, $(\mu F, F\iota \circ \mathsf{in}_F^{-1})$ is a recursive FD-coalgebra. □

We learn that the result in [39] was provable not so much because of the initiality of the initial F-algebra $(\mu F, \mathsf{in}_F)$ as it was because of the recursiveness of its inverse F-coalgebra $(\mu F, \mathsf{in}_F^{-1})$: the coalgebra $(\mu F, \mathsf{in}_F^{-1})$ can be replaced by a recursive coalgebra (A, α) to obtain a more general statement whereas one cannot replace $(\mu F, \mathsf{in}_F)$ with some other algebra. This is an indication that recursive coalgebras are a more basic concept for discussing structured recursion than initial algebras!

4.2. Recursive coalgebras from product comonads

An interesting special case of comonads are product comonads. In a cartesian category, every object E gives rise to a comonad $\boldsymbol{D}^E = (D^E, \varepsilon^E, \delta^E)$ defined by $D^E X = X \times E$, $\varepsilon^E_X = \mathsf{fst}_{X,E}$ and $\delta^E_X = \langle\, \mathsf{id}_{X\times E}, \mathsf{snd}_{X,E}\,\rangle$. The $\boldsymbol{D}^E$-coalgebras are in 1-1 correspondence with the maps to E and the distributive laws of F over $\boldsymbol{D}^E$ are in 1-1 correspondence with the F-algebra structures carried by E. For a map $e : A \to E$, the corresponding $\boldsymbol{D}^E$-coalgebra structure $\bar{e} : A \to D^E A$ is defined by $\bar{e} = \langle\, \mathsf{id}_A, e\,\rangle$; for an F-algebra (E, χ), the corresponding distributive law $\bar{\chi}$ of F over $\boldsymbol{D}^E$ is defined by $\bar{\chi}_X = \langle\, F\mathsf{fst}_{X,E}, \chi \circ F\mathsf{snd}_{X,E}\,\rangle$.

For product comonads Theorem 19 specializes considerably, yielding the second theorem of this section.

Theorem 21 (Specialization for product comonads)**.** *Assume $\mathcal{C}$ is cartesian. Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A, α) a recursive F-coalgebra and (E, χ) an F-algebra. Then $(A, F\overline{\mathsf{fix}_{F,\alpha}(\chi)} \circ \alpha)$ is a recursive FD^E-coalgebra.*

Proof. We apply Theorem 19 with $\boldsymbol{D} = \boldsymbol{D}^E$, $\iota = \overline{\mathsf{fix}_{F,\alpha}(\chi)}$ and $\kappa = \bar{\chi}$. The only fact that we need to verify is the commutativity of the diagram in the statement of the theorem:

$$\begin{aligned}
D\alpha \circ \iota &= D\alpha \circ \overline{\mathsf{fix}_{F,\alpha}(\chi)}\\
&= (\alpha \times \mathsf{id}_E) \circ \langle\, \mathsf{id}_A, \mathsf{fix}_{F,\alpha}(\chi)\,\rangle)\\
&= \langle\, \alpha, \mathsf{fix}_{F,\alpha}(\chi)\,\rangle\\
&= \langle\, \alpha, \chi \circ F\mathsf{fix}_{F,\alpha}(\chi) \circ \alpha\,\rangle\\
&= \langle\, F\mathsf{fst}_{A,E}, \chi \circ F\mathsf{snd}_{A,E}\,\rangle \circ F\langle\, \mathsf{id}_A, \mathsf{fix}_{F,\alpha}(\chi)\,\rangle \circ \alpha\\
&= \bar{\chi}_A \circ F\overline{\mathsf{fix}_{F,\alpha}(\chi)} \circ \alpha\\
&= \kappa_A \circ F\iota \circ \alpha.
\end{aligned}$$

The conclusion of Theorem 19 then gives us that $(A, F\iota \circ \alpha) = (A, F\overline{\mathsf{fix}_{F,\alpha}(\chi)} \circ \alpha)$ is a recursive FD^E-coalgebra as desired. □

The following corollary generalizes primitive recursion from initial algebras to recursive coalgebras with split monic structure morphism.

Corollary 22. *If F is a functor and (A, α) is a recursive F-coalgebra with α split monic*, *then* (*by Theorem* 21 *for* $E = A$, $\chi = \alpha^-$, *where* α^- *is the postinverse of* α) *the* FD^A-*coalgebra* $(A, F\langle\, \mathsf{id}_A, \mathsf{id}_A\,\rangle \circ \alpha)$ *is recursive* (*since* $\mathsf{fix}_{F,\alpha}(\alpha^-) = \mathsf{id}_A$).

Let us give a concrete example of application of Theorem 21. Let Tree be the set of finitely branching wellfounded trees, in other words, $\mathsf{Tree} = \mu\mathsf{List}$. Let us call $\mathsf{subtrees}$ the function computing the list of subtrees: $\mathsf{subtrees} = \mathsf{in}^{-1}_{\mathsf{List}}$; and let $\mathsf{depth} : \mathsf{Tree} \to \mathbb{N}$ be the function returning the depth of a tree, $\mathsf{depth} = \mathsf{fix}_{\mathsf{List},\mathsf{subtrees}}(\max)$. We might wish to define a new function $\mathsf{depth\_power} : \mathsf{Tree} \to \mathbb{N}$ by the following recursive equation:

$$\mathsf{depth\_power}\ t = \sum_{i=1}^{n} (\mathsf{depth\_power}\ t_i)^{(\mathsf{depth}\ t_i)},$$

where $t_1, \ldots, t_n$ are the subtrees of t (we use the convention that $0^0 = 1$).

The existence of a unique solution to this recursive equation is guaranteed by the theorem: take $F = \mathsf{List}, A = \mathsf{Tree}, \alpha = \mathsf{subtrees}, E = \mathbb{N}, \chi = \max$.

4.3. Recursive coalgebras from cofree comonads

A wider useful class of comonads are comonads cofree over a functor. Product comonads are a degenerate case corresponding to comonads cofree over constant functors.

Let H be a functor such that, for every object X, the functor $K_X \times H$ has a final coalgebra $(D^H X, \langle \varepsilon^H_X, \theta^H_X \rangle)$ (this has been called *iteratability* [2]). (Intuitively, one should think of $D^H X$ as the set of X-decorated non-wellfounded trees with branching type H, $\varepsilon^H_X : D^H X \to X$ then returns the root decoration of a given tree and $\theta^H_X : D^H X \to H D^H X$ returns the subtrees.) The H-coalgebra $(D^H X, \theta^H_X)$ is the *cofree H-coalgebra* generated by X. The morphism ε_X is the associated projection.

