Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 106 (2004) 43–61
www.elsevier.com/locate/entcs

Recursive Coalgebras from Comonads ⋆

Venanzio Capretta[1,2]

Dept. of Mathematics and Statistics, University of Ottawa
585 King Edward Ave., Ottawa, Ont. K1N 6N5, Canada

Tarmo Uustalu[3]

Institute of Cybernetics at Tallinn Technical University
Akadeemia tee 21, EE-12618 Tallinn, Estonia

Varmo Vene[4]

Dept. of Computer Science, University of Tartu
J. Liivi 2, EE-50409 Tartu, Estonia

Abstract

We discuss Osius's [22] concept of a recursive coalgebra of a functor from the perspective of programming semantics and give some new sufficient conditions for the recursiveness of a functor-coalgebra that are based on comonads, comonad-coalgebras and distributive laws.

Keywords: Recursive coalgebra, monad, functor-coalgebra

⋆ Research supported by the Estonian-French scientific cooperation programme Parrot and by the Estonian Science Foundation under grant No. 5567.

[1] Previously at INRIA Sophia Antipolis, 2004 route des Lucioles, BP 93, F-06902 Sophia Antipolis, France.

[2] Email: vcapr396@science.uottawa.ca

[3] Email: tarmo@cs.ioc.ee

[4] Email: varmo@cs.ut.ee

doi:10.1016/j.entcs.2004.02.034

1 Introduction

This paper is dedicated to the study of recursive functor-coalgebras. In the sense of [22], 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

$$f = \varphi \circ F f \circ \alpha \quad (*)$$

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

Our prime interest in recursive coalgebras comes from their application to programming semantics. In 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

$$f = \Phi(f) \quad (**)$$

as a function definition. The problem is that, for arbitrary givens, 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 solutions or multiple solutions among which there is no most preferable solution. But for more specific givens, the equation may indeed be predestined to have exactly one solution (or at least one solution, but among them a canonical one) and in this case it is really meaningful to see it as a definition.

For (*), which is a structured instance of (**), one of the ways to know that it properly defines a morphism 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: (a) 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. (b) 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 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. The set $\mathsf{List}Z$ equipped with the analysis of every non-empty, non-singleton list into two halves is a recursive coalgebra of the functor $\mathsf{BT}_Z = K_1 + K_Z + \mathsf{Id} \times \mathsf{Id}$. Etc. (c) A functor may well have recursive coalgebras without having an initial algebra. E.g., a set with a relation on it carries a recursive coalgebra of the powerset functor iff the relation is wellfounded.

In this paper, we present some motivation for the use of recursive coalgebras as a paradigm of structured recursion in programming semantics, present the basic theory of recursive coalgebras and, centrally, give some new conditions for the recursiveness of a coalgebra based on comonads, comonad-coalgebras and distributive laws. The latter results are a generalization of our results in [27] on structured recursion schemes for initial algebras and, modulo the duality, the dual results in [4,7] on structured corecursion schemes for final coalgebras.

Related work Recursive 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 [22] 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 [23,24,25] took Osius's ideas further, showing that the general recursion theorem holds for any functor on **Set** preserving monos and inverse image diagrams. Eppendahl [9,10] studied recursive (a.k.a. algebra-initial) coalgebras with the objective of obtaining an explanation to Freyd's [12,13,14] transposition of invariant objects.

The dual concept of a corecursive (a.k.a. coalgebra-final, iterative) algebra was used by Escardó and Simpson [11] to provide a universal characterization of the closed euclidean interval. The newest work by Adámek, Milius and Velebil [19,3] 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 [21]).

Structured recursion schemes for initial algebras have been studied by the authors [27] and the dual schemes for final coalgebras by Bartels [4] and Cancila, Honsell and Lenisa [7]. To functional programming, the structured general recursion scheme was first introduced by Meijer, Fokkinga and Paterson [18] who called it the hylo scheme. Doornbos and Backhouse [8] have asked the question under what conditions the hylo diagram has a unique solution. In type theory, structured (co)recursion schemes for initial algebras (final coalgebras) have been studied by, e.g., Giménez [15,16] and (co)recursion more generally by, e.g., Bove and Capretta [5,6] and McBride and McKinna [17].