The finality of the coalgebra $(D^H X, \langle \varepsilon^H_X, \theta^H_X \rangle)$ can be expressed as follows: For every H-coalgebra (C, φ), object X and morphism $\chi : C \to X$, there exists a unique morphism f, denoted $\mathsf{gen}^H_X(\chi, \varphi)$, making the following diagram commute.

$$\begin{tikzcd}
 & C \arrow[r, "\varphi"] \arrow[dl, "\chi"'] \arrow[d, "f"] & HC \arrow[d, "Hf"] \\
X & D^H X \arrow[l, "\varepsilon^H_X"] \arrow[r, "\theta^H_X"'] & H D^H X
\end{tikzcd}$$

A special case is obtained by choosing both X and C to be $D^H X$ and χ and φ to be $\mathsf{id}_{D^H X}$ resp. θ^H_X. Then we can define a morphism $\delta^H_X = \mathsf{gen}^H_{D^H X}(\mathsf{id}_{D^H X}, \theta^H_X) : D^H X \to D^H D^H X$. As a result, we obtain a comonad $\boldsymbol{D}^H = (D^H, \varepsilon^H, \delta^H)$. This is the *cofree comonad* generated by H. The associated projection is $\sigma^H : D^H \dot{\to} H$, defined by $\sigma^H_X = H\varepsilon^H_X \circ \theta^H_X : D^H X \to HX$ (think of it as giving the root decorations of the subtrees of a given tree). For any H-coalgebra (C, φ), object X and morphism $\chi : C \to X$, we have that $\sigma^H_X \circ \mathsf{gen}^H_X(\chi, \varphi) = H(\varepsilon^H_X \circ \mathsf{gen}^H_X(\chi, \varphi)) \circ \varphi = H\chi \circ \varphi$.

The $\boldsymbol{D}^H$-coalgebras are in 1-1 correspondence with the H-coalgebras. The distributive laws of F over $\boldsymbol{D}^H$ are in 1-1 correspondence with the natural transformations $FD^H \to HF$ (see, e.g. [5]). The $\boldsymbol{D}^H$-coalgebra structure $\bar{\jmath} : A \to D^H A$ given by a H-coalgebra structure $\jmath : A \to HA$ is defined by $\bar{\jmath} = \mathsf{gen}^H_A(\mathsf{id}_A, \jmath)$. The distributive law $\bar{\lambda}$ of F over $\boldsymbol{D}^H$ given by a natural transformation $\lambda : FD^H \to HF$ is defined by $\bar{\lambda}_X = \mathsf{gen}^H_{FX}(F\varepsilon^H_X, \lambda_{D^H X} \circ F\delta^H_X)$.

Note that product comonads are a special case of cofree comonads: The product comonad given by an object E is the cofree comonad of the functor K_E.

For cofree comonads, by specializing Theorem 19, we obtain the following theorem (the third theorem).

Theorem 23 (Specialization for cofree comonads)**.** *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A,α) a recursive F-coalgebra, $H : \mathcal{C} \to \mathcal{C}$ a functor with a cofree comonad and $(A, \jmath)$ a H-coalgebra. If $\lambda : FD^H \dot{\to} HF$ is a natural transformation satisfying*

$$\begin{tikzcd}
FA \arrow[d, "F\bar{\jmath}"'] & & A \arrow[ll, "\alpha"'] \arrow[d, "\jmath"] \\
FD^H A \arrow[r, "\lambda_A"] & HFA & HA \arrow[l, "H\alpha"']
\end{tikzcd}$$

then $(A, F\bar{\jmath} \circ \alpha)$ is a recursive FD^H-coalgebra.

Proof. The commutativity of the outer triangles and squares in the following diagrams gives us that $D^H\alpha \circ \bar{\jmath} = \mathsf{gen}^H_{FA}(\alpha, \jmath) = \bar{\lambda}_A \circ F\bar{\jmath} \circ \alpha$.

$$\begin{tikzcd}
 & A \arrow[r, "\jmath"] \arrow[d, "\bar{\jmath}"] \arrow[dl, equal] & HA \arrow[d, "H\bar{\jmath}"] \\
A \arrow[d, "\alpha"'] & D^H A \arrow[l, "\varepsilon^H_A"'] \arrow[r, "\theta^H_A"] \arrow[d, "D^H\alpha"] & HD^H A \arrow[d, "HD^H\alpha"] \\
FA & D^H FA \arrow[l, "\varepsilon^H_{FA}"'] \arrow[r, "\theta^H_{FA}"] & HD^H FA
\end{tikzcd}$$

$$\begin{tikzcd}
 & A \arrow[rr, "\jmath"] \arrow[d, "\alpha"] & & HA \arrow[d, "H\alpha"] \\
 & FA \arrow[r, "F\bar{\jmath}"] \arrow[d, "F\bar{\jmath}"] \arrow[dl, equal] & FD^H A \arrow[r, "\lambda_A"] \arrow[d, "FD^H\bar{\jmath}"] & HFA \arrow[d, "HF\bar{\jmath}"] \\
FA \arrow[d, equal] & FD^H A \arrow[l, "F\varepsilon^H_A"'] \arrow[r, "F\delta^H_A"] \arrow[d, "\bar{\lambda}_A"] & F(D^H)^2 A \arrow[r, "\lambda_{D^H A}"] & HFD^H A \arrow[d, "H\bar{\lambda}_A"] \\
FA & D^H FA \arrow[l, "\varepsilon^H_{FA}"'] \arrow[rr, "\theta^H_{FA}"] & & HD^H FA
\end{tikzcd}$$

Therefore, by Theorem 19 (taking $\boldsymbol{D} = \boldsymbol{D}^H$, $\iota = \bar{\jmath}$, $\kappa = \bar{\lambda}$), we get that $(A, F\bar{\jmath} \circ \alpha)$ is a recursive FD^H-coalgebra. □

From this theorem we immediately get an extension of course-of-value iteration [36] from initial algebras to arbitrary recursive coalgebras:

Corollary 24. *If F is a functor with a cofree comonad and (A,α) is a recursive F-coalgebra, then (by Theorem 23 for $H = F$, $\jmath = \alpha$, $\lambda = F\sigma^F$) the FD^F-coalgebra $(A, F\mathsf{gen}^F_A(\mathsf{id}_A, \alpha) \circ \alpha)$ is recursive.*

The original course-of-value iteration becomes a corollary of a corollary:

Corollary 25. *If F is a functor with a cofree comonad and an initial algebra, then* (*by Corollary* 24 *for $A = \mu F, \alpha = \mathsf{in}_F^{-1}$*) *the FD^F-coalgebra $(\mu F, F\mathsf{gen}^F_{\mu F}(\mathsf{id}_{\mu F}, \mathsf{in}_F^{-1}) \circ \mathsf{in}_F^{-1})$ is recursive.*

4.4. Recursive coalgebras from cofree comonads (2)

As a consequence of Theorem 23, we can also derive a theorem where the cofree comonad no longer appears manifestly, but is nonetheless present in the background (our fourth theorem). Instead of a comonad, comonad-coalgebra and a distributive law of a functor over a comonad as in Theorem 19, this theorem uses a functor, a functor-coalgebra and a distributive law of a functor over a functor.