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. In Section 5, 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 $\leq$.

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

where $l_{\leq x} = [y \leftarrow l \mid y \leq x]$ 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 $\mathsf{qmerge} \circ \mathsf{BT}\ qsort \circ \mathsf{qsplit}$ where $\mathsf{BT}_Z\ X = 1 + Z \times X \times X$. The first morphism qsplit of the composition determines the arguments for the recursive calls; $(\mathsf{List}Z, \mathsf{qsplit})$ is a BT_Z-coalgebra:

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

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

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

The equation $qsort = \mathsf{qmerge} \circ \mathsf{BT}\,qsort \circ \mathsf{qsplit}$ is meaningful as a definition since it determines a unique function. The reason is that the arguments of the recursive calls are always strictly shorter than that of the main call—a property of the coalgebra $(\mathsf{List}Z, \mathsf{qsplit})$. The equation remains uniquely solvable also, if we replace $(\mathsf{List}Z, \mathsf{qmerge})$ with some other F-algebra (C, φ): we may say that $(\mathsf{List}Z, \mathsf{qsplit})$ is recursive.

Abstracting away the concrete data of the above example, we are led to the following definition.

Definition 2.1 (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*

$$\begin{array}{ccc} FA & \xleftarrow{\ \alpha\ } & A \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ FC & \xrightarrow{\ \varphi\ } & C \end{array}$$

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

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 *hylo diagram* [18]. The recursion scheme used—*hylo scheme*—says that, if F has an initial algebra whose inverse is its final coalgebra (which happens if $\mathcal{C}$ is algebraically compact), then the post-composition of the initial algebra morphism to (C, φ) with the final coalgebra morphism from (A, α) (the hylomorphism) is a coalgebra-to-algebra morphism from (A, α) to (C, φ). The hylomorphism is not necessarily a unique solution of the hylo diagram, just a canonical one.

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.

3 Recursive coalgebras: basic constructions

As exemplified by the last example (determining the wellfoundedness of a decidable relation on natural numbers is undecidable), it can be hard to determine whether a coalgebra of a given functor F is recursive. 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. 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)).

Proposition 3.1 *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, notice that the unique coalgebra-to-algebra morphism from a recursive F-coalgebra (A, α) to $(\mu F, \mathsf{in}_F)$ is also a unique coalgebra morphism from (A, α) to $(\mu F, \mathsf{in}_F^{-1})$. □

Corollary 3.2 *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:*

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

Proposition 3.3 *Let (A, α) be 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.*

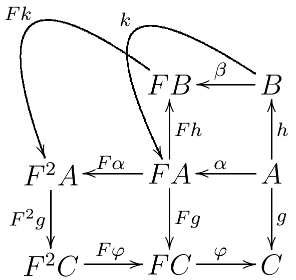
Proof. (if) Let the postinverse of $\alpha : A \to FA$ be $\alpha^- : FA \to A$. Then $h = \mathsf{It}_F(\,\alpha^-\,) : \mu F \to A$ is a postinverse of $m : A \to \mu F$: indeed, we have $h \circ m = h \circ \mathsf{in}_F \circ Fm \circ \alpha = \alpha^- \circ F(h \circ m) \circ \alpha$, but we also have $\mathsf{id}_A = \alpha^- \circ F\mathsf{id}_A \circ \alpha$, hence $h \circ m = \mathsf{fix}_{F,\alpha}(\alpha^-) = \mathsf{id}_A$.

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

Here is the first proposition useable to reduce the question of recursiveness of one coalgebra to that of some other, related coalgebra.

Proposition 3.4 *Let (A, α) be 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.