Theorem 26 (Main theorem 2). *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor,* (A, α) *a recursive F-coalgebra, $H : \mathcal{C} \to \mathcal{C}$ a functor with a cofree comonad and* $(A, \jmath)$ *a H-coalgebra. If $\lambda' : FH \dot{\to} HF$ is a natural transformation satisfying*

$$\begin{tikzcd}
FA & & A \arrow[ll, "\alpha"'] \arrow[d, "\jmath"] \\
FHA \arrow[u, leftarrow, "F\jmath"'] \arrow[r, "\lambda'_A"] & HFA & HA \arrow[l, "H\alpha"']
\end{tikzcd}$$

then $(A, F\jmath \circ \alpha)$ is a recursive FH-coalgebra.

Proof. Define a natural transformation $\lambda : FD^H \dot{\to} HF$ by $\lambda_X = \lambda'_X \circ F\sigma^H_X$. We get that $\lambda_A \circ F\bar{\jmath} = \lambda'_A \circ F(\sigma^H_A \circ \mathsf{gen}^H_A(\mathsf{id}_A, \jmath)) = \lambda'_A \circ F(H\mathsf{id}_A \circ \jmath) = \lambda'_A \circ F\jmath$. Hence, by Theorem 23, $(A, F\bar{\jmath} \circ \alpha)$ is a recursive FD^H-coalgebra.

Now consider an arbitrary FH-algebra (C, φ). Let $\psi = \varphi \circ F\sigma^H_C : FD^H C \to C$. The following diagram witnesses that a morphism $f : A \to C$ is a FH-coalgebra-to-algebra morphism from $(A, F\jmath \circ \alpha)$ to (C, φ) iff it is a FD^H-coalgebra-to-algebra morphism from $(A, F\bar{\jmath} \circ \alpha)$ to (C, ψ).

$$\begin{tikzcd}
FD^H A \arrow[r, "F\sigma^H_A"'] \arrow[d, "FD^H f"'] & FHA \arrow[d, "FHf"'] & FA \arrow[l, "F\jmath"] \arrow[ll, bend right, "F\bar{\jmath}"'] & A \arrow[l, "\alpha"'] \arrow[d, "f"] \\
FD^H C \arrow[r, "F\sigma^H_C"] \arrow[rrr, bend right, "\psi"'] & FHC \arrow[rr, "\varphi"] & & C
\end{tikzcd}$$

Hence $(A, F\jmath \circ \alpha)$ is a recursive FH-coalgebra, with $\mathsf{fix}_{FH, F\jmath\circ\alpha}(\varphi) = \mathsf{fix}_{FD^H, F\bar{\jmath}\circ\alpha}(\psi)$. □

Corollary 27. *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor with an initial algebra and $H : \mathcal{C} \to \mathcal{C}$ a functor with a cofree comonad. Let $\lambda' : FH \dot{\to} HF$ be a natural transformation; define a morphism $\jmath = \mathsf{It}_F(H\mathsf{in}_F \circ \lambda'_{\mu F}) : \mu F \to H\mu F$. Then $(\mu F, F\jmath \circ \mathsf{in}_F^{-1})$ is a recursive FH-coalgebra.*

Proof. It is obvious that the H-coalgebra $(\mu F, \jmath)$ relates appropriately to the recursive F-coalgebra $(\mu F, \mathsf{in}_F^{-1})$ via λ'. $\square$

From Theorem 26, Proposition 9 from Section 3 is immediate under the assumption that there is a cofree comonad for the functor $\mathsf{Id} \times F$: Given a recursive F-coalgebra (A, α), the recursiveness of the $F(\mathsf{Id} \times F)$-coalgebra $(A, F\langle\, \mathsf{id}_A, \alpha\,\rangle \circ \alpha)$ is the conclusion of Theorem 26 for $H = \mathsf{Id} \times F$, $\jmath = \langle\, \mathsf{id}_A, \alpha\,\rangle$ and $\lambda'_X = \langle\, F\mathsf{fst}_{X,FX}, F\mathsf{snd}_{X,FX}\,\rangle : F(X \times FX) \to FX \times F^2X$.

5. Recursive coalgebras from countable products

Theorem 26 gives a condition for the recursiveness of an FH-coalgebra assuming the existence of a cofree comonad of the functor H, but no real data involving that comonad. Under such circumstances one may ask if the existence of the cofree comonad is really necessary. The answer to this question is: no, it is not. We proceed now to a variant of Theorem 26 where instead of the existence of a cofree comonad, existence of countable products is assumed. The statement and proof of this theorem are inspired by a theorem of Bartels [5]. Our theorem generalizes the dual of Bartels's theorem from initial algebras to recursive coalgebras.

Theorem 28 (Variant of Main theorem 2). *Assume $\mathcal{C}$ has countable products. Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A, α) a recursive F-coalgebra, $H : \mathcal{C} \to \mathcal{C}$ a functor and $(A, \jmath)$ an H-coalgebra. If $\lambda : FH \dot{\to} HF$ is a natural transformation satisfying*

$$\begin{tikzcd}
FA \arrow[d, "F\jmath"'] & & A \arrow[ll, "\alpha"'] \arrow[d, "\jmath"] \\
FHA \arrow[r, "\lambda_A"] & HFA & HA \arrow[l, "H\alpha"']
\end{tikzcd}$$

then $(A, F\jmath \circ \alpha)$ is a recursive FH-coalgebra.

Proof. First define, for any natural number i, a functor H^i by $H^0X = X, H^{i+1}X = H^iHX$, a map $\jmath^i : A \to H^iA$ by $\jmath^0 = \mathsf{id}_A$, $\jmath^{i+1} = H^i\jmath \circ \jmath^i$ and a natural transformation $\lambda^i : FH^i \dot{\to} H^iF$ by $\lambda^0_X = \mathsf{id}_{FX}$, $\lambda^{i+1}_X = H^i\lambda_X \circ \lambda^i_{HX}$. It is immediate (by trivial inductions on i) that one also gets $H^{i+1}X = HH^iX$, $\jmath^{i+1} = H\jmath^i \circ \jmath$ and $\lambda^{i+1}_X = H\lambda^i_X \circ \lambda_{H^iX}$. By induction on i also the pentagon

$$\begin{tikzcd}
FA \arrow[d, "F\jmath^i"'] & & A \arrow[ll, "\alpha"'] \arrow[d, "\jmath^i"] \\
FH^iA \arrow[r, "\lambda^i_A"] & H^iFA & H^iA \arrow[l, "H^i\alpha"']
\end{tikzcd}$$