Let $f = \varphi \circ Fg \circ k : B \to C$. We show that $\mathsf{fix}_{F,\beta}(\varphi) = f$. We have $f = \varphi \circ Fg \circ k = \varphi \circ F(\varphi \circ Fg \circ \alpha) \circ k = \varphi \circ F(\varphi \circ Fg \circ k) \circ \beta = \varphi \circ Ff \circ \beta$, 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 $f' \circ h = \varphi \circ Ff' \circ \beta \circ h = \varphi \circ F(f' \circ h) \circ \alpha$, which implies $f' \circ h = \mathsf{fix}_{F,\alpha}(\varphi) = g$. 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 Prop. 3.4. First, recursive F-coalgebras are preserved by F.

Proposition 3.5 *If (A, α) is a recursive F-coalgebra, then $(FA, F\alpha)$ is also a recursive F-coalgebra.*

Proof. From Prop. 3.4 for $h = \alpha$ and $k = \mathsf{id}_{FA}$. □

The implication of Prop. 3.1 can be turned around.

Proposition 3.6 *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) to (C, φ).

(b) By Prop. 3.5, 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:

$$\begin{array}{ccc} FA & \xleftarrow{\alpha} & A \\ {\scriptstyle F\alpha}\downarrow & & \downarrow{\scriptstyle \alpha} \\ F^2A & \xleftarrow{F\alpha} & FA \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle h} \\ FA & \xleftarrow{\alpha} & A \end{array}$$

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 for any category that a subcoalgebra of a recursive coalgebra is recursive. But the following weaker statement is always true.

Proposition 3.7 *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 be the postinverse of m. (a) Let $k = \alpha \circ m$. Then h is trivially a coalgebra morphism and k is a coalgebra morphism as $F\alpha \circ k = F\alpha \circ \alpha \circ m = F(\alpha \circ m) \circ \beta = Fk \circ \beta$. Furthermore, $\beta = \beta \circ h \circ m = Fh \circ \alpha \circ m = Fh \circ k$. By Prop. 3.4, (B, β) is recursive.

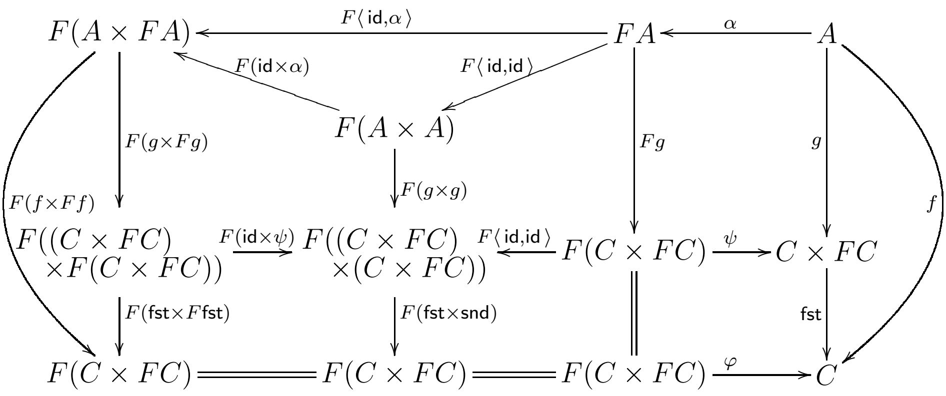
(b) Let α^- be the postinverse of α. Then $\beta^- = h \circ \alpha^- \circ Fm$ is a postinverse of β, since $\beta^- \circ \beta = h \circ \alpha^- \circ Fm \circ \beta = h \circ \alpha^- \circ \alpha \circ m = h \circ m = \mathsf{id}_B$. □

Here is another useful proposition, with a relatively involved proof. In the next section, we shall see that, under an extra assumption, this proposition is an instance of a more general theorem.

Proposition 3.8 *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 commutation of the outer square in the diagram



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

The following two transposition propositions appeared in Eppendahl [9,10].