commutes. Indeed, case 0 is trivial and case $i + 1$ is proved by the diagram

$$\begin{tikzcd}
FA \arrow[d, "F\jmath^i"] \arrow[dd, bend right=60, "F\jmath^{i+1}"'] & & & A \arrow[lll, "\alpha"'] \arrow[d, "\jmath^i"'] \arrow[dd, bend left=60, "\jmath^{i+1}"] \\
FH^iA \arrow[r, "\lambda^i_A"] \arrow[d, "FH^i\jmath"] & H^iFA \arrow[d, "H^iF\jmath"] \arrow[r, phantom, "\text{ind.hyp.}" near start, shift left=25] & & H^iA \arrow[ll, "H^i\alpha"'] \arrow[d, "H^i\jmath"'] \\
FH^{i+1}A \arrow[r, "\lambda^i_{HA}"] \arrow[rr, bend right, "\lambda^{i+1}_A"'] & H^iFHA \arrow[r, "H^i\lambda_A"] & H^{i+1}FA & H^{i+1}A \arrow[l, "H^{i+1}\alpha"']
\end{tikzcd}$$

Now consider any FH-algebra (C,φ). Let $\psi = \langle H^i\varphi \circ \lambda^i_{HC} \circ Fi+1\rangle_i : F\Pi_iH^iC \to \Pi_iH^iC$, $g = \mathsf{fix}_{F,\alpha}(\psi) : A \to \Pi_iH^iC$ and $f = \pi^0 \circ g : A \to C$. We show that (i) f is a FH-coalgebra-to-algebra morphism from $(A, F\jmath \circ \alpha)$ to (C,φ) and (ii) it is the only one, i.e., $\mathsf{fix}_{FH,F\jmath\circ\alpha}(\varphi) = f$.

Proof of (i): We first notice that

$$\langle H\pi^i\rangle_i \circ Hg \circ \jmath = \mathsf{fix}_{F,\alpha}(\langle H^{i+1}\varphi \circ \lambda^{i+1}_{HC} \circ F\pi^{i+1}\rangle_i) = \langle \pi^{i+1}\rangle_i \circ g.$$

This is witnessed by the commutativity of the outer squares in the following diagrams.

$$\begin{tikzcd}
FA \arrow[d, "F\jmath"'] & & & & A \arrow[llll, "\alpha"'] \arrow[d, "\jmath"] \\
FHA \arrow[r, "\lambda_A"] \arrow[d, "FHg"'] & HFA \arrow[d, "HFg"] & & & HA \arrow[lll, "H\alpha"'] \arrow[d, "Hg"] \\
FH\Pi_iH^iC \arrow[r, "\lambda_{\Pi_iH^iC}"] \arrow[d, "F\langle H\pi^i\rangle_i"'] & HF\Pi_iH^iC \arrow[r, "H\langle F\pi^{i+1}\rangle_i"'] \arrow[rrr, bend left, "H\psi"] & H\Pi_iFH^{i+1}C \arrow[r, "H\Pi_i\lambda^i_{HC}"'] \arrow[d, "\langle H\pi^i\rangle_i"] & H\Pi_iH^iFHC \arrow[r, "H\Pi_iH^i\varphi"'] \arrow[d, "\langle H\pi^i\rangle_i"] & H\Pi_iH^iC \arrow[d, "\langle H\pi^i\rangle_i"'] \\
F\Pi_iH^{i+1}C \arrow[r, "\langle F\pi^{i+1}\rangle_i"] & \Pi_iFH^{i+2}C \arrow[r, "\Pi_i\lambda_{H^{i+1}C}"] \arrow[rr, bend right, "\Pi_i\lambda^{i+1}_{HC}"'] & \Pi_iHFH^{i+1}C \arrow[r, "\Pi_iH\lambda^i_{HC}"] & \Pi_iH^{i+1}FHC \arrow[r, "\Pi_iH^{i+1}\varphi"] & \Pi_iH^{i+1}C
\end{tikzcd}$$

$$\begin{tikzcd}
FA \arrow[d, "Fg"'] & & & A \arrow[lll, "\alpha"'] \arrow[d, "g"] \\
F\Pi_iH^iC \arrow[r, "\langle Fi+1\rangle_i"'] \arrow[rrr, bend left, "\psi"] \arrow[d, "F\langle i+1\rangle_i"'] & \Pi_iFH^{i+1}C \arrow[r, "\Pi_i\lambda^i_{HC}"'] \arrow[d, "\langle\pi^{i+1}\rangle_i"] & \Pi_iH^iFHC \arrow[r, "\Pi_iH^i\varphi"'] \arrow[d, "\langle\pi^{i+1}\rangle_i"] & \Pi_iH^iC \arrow[d, "\langle\pi^{i+1}\rangle_i"] \\
F\Pi_iH^{i+1}C \arrow[r, "\langle F\pi^{i+1}\rangle_i"'] & \Pi_iFH^{i+2}C \arrow[r, "\Pi_i\lambda^{i+1}_{HC}"'] & \Pi_iH^{i+1}FHC \arrow[r, "\Pi_iH^{i+1}\varphi"'] & \Pi_iH^{i+1}C
\end{tikzcd}$$

Now the desired equality $f = \varphi \circ F(Hf \circ \jmath) \circ \alpha$ is witnessed by the commutativity of the outer square in the diagram

$$\begin{tikzcd}
FHA \arrow[d, "FHg"] \arrow[ddd, bend right=60, "FHf"'] & FA \arrow[l, "F\jmath"'] \arrow[d, "Fg"] & & & A \arrow[lll, "\alpha"'] \arrow[d, "g"'] \arrow[ddd, bend left=60, "f"] \\
 & F\Pi_i H^i C \arrow[rrr, "\psi"] \arrow[d, "F\langle\pi^{i+1}\rangle_i"'] & & & \Pi_i H^i C \arrow[d, equal] \\
FH\Pi_i H^i C \arrow[r, "F\langle H\pi^i\rangle_i"'] \arrow[d, "FH\pi^0"] & F\Pi_i H^{i+1} C \arrow[r, "\langle F\pi^i\rangle_i"'] \arrow[d, "F\pi^0"] & \Pi_i FH^{i+1} C \arrow[r, "\Pi_i\lambda^i_{HC}"'] \arrow[d, "\pi^0"] & \Pi_i H^i FHC \arrow[r, "\Pi_i H^i\varphi"'] \arrow[d, "\pi^0"] & \Pi_i H^i C \arrow[d, "\pi^0"'] \\
FHC \arrow[r, equal] & FHC \arrow[r, equal] & FHC \arrow[r, equal] & FHC \arrow[r, "\varphi"] & C
\end{tikzcd}$$

Proof of (ii): Suppose f' is a FH-coalgebra-to-algebra morphism from $(A, F\jmath \circ \alpha)$ to (C, φ). We observe that the commuting outer square in the following diagram proves that $g = \mathsf{fix}_{F,\alpha}(\psi) = \langle H^i f' \circ \jmath^i\rangle_i$.

$$\begin{tikzcd}
FA \arrow[dr, "F\langle\jmath^i\rangle_i"] \arrow[dd, "F\langle\jmath^i\rangle_i"'] \arrow[ddr, "F\langle\jmath^{i+1}\rangle_i"'] & & & & A \arrow[llll, "\alpha"'] \arrow[d, "\langle\jmath^i\rangle_i"] \\
 & F\Pi_i H^i A \arrow[r, "\langle F\pi^i\rangle_i"] \arrow[d, "F\Pi_i H^i\jmath"] & \Pi_i FH^i A \arrow[r, "\Pi_i\lambda^i_A"] & \Pi_i H^i FA \arrow[d, "\Pi_i H^i F\jmath"] & \Pi_i H^i A \arrow[l, "\Pi_i H^i\alpha"'] \arrow[dd, "\Pi_i H^i f'"] \\
F\Pi_i H^i A \arrow[r, "F\langle\pi^{i+1}\rangle_i"] \arrow[d, "F\Pi_i H^i f'"'] & F\Pi_i H^{i+1} A \arrow[r, "\langle F\pi^i\rangle_i"] \arrow[d, "F\Pi_i H^{i+1} f'"] & \Pi_i FH^{i+1} A \arrow[r, "\Pi_i\lambda^i_{HA}"] & \Pi_i H^i FHA \arrow[d, "\Pi_i H^i FHf'"] & \\
F\Pi_i H^i C \arrow[r, "F\langle\pi^{i+1}\rangle_i"] \arrow[rrrr, bend right=15, "\psi"'] & F\Pi_i H^{i+1} C \arrow[r, "\langle Fi\rangle_i"] & \Pi_i FH^{i+1} C \arrow[r, "\Pi_i\lambda^i_{HC}"] & \Pi_i H^i FHC \arrow[r, "\Pi_i H^i\varphi"] & \Pi_i H^i C
\end{tikzcd}$$

It follows that $f' = \pi^0 \circ \langle H^i f' \circ \jmath^i\rangle_i = \pi^0 \circ g = f$. □

The dual of Bartels's [5] original theorem becomes a corollary.

Corollary 29. *Assume $\mathcal{C}$ has countable products. Let $F : \mathcal{C} \to \mathcal{C}$ be a functor with an initial algebra and $H : \mathcal{C} \to \mathcal{C}$ a functor. If $\lambda : FH \dot{\to} HF$ is a natural transformation, then $(\mu F, F\mathsf{It}_F(H\mathsf{in}_F \circ \lambda_{\mu F}) \circ \mathsf{in}_F^{-1})$ is a recursive FH-coalgebra.*

Choosing $H = \mathsf{Id}$, we obtain the following instance, which was proved by Fokkinga [15, Section 4d] from the first principles as early as in 1992:

Corollary 30. *Assume $\mathcal{C}$ has countable products. Let $F : \mathcal{C} \to \mathcal{C}$ be a functor with an initial algebra. If $\lambda : F \dot{\to} F$ is a natural transformation, then $(\mu F, F\mathsf{It}_F(\mathsf{in}_F \circ \lambda_{\mu F}) \circ \mathsf{in}_F^{-1})$ is a recursive F-coalgebra.*

6. An example: dual of the Solution Theorem

As a non-trivial example of the use of the theorems we have presented above (principally of Theorem 19), we will now present a proof of the dual of the Solution Theorem of Moss [28] and Aczel et al. [2]. This section does not contain a "new" theorem. Instead, its purpose is to demonstrate that use of our theorems from the previous sections can make proving theorems like the Solution Theorem very easy (thus spelling out the details of the observation we made in an earlier paper [38]). A similar project – proving the Solution Theorem from the validity of an advanced structured corecursion scheme – has also been carried out by Jacobs [21]. His proof nevertheless became quite tedious. We give a very simple reduction to Theorem 19 in combination with the basic facts about recursive coalgebras given in Section 3.

We briefly recall the context of the Solution Theorem, including the Substitution Theorem, on which its statement and proof depend.

We are concerned with a functor $H : \mathcal{C} \to \mathcal{C}$ on a cocartesian category such that a final $K_X + H$-coalgebra exists for every object X of $\mathcal{C}$. (Recall that K_X denotes the functor which constantly returns X.) Let us write $(TX, [\,\eta, \tau\,])$ for the inverse of the final $K_X + H$-coalgebra. The Substitution Theorem says that, for any map $k : B \to TC$ there is a unique map $k^\bigstar : (TB, \tau_B) \to (TC, \tau_C)$ of H-coalgebras, such that $k = k^\bigstar \circ \eta_B$. Defining $\mu_X : TTX \to TX$ by $\mu_X = \mathsf{id}_{TX}{}^\bigstar$, one obtains that $\boldsymbol{T} = (T, \eta, \mu)$ is a monad and, moreover $(\boldsymbol{T}, HT, \tau, H\mu)$ is an ideal monad (see [27], Definition 3.3).

The Solution Theorem says that, for any map $\varphi : B \to C + HT(B + C)$ (guarded equation with unknowns B and parameters C), there is a unique map $f : B \to TC$ (solution) satisfying

$$\begin{tikzcd}
T(B+C) \arrow[d, "T[\,f{,}\eta_C\,]"'] & C + HT(B+C) \arrow[l, "{[\,\eta_{B+C} \circ \mathsf{inr}_{B,C}, \tau_{B+C}\,]}"'] & B \arrow[l, "\varphi"'] \arrow[d, "f"] \\
TTC \arrow[rr, "\mu_C"] & & TC
\end{tikzcd}$$

The Solution Theorem establishes that the ideal monad $(\boldsymbol{T}, HT, \theta, H\mu)$ is completely iterative. It is also cofree on H among the completely iterative monads, but this is a further theorem.

Dualizing the above account, for the co-Solution Theorem we must consider a functor $H : \mathcal{C} \to \mathcal{C}$ on a cartesian category such that an initial $K_X \times H$-algebra exists for every object X. Write $(DX, \langle\, \varepsilon_X, \theta_X \,\rangle)$ for the inverse of the initial $K_X \times H$-algebra. The co-Substitution Theorem (not proved here) tells us that, for any map $k : DB \to C$, there exists a unique map $k^\dagger : (DB, \theta_B) \to (DC, \theta_C)$ of H-coalgebras, such that $k = \varepsilon_C \circ k^\dagger$. Putting $\delta_X = \mathsf{id}_{DX}^\dagger$, we get that $\boldsymbol{D} = (D, \varepsilon, \delta)$ is a comonad. [This comonad is not to be confused with the cofree comonad over H which we discussed in Section 4 and which was obtained from the final $K_X \times H$-coalgebras!][4] Finally, we introduce the notation σ_X for the map $H\varepsilon_X \circ \theta_X : DX \to HX$.