Proposition 3.9 *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-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 $\varphi \circ Gf \circ \tau_A \circ \alpha = \varphi \circ \tau_C \circ Ff \circ \alpha$.

(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 3.10 *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 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 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 $(g^\dagger)^\star = g$.

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

The following proposition builds on Props. 3.9, 3.10.

Proposition 3.11 *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 $\tau : LF \dot{\to} GL$ 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 $\eta : \mathsf{Id} \dot{\to} RL$ and $\varepsilon : LR \dot{\to} \mathsf{Id}$ the unit resp. counit of the adjunction. Let $\lambda(\cdot)$ denote the natural bijection between the homsets $\mathcal{C}(L-, =)$ and $\mathcal{C}(-, R=)$. Now, let $\beta = \lambda(\tau_A \circ L\alpha) = R(\tau_A \circ L\alpha) \circ \eta_A = R\tau_A \circ \eta_{FA} \circ \alpha : A \to RGLA$.

According to Prop. 3.9, the RGL-coalgebra (A, β) is recursive. But then by Prop. 3.10, the LRG-coalgebra $(LA, L\beta)$ is recursive. By Prop. 3.9 once more, the G-coalgebra $(LA, \varepsilon_{GLA} \circ L\beta)$ is recursive. But $\varepsilon_{GLA} \circ L\beta = \lambda^{-1}(\beta) = \lambda^{-1}(\lambda(\tau_A \circ L\alpha)) = \tau_A \circ L\alpha$. □

We conclude this section by briefly looking at two useful strengthenings of the notion of recursiveness, which we call strong recursiveness and (for the time being, for the lack of a better name) very recursiveness. Strong recursiveness relates to recursiveness for coalgebras as allowing strong iteration (iteration with parameters) relates to allowing iteration (i.e., initiality) for algebras.

Definition 3.12 (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*

$$\begin{array}{ccccc} F(\Gamma \times A) & \xleftarrow{\sigma_{\Gamma,A}} & \Gamma \times FA & \xleftarrow{\mathsf{id}_\Gamma \times \alpha} & \Gamma \times A \\ {\scriptstyle Ff}\downarrow & & & & \downarrow{\scriptstyle f} \\ FC & & \xrightarrow{\quad\varphi\quad} & & C \end{array}$$

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 Prop. 3.11 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.

Very recursiveness is roughly in the same position wrt. recursiveness for coalgebras as allowing primitive recursion is wrt. initiality for algebras. The new work of Adámek, Milius and Velebil [19,3] on the free completely iterative (resp. iterative) monad of a functor (elaborating on their original approach in [1,2]) is centered around the dual concept (resp. a finitary version of it).

Definition 3.13 (very recursive coalgebra) *Let $\mathcal{C}$ be cartesian and $F : \mathcal{C} \to \mathcal{C}$ a functor. An F-coalgebra (A, α) is* very recursive *iff, for any $(K_A \times F)$-algebra (C, φ), there is a unique morphism $f : A \to C$, denoted $\mathsf{vfix}_{A,\alpha}(\varphi)$, satisfying*

$$\begin{array}{ccc} A \times FA & \xleftarrow{\langle \mathsf{id}_A, \alpha \rangle} & A \\ {\scriptstyle \mathsf{id}_A \times Ff}\downarrow & & \downarrow{\scriptstyle f} \\ A \times FC & \xrightarrow{\ \varphi\ } & C \end{array}$$

An F-coalgebra (A, α) is very recursive iff the $(K_A \times F)$-coalgebra $(A, \langle\, \mathsf{id}_A, \alpha \,\rangle)$ is recursive. A very recursive F-coalgebra (A, α) is necessarily recursive: for an F-algebra (C, φ), $\mathsf{fix}_{F,\alpha}(\varphi) = \mathsf{vfix}_{F,\alpha}(\varphi \circ \mathsf{snd}_{A,FC})$. But not every recursive coalgebra is very recursive.

The concept of very recursive coalgebras and its dual are elegant and useful because of the following fact whose dual is central in [19].

Proposition 3.14 *For any object X, an object DX is the carrier of a cofree very recursive F-coalgebra over X iff DX is the carrier of an initial $(K_X \times F)$-algebra.*

With 'very 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.

4 Recursive coalgebras from 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 4.1 (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,*

$$\begin{array}{ccc} DX & \xrightarrow{\delta_X} & D^2X \\ {\scriptstyle\delta_X}\downarrow & \diagdown\!\!\diagdown & \downarrow{\scriptstyle\varepsilon_{DX}} \\ D^2X & \xrightarrow{D\varepsilon_X} & DX \end{array} \qquad \begin{array}{ccc} DX & \xrightarrow{\delta_X} & D^2X \\ {\scriptstyle\delta_X}\downarrow & & \downarrow{\scriptstyle\delta_{DX}} \\ D^2X & \xrightarrow{D\delta_X} & D^3X \end{array}$$

Definition 4.2 (coalgebra of a comonad) *A* coalgebra *of a* comonad *(D, ε, δ) on $\mathcal{C}$ is a coalgebra $(A, \imath)$ of the functor D satisfying*

$$\begin{array}{ccc} A & \xrightarrow{\imath} & DA \\ & \diagdown\!\!\diagdown & \downarrow{\scriptstyle\varepsilon_A} \\ & & A \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{\imath} & DA \\ {\scriptstyle\imath}\downarrow & & \downarrow{\scriptstyle\delta_A} \\ DA & \xrightarrow{D\imath} & D^2A \end{array}$$

Definition 4.3 (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,

$$
\begin{array}{ccc}
FDX & \xrightarrow{\kappa_X} & DFX \\
{\scriptstyle F\varepsilon_X}\downarrow & & \downarrow{\scriptstyle \varepsilon_{FX}} \\
FX & = & FX
\end{array}
\qquad
\begin{array}{ccccc}
FDX & & \xrightarrow{\kappa_X} & & DFX \\
{\scriptstyle F\delta_X}\downarrow & & & & \downarrow{\scriptstyle \delta_{FX}} \\
FD^2X & \xrightarrow{\kappa_{DX}} & DFDX & \xrightarrow{D\kappa_X} & D^2FX
\end{array}
$$

We present three theorems, each saying that a coalgebra constructed in a certain fashion from a coalgebra known to be recursive is recursive as well. We begin by the main theorem, which uses a general comonad.

Theorem 4.4 *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A, α) a recursive F-coalgebra, $\boldsymbol{D} = (D, \varepsilon, \delta)$ a comonad on $\mathcal{C}$ and $(A, \imath)$ a $\boldsymbol{D}$-coalgebra. If κ is a distributive law of F over $\boldsymbol{D}$ satisfying*

$$
\begin{array}{ccccc}
FA & & \xleftarrow{\alpha} & & A \\
{\scriptstyle F\imath}\downarrow & & & & \downarrow{\scriptstyle \imath} \\
FDA & \xrightarrow{\kappa_A} & DFA & \xleftarrow{D\alpha} & DA
\end{array}
\qquad (*)
$$

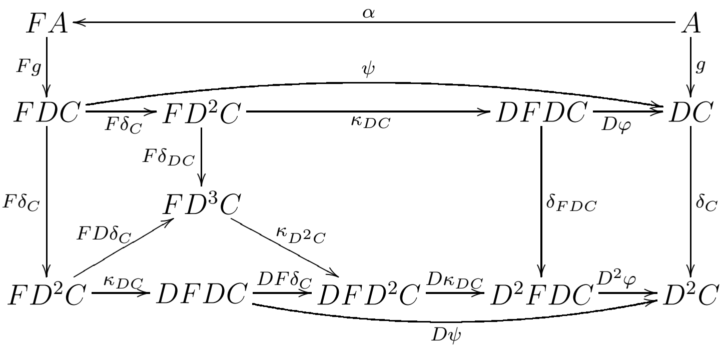
then $(A, F\imath \circ \alpha)$ is a recursive FD-coalgebra (and, consequently, by Prop. 3.9, $(A, D\alpha \circ \imath)$ is a recursive DF-coalgebra).