These preparations done, we can formulate the co-Solution Theorem and give a proof. We do this in two steps, by first considering only a special case of the co-Solution Theorem (corresponding

[4] The ideal comonad $(\mathrm{D}, HD, \theta, H\delta)$ is cofree among the recursive comonads, but we will not delve into the topic of recursive comonads here and refrain from giving the definition. Of course they are the dual of completely iterative monads and the co-Solution Theorem states that $((\mathrm{D}, HD, \theta, H\delta)$ is recursive.

to the Solution Theorem without parameters) to then derive the general case (with parameters) as a corollary.

Proposition 31 (Dual of the Solution Theorem without parameters)**.** *Assume $\mathcal{C}$ is cartesian and let $H : \mathcal{C} \to \mathcal{C}$ be a functor such that an initial $K_X \times H$-algebra exists for every object X. Let D, ε, θ, δ, σ be as explained above. Then, for any B, the HD-coalgebra $(DB, H\delta_B \circ \theta_B)$ is recursive, i.e., for any HD-algebra (C, φ), there exists a unique map $f : DB \to C$ satisfying*

$$\begin{tikzcd}
HDDB \arrow[d, "HDf"'] & HDB \arrow[l, "H\delta_B"'] & DB \arrow[l, "\theta_B"'] \arrow[d, "f"] \\
HDC \arrow[rr, "\varphi"] & & C
\end{tikzcd}$$

or, which is equivalent (by δ_B being a coalgebra map and θ being natural),

$$\begin{tikzcd}
DDB \arrow[d, "Df"'] & & DB \arrow[ll, "\delta_B"'] \arrow[d, "f"] \\
DC \arrow[r, "\theta_C"] & HDC \arrow[r, "\varphi"] & C
\end{tikzcd}$$

Proof. We prove this proposition as an application of Theorem 19.

Fix some object B. Being the inverse of an initial $K_B \times H$-algebra, the $K_B \times H$-coalgebra $(DB, \langle \varepsilon_B, \theta_B \rangle)$ is recursive. By Proposition 10(b) hence also the H-coalgebra (DB, θ_B) is recursive. As explained above, $\boldsymbol{D} = (D, \varepsilon, \delta)$ is a comonad. From generalia about comonads, (DB, δ_B) is a $\boldsymbol{D}$-coalgebra.

Define now a natural transformation $\kappa : HD \dot{\to} DH$ by $\kappa_X = \langle \varepsilon_{HX}, \theta_{HX} \rangle^{-1} \circ \langle H\varepsilon_X, H(D\sigma_X \circ \delta_X) \rangle$. It is easy to verify that κ is a distributive law of H over $\boldsymbol{D}$. We also have that $D\theta_B \circ \delta_B = \kappa_{DB} \circ H\delta_B \circ \theta_B$ as witnessed by the commutativity of the outer square in the diagram

$$\begin{tikzcd}
HDB \arrow[ddd, "H\delta_B"'] \arrow[dr, "\langle \mathsf{id}_{HDB}, H\delta_B \rangle"] & & & & DB \arrow[llll, "\theta_B"'] \arrow[dl, "\langle \mathsf{id}_{DB}, \theta_B \rangle"'] \arrow[ddd, "\delta_B"] \\
& HDB \times HD^2B \arrow[r, equal] \arrow[d, "\mathsf{id}_{HDB} \times HD\delta_B"'] & HDB \times HD^2B \arrow[d, "\mathsf{id}_{HDB} \times HD\theta_B"] & DB \times HDB \arrow[l, "\theta_B \times H\delta_B"'] \arrow[d, "\mathsf{id}_{DB} \times H\delta_B"] & \\
& HDB \times HD^3B \arrow[r, "\mathsf{id}_{HDB} \times HD\sigma_{DB}"] & HDB \times (HD)^2B \arrow[dr, "\langle \varepsilon_{HDB}, \theta_{HDB} \rangle^{-1}"] & DB \times HD^2B \arrow[l, "\theta_B \times HD\theta_B"'] \arrow[dr, "\langle \varepsilon_{DB}, \theta_{DB} \rangle^{-1}"] & \\
HD^2B \arrow[ur, "\langle H\varepsilon_{DB}, H\delta_{DB} \rangle"] \arrow[rrr, "\kappa_{DB}"] & & & DHDB & D^2B \arrow[l, "D\theta_B"']
\end{tikzcd}$$

Theorem 19 (applied to $F = H$, $A = DB$, $D = D$, $\iota = \delta_B$, $\kappa = \kappa$) gives us that $(DB, H\delta_B \circ \theta_B)$ is a recursive HD-coalgebra. □

Corollary 32 (Dual of the full Solution Theorem with parameters). *Under the assumptions of the proposition, for any B, the $K_B \times HD(\mathsf{Id} \times K_B)$-coalgebra $(DB, \langle\, \varepsilon_B, H(D\langle\, \mathsf{id}_{DB}, \varepsilon_B \,\rangle \circ \delta_B) \circ \theta_B \,\rangle)$ is recursive, i.e., for any $K_B \times HD(\mathsf{Id} \times K_B)$-algebra (C, φ), there exists a unique map $f : DB \to C$ satisfying*

$$\begin{tikzcd}
B \times HD(DB \times B) \arrow[d, "\mathsf{id}_B \times HD(f \times \mathsf{id}_B)"] & & B \times HDB \arrow[ll, "\mathsf{id}_B \times H(D\langle\, \mathsf{id}_{DB}, \varepsilon_B \,\rangle \circ \delta_B)"'] & DB \arrow[l, "\langle\, \varepsilon_B, \theta_B \,\rangle"'] \arrow[d, "f"] \\
B \times HD(C \times B) \arrow[rrr, "\varphi"] & & & C
\end{tikzcd}$$

*or, which is equivalent (by **D** being a comonad, δ_B being a coalgebra map and θ being natural),*

$$\begin{tikzcd}
DDB \arrow[d, "D\langle\, f, \varepsilon_B \,\rangle"] & & DB \arrow[ll, "\delta_B"'] \arrow[d, "f"] \\
D(C \times B) \arrow[r, "\langle\, \mathsf{snd}_{C,B} \circ \varepsilon_{C \times B}, \theta_{C \times B} \,\rangle"] & B \times HD(C \times B) \arrow[r, "\varphi"] & C
\end{tikzcd}$$

Proof. This result can be proved as a direct application of Theorem 19 by appropriately adapting (scaling up) the proof of Proposition 31, but this is relatively tedious because of the extensive calculations with the product laws it takes; nevertheless, Jacobs [21] has essentially spelled out the dual proof. Instead, we will combine Proposition 31 as it stands with some of the basic facts about recursive coalgebras discussed in Section 3.