It might make sense to define that the data $(A, \alpha, \imath)$ form, say, a *dicoalgebra* of $(F, \boldsymbol{D}, \kappa)$ iff they meet the condition (*) and to then develop a theory of functor-comonad-dicoalgebras (cf. the functor-functor-bialgebras of Turi and Plotkin [26] or the monad-functor-bialgebras of [4,7]), but we have chosen not to specifically pursue this line here, as we will not need many properties of dicoalgebras.

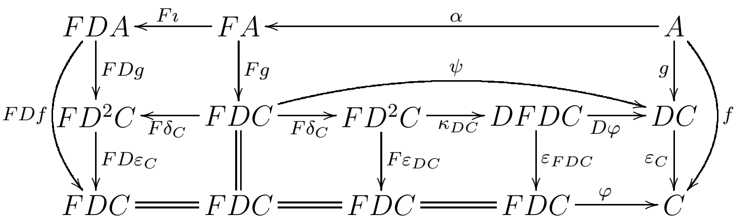
Proof. Consider any FD-algebra (C, φ). Let $\psi = D\varphi \circ \kappa_{DC} \circ F\delta_C : FDC \to DC$, $g = \mathsf{fix}_{F,\alpha}(\psi) : A \to DC$ and $f = \varepsilon_C \circ g : A \to C$. We show that (i) f is a FD-coalgebra-to-algebra morphism from $(A, F\imath \circ \alpha)$ to (C, φ) and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

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

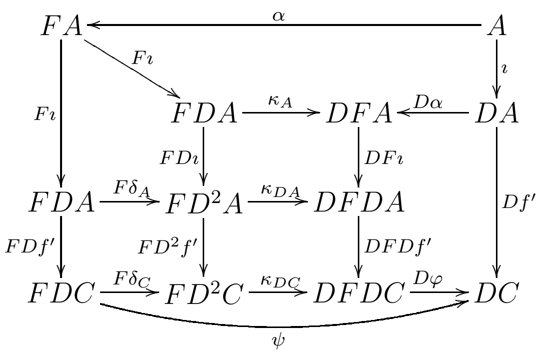
$$
\begin{array}{ccccc}
FA & & \xleftarrow{\alpha} & & A \\
{\scriptstyle F\imath}\downarrow & & & & \downarrow{\scriptstyle \imath} \\
FDA & \xrightarrow{\kappa_A} & DFA & \xleftarrow{D\alpha} & DA \\
{\scriptstyle FDg}\downarrow & & \downarrow{\scriptstyle DFg} & & \downarrow{\scriptstyle Dg} \\
FD^2C & \xrightarrow{\kappa_{DC}} & DFDC & \xrightarrow{D\psi} & D^2C
\end{array}
$$



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



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



It follows that $f' = f' \circ \varepsilon_A \circ \imath = \varepsilon_C \circ Df' \circ \imath = \varepsilon_C \circ g = f$. □

Theorem 4.4 provides a powerful generalization of the central theorem in [27], which was on structured recursion schemes for initial algebras derivable from comonads (cf. also the dual result stated in [4,7]; we note that in [28], the substitution and solution theorems of [20,1] were proved from this result). Indeed, the theorem of [27] is just a special case of Theorem 4.4 now.

Corollary 4.5 *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor with an initial algebra and $\boldsymbol{D} = (D, \varepsilon, \delta)$ a comonad on $\mathcal{C}$. If κ is a distributive law of F over $\boldsymbol{D}$, then $(\mu F, F\mathsf{lt}_F(\, D\mathsf{in}_F \circ \kappa_{\mu F}\,) \circ \mathsf{in}_F^{-1})$ is a recursive FD-coalgebra.*

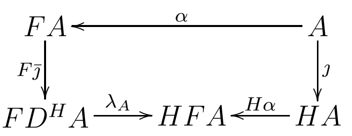
Proof. It is easy to check that $(\mu F, \mathsf{lt}_F(\, D\mathsf{in}_F \circ \kappa_{\mu F}\,))$ is a $\boldsymbol{D}$-coalgebra. It is also immediate that it relates appropriately to the recursive F-coalgebra $(\mu F, \mathsf{in}_F^{-1})$ via κ. Hence, by Theorem 4.4, $(\mu F, F\mathsf{lt}_F(\, D\mathsf{in}_F \circ \kappa_{\mu F}\,) \circ \mathsf{in}_F^{-1})$ is a recursive FD-coalgebra. □

We learn that the result in [27] 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.

A useful class of comonads are comonads cofree over a functor. For a functor H which has a cofree comonad, let us agree to write $\boldsymbol{D}^H = (D^H, \varepsilon^H, \delta^H)$ for this comonad and σ^H for the extraction of H from D^H. We recall the well known fact that D^H sends an object X to the carrier of a cofree H-coalgebra over X. We write θ_X^H for the structure map of this coalgebra. For the unique coalgebra morphism from an H-coalgebra (C, φ) to $(D^H X, \theta_X^H)$ that sends a morphism $\chi : C \to X$ to $\varepsilon_X^H : D^H X \to X$, we write $\mathsf{gen}_X^H(\chi, \varphi)$. For any object X, $\sigma_X^H = H\varepsilon_X^H \circ \theta_X^H : D^H X \to HX$. For any H-coalgebra (C, φ), object X and morphism $\chi : C \to X$, $\sigma_X^H \circ \mathsf{gen}_X^H(\chi, \varphi) = H\chi \circ \varphi : C \to HX$.

For cofree comonads, by specializing Theorem 4.4, we obtain our second theorem.

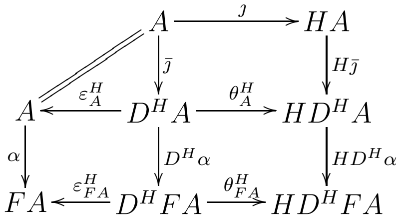
Theorem 4.6 *Let $F : \mathcal{C} \to \mathcal{C}$ be a functor, (A, α) a recursive F-coalgebra, $H : \mathcal{C} \to \mathcal{C}$ a functor with a cofree comonad and $(A, \jmath)$ a H-coalgebra. If $\lambda : FD^H \dot{\to} HF$ is a natural transformation satisfying*

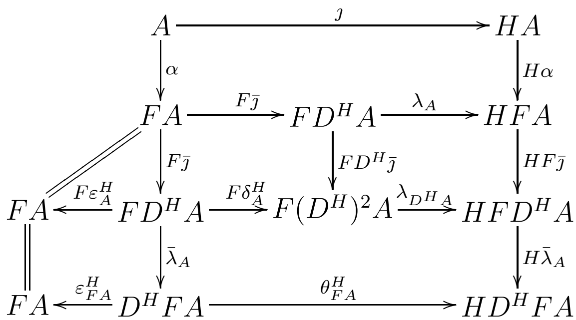


where $\bar{\jmath} = \mathsf{gen}_A^H(\mathsf{id}_A, \jmath)$, then $(A, F\bar{\jmath} \circ \alpha)$ is a recursive FD^H-coalgebra (and, consequently, $(A, H\alpha \circ \jmath)$ is a recursive HF-coalgebra).