Fix an object B. Write H^B for the functor $K_B \times H$, write $(D^B, \varepsilon^B, \delta^B)$ for the monad given by the initial $K_X \times H^B$-algebras (they must exist as an object carrying an initial $K_{X \times B} \times H$-algebra also carries an initial $K_X \times H^B$-algebra).

Now clearly there is an isomorphism

$$i : DB = \mu(K_B \times H) \cong \mu(K_1 \times (K_B \times H)) = D^B 1.$$

Also there is a natural isomorphism

$$\begin{aligned} j : K_B \times HD(\mathsf{Id} \times K_B) &= K_B \times H\mu((\mathsf{Id} \times K_B) \times H) \\ &\overset{\cdot}{\cong} K_B \times H\mu(\mathsf{Id} \times (K_B \times H)) = H^B D^B. \end{aligned}$$

Also it is straightforward that the polygon

$$\begin{tikzcd}
H^B D^B DB \arrow[d, "H^B D^B i"] & B \times HD(DB \times B) \arrow[l, "j_{DB}"'] & B \times HDB \arrow[l, "\mathsf{id}_B \times H(D\langle\, \mathsf{id}_{DB}, \varepsilon_B \,\rangle \circ \delta_B)"'] & DB \arrow[l, "\langle\, \varepsilon_B, \theta_B \,\rangle"'] \arrow[d, "i"] \\
H^B D^B D^B 1 & & H^B D^B 1 \arrow[ll, "H^B \delta^B_1"'] & D^B 1 \arrow[l, "\theta^B_1"']
\end{tikzcd}$$

commutes.

By the proposition we know that $(D^B 1, H^B \delta^B_1 \circ \theta^B_1)$ is a recursive $H^B D^B$-coalgebra. But from this by Proposition 8 it follows that $(DB, j_{DB} \circ \langle\, \varepsilon_B, H(D\langle\, \mathsf{id}_{DB}, \varepsilon_B \,\rangle \circ \delta_B) \circ \theta_B \,\rangle)$ is also a recursive

$H^B D^B$-coalgebra. By Proposition 10(b) therefore $(DB, \langle \varepsilon_B, H(D\langle \mathsf{id}_{DB}, \varepsilon_B \rangle \circ \delta_B) \circ \theta_B \rangle)$ is a recursive $K_B \times HD(\mathsf{Id} \times K_B)$-coalgebra. □

While the Substitution Theorem and Solution Theorem have a natural interpretation for **Set** and polynomial functors in terms of non-wellfounded trees with graft-points and grafting (non-wellfounded terms and substitution), the dual theorems have a similar interpretation in terms of decorated wellfounded trees and redecoration. We have pointed out the programming relevance of the resulting combinators in [38] where we also discussed the proofs of the co-Substitution and co-Solution Theorems from Corollary 20.

7. Conclusions and future work

We have motivated the relevance of recursive functor-coalgebras for programming: the recursiveness of the coalgebra appearing in a structured general-recursion equation is a sufficient condition for its solvability. Since there is no practical general method for checking whether a given coalgebra is recursive, one should strive for useful sufficient conditions. We have shown how to use comonads, comonad-coalgebras and distributive laws to construct new recursive coalgebras from coalgebras already known to be recursive. These results provide a generalization and modularization of the proofs of the results of [39] on structured recursion schemes for initial algebras. By duality, they also generalize the dual results of [5,8].

This paper reports our first results on recursive coalgebras and many of our questions are unanswered yet. Apart from checking whether the theorems of Sections 4 and 5 can be strengthened in further useful ways, we will take a closer look at wellfounded induction. We have already identified weak conditions guaranteeing implications between recursiveness, Taylor's wellfoundedness and various intermediate properties. Finally, we are interested in seeing if the results admit any useful type-theoretic versions. One might wish to be able to turn the structured general recursion scheme of a recursive coalgebra into a reduction rule in a typed lambda calculus without giving rise to non-terminating reduction sequences of welltyped terms. The questions are when this is possible and how to accomplish it.

Acknowledgment

We are thankful to our two anonymous referees for a number of very useful suggestions and remarks.

References

[1] J. Adámek, D. Lücke, S. Milius, Recursive coalgebras of finitary functors, in: P. Mosses, J. Power, M. Seisenberger (Eds.), Selected Papers from the CALCO 2005 Young Researchers Workshop, CALCO-jnr 2005 (Swansea, September 2005), Tech. Report CSR 18-2005, Department of Computer Science, University of Wales Swansea, 2005, pp. 1–14.

[2] P. Aczel, J. Adámek, S. Milius, J. Velebil, Infinite trees and completely iterative theories: a coalgebraic view, Theoret. Comput. Sci. 300 (1–3) (2003) 1–45.

[3] J. Adámek, S. Milius, J. Velebil, Free iterative theories: A coalgebraic view, Math. Struct. Comput. Sci. 13 (2) (2003) 259–320.

[4] J. Adámek, S. Milius, J. Velebil, From iterative algebras to iterative theories (extended abstract), in: J. Adámek, S. Milius (Eds.), Proceedings of the 7th Workshop on Coalgebraic Methods in Computer Science, CMCS'04 (Barcelona, March 2004), Electron. Notes in Theoret. Comput. Sci., vol. 106, Elsevier, Amsterdam, 2004, pp. 3–24.

[5] F. Bartels, Generalised coinduction, Math. Struct. Comput. Sci. 13 (2) (2003) 321–348.

[6] A. Bove, V. Capretta, Nested general recursion and partiality in type theory, in: R.J. Boulton, P.B. Jackson (Eds.), Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2001 (Edinburgh, September 2001), Lecture Notes in Comput. Sci., vol. 2152, Springer-Verlag, Berlin, 2001, pp. 121–135.

[7] A. Bove, V. Capretta, Modelling general recursion in type theory, Math. Struct. Comput. Sci. 15 (4) (2005) 671–708.

[8] D. Cancila, F. Honsell, M. Lenisa, Generalized coiteration schemata, in: H.P. Gumm (Ed.), Proceedings of the 6th Workshop on Coalgebraic Methods in Computer Science, CMCS'03 (Warsaw, Apr. 2003), vol. 82, No. 1, Electron. Notes in Theoret. Comput. Sci., Elsevier, Amsterdam, 2003.

[9] R. Cockett, T. Fukushima, About Charity, Yellow Series Report 92/480/18, Department of Computer Science, University of Calgary, 1992.