Proof. Define a natural transformation $\bar{\lambda} : FD^H \dot{\to} D^H F$ by $\bar{\lambda}_X = \mathsf{gen}_{FX}^H(F\varepsilon_X^H, \lambda_{D^H X} \circ F\delta_X^H)$. It is easy to verify (these are standard lifting results) that $(A, \bar{\jmath})$ is a $\boldsymbol{D}^H$-coalgebra and $\bar{\lambda}$ a distributive law of F over $\boldsymbol{D}^H$.

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

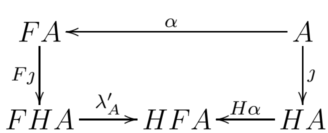




Therefore, by Theorem 4.4 (taking $\boldsymbol{D} = \boldsymbol{D}^H$, $\imath = \bar{\jmath}$, $\kappa = \bar{\lambda}$), we get that $(A, F\bar{\jmath} \circ \alpha)$ is a recursive FD^H-coalgebra. □

Our third theorem, where the cofree comonad does not appear manifestly, but is nonetheless present in the background, is a consequence from Theorem 4.6.

Theorem 4.7 *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*

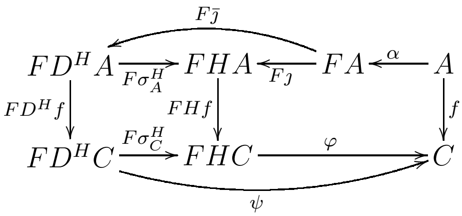


then $(A, F\jmath \circ \alpha)$ is a recursive FH-coalgebra.

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

Now consider an arbitrary FH-algebra (C, φ). Let $\psi = \varphi \circ F\sigma^H_C : FD^HC \to C$. 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).



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

Prop. 3.8 is now immediate provided 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 4.7 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 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 significant generalization (and modularization of the proofs) of the results of [27] on structured recursion schemes for initial algebras. By duality, they also generalize the dual results of [4,7].

This paper reports only our first results on recursive coalgebras and most of our questions are unanswered yet. Apart from checking whether the theorems of Section 4 can be strengthened in some useful ways, e.g. along the lines considered in [4] (modulo the duality) (replacing the assumption about the existence of a cofree comonad over H in Theorem 4.7 by some weaker condition), we would like to take a closer look at wellfounded induction. Taylor [24] has shown that a functor-algebra is recursive iff it is wellfounded in the sense of his categorical notion, but only for **Set** (or an elementary topos) and for functors preserving monos and inverse image diagrams. We would like to find out weaker useful conditions under which the implications in each direction remain valid. 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.

References

[1] 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.

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

[3] J. Adámek, S. Milius, J. Velebil, From iterative algebras to iterative theories, in this volume.

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

[5] A. Bove, V. Capretta, Nested general recursion and partiality in type theory, in: R. J. Boulton, P. B. Jackson (Eds.), Proc. of 14th Int. Conf. on Theorem Proving in Higher Order Logics, TPHOLs 2001 (Edinburgh, Sept. 2001), Vol. 2152 of Lecture Notes in Comput. Sci., Springer-Verlag, 2001, pp. 121–135.

[6] A. Bove, V. Capretta, Modelling general recursion in type theory, submitted to Math. Struct. in Comput. Sci. (Feb. 2003).

[7] D. Cancila, F. Honsell, M. Lenisa, Generalized coiteration schemata, in: H. P. Gumm (Ed.), Proc. of 6th Wksh. on Coalgebraic Methods in Computer Science, CMCS'03 (Warsaw, Apr. 2003), Vol. 82(1) of Electron. Notes in Theoret. Comput. Sci., Elsevier, 2003.

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

[9] A. Eppendahl, Coalgebra-to-algebra morphisms, in: M. Hofmann, G. Rosolini, D. Pavlović (Eds.), Proc. of 8th Int. Conf. on Category Theory and Computer Science, CTCS'99 (Edinburgh, Sept. 1999), Vol. 29 of Electron. Notes in Theoret. Comput. Sci., Elsevier, 1999.

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

[11] M. Escardó, A. K. Simpson, A universal characterization