[10] H. Doornbos, R. Backhouse, Mathematics of recursive program construction, draft manuscript (July 2001).

[11] A. Eppendahl, Coalgebra-to-algebra morphisms, in: M. Hofmann, G. Rosolini, D. Pavlović (Eds.), Proceedings of the 8th International Conference on Category Theory and Computer Science, CTCS'99 (Edinburgh, September 1999), Electron. Notes in Theoret. Comput. Sci., vol. 29, Elsevier, Amsterdam, 1999.

[12] A. Eppendahl, Fixed point objects corresponding to Freyd algebras, manuscript (May 2000).

[13] M. Escardó, A.K. Simpson, A universal characterization of the closed euclidean interval, in: Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science, LICS'01 (Boston, June 2001), IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 115–128.

[14] P.J Freyd, Algebraically complete categories, in: A. Carboni, M.C. Pedicchio, G. Rosolini (Eds.), Proceedings of the International Conference on Category Theory '90, CT'90 (Como, July 1990), Lecture Notes in Mathematics, vol. 1488, Springer-Verlag, Berlin, 1991, pp. 95–104.

[15] M.M. Fokkinga, Law and Order in Algorithmics, PhD thesis, Department of Informatics, University of Twente, 1992.

[16] P.J. Freyd, Recursive types reduced to inductive types, in: Proceedings of the 5th IEEE Annual Symposium on Logic in Computer Science, LICS'90 (Philadelphia, PA, June 1990), IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 498–507.

[17] P.J. Freyd, Remarks on algebraically compact categories, in: M.P. Fourman, P.T. Johnstone, A.M. Pitts (Eds.), Applications of Categories in Computer Science, LMS Lecture Note Series, vol. 177, Cambridge University Press, Cambridge, 1992, pp. 95–106.

[18] E. Giménez, Codifying guarded definitions with recursion schemes, in: P. Dybjer, B. Nördstrom (Eds.), Selected Papers from 2nd International Workshop on Types for Proofs and Programs, TYPES'94 (Båstad, June 1994), Lecture Notes in Computer Science, vol. 996, Springer-Verlag, Berlin, 1995, pp. 39–59.

[19] E. Giménez, Structural recursive definitions in type theory, in: K.G. Larsen, S. Skyum, G. Winskel (Eds.), Proceedings of the 25th International Colloquium on Automata, Languages and Programming, ICALP'98 (Aalborg, July 1998), Lecture Notes in Computer Science, vol. 1443, Springer-Verlag, Berlin, 1998, pp. 397–408.

[20] T. Hagino, A typed lambda calculus with categorical type constructors, in: D.H. Pitt, A. Poigné, D.E. Rydeheard (Eds.), Proceedings of the 2nd International Conference on Category Theory and Computer Science, CTCS'87 (Edinburgh, September 1987), Lecture Notes in Computer Science, vol. 283, Springer-Velrag, Berlin, 1987, pp. 140–157.

[21] B. Jacobs, Relating two approaches to coinductive solution of recursive equations, in: J. Adámek, S. Milius (Eds.), Proceedings of the 7th Workshop on Coalgebraic Methods in Computer Science, CMCS'04 (Barcelona, March 2004), Electron. Notes in Theoret. Comput. Sci., vol. 106, Elsevier, Amsterdam, 2004, pp. 145–166.

[22] J. Lambek, A fixpoint theorem for complete categories, Math. Z. 103 (1968) 151–161.

[23] G. Malcolm, Data structures and program transformation, Sci. Comput. Program. 14 (2–3) (1990) 255–279.

[24] C. McBride, J. McKinna, The view from the left, J. Funct. Prog. 14 (1) (2004) 69–111.

[25] C. McBride, Epigram: practical programming with dependent types, in: V. Vene, T. Uustalu (Eds.), Revised Lectures from the 5th International School on Advanced Functional Programming, AFP 2004 (Tartu, Aug. 2004), Lecture Notes in Computer Science, vol. 3622, Springer-Verlag, Berlin, 2005, pp. 130–170.

[26] E. Meijer, M. Fokkinga, R. Paterson, Functional programming with bananas, lenses, envelopes and barbed wire, in: J. Hughes (Ed.), Proceedings of the 5th ACM Conference on Functional Programming Languages and Computer Architecture, FPCA'91 (Cambridge, MA, Aug. 1991), Lecture Notes in Computer Science, vol. 523, Springer-Verlag, Berlin, 1991, pp. 124–144.

[27] S. Milius, Completely iterative algebras and completely iterative monads, Inform. Comput. 196 (1) (2005) 1–41.

[28] L.S. Moss, Parametric corecursion, Theoret. Comput. Sci. 260 (1–2) (2001) 139–163.

[29] E. Nelson, Iterative algebras, Theoret. Comput. Sci. 25 (1983) 67–94.

[30] G. Osius, Categorical set theory: A characterisation of the category of sets, J. Pure Appl. Algebra 4 (1974) 79–119.

[31] P. Taylor, Intuitionistic sets and ordinals, J. Symb. Logic 61 (3) (1996) 705–744.

[32] P. Taylor, Towards a unified treatment of induction, I: the general recursion theorem, unfinished draft manuscript (Aug. 1996).

[33] P. Taylor, Practical Foundations of Mathematics, Cambridge Studies in Advanced Mathematics, Cambridge University Press, Cambridge, 1999.

[34] D. Turi, G.D. Plotkin, Towards a mathematical operational semantics, in: Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS'97 (Warsaw, June/July 1997), IEEE Computer Science Press, Los Alamitos, CA, 1997, pp. 280–291.

[35] D.A. Turner, Elementary strong functional programming, in: P.H. Hartel, R. Plasmeijer (Eds.), Proceedings of the 1st International Symposium on Functional Programming Language in Education, FPLE'95 (Nijmegen, December 1995), Lecture Notes in Computer Science, vol. 1022, Springer-Verlag, Berlin, 1995, pp. 1–13.

[36] T. Uustalu, V. Vene, Primitive (co)recursion and course-of-value (co)iteration, categorically, Informatica 10 (1) (1999) 5–26.

[37] T. Uustalu, V. Vene, Coding recursion à la Mendler (extended abstract), in: J. Jeuring (Ed.), Proceedings of the 2nd Workshop on Generic Programming, WGP'2000 (Ponte de Lima, July 2000), Tech. Report UU-CS-2000-19, Department of Computer Science, Utrecht University, 2000, pp. 69–85.

[38] T. Uustalu, V. Vene, The dual of substitution is redecoration, in: K. Hammond, S. Curtis (Eds.), Trends in Functional Programming, vol. 3, Intellect, Bristol/Portland, OR 2002, pp. 99–110.

[39] T. Uustalu, V. Vene, A. Pardo, Recursion schemes from comonads, Nordic J. Comput. 8 (3) (2001) 366–390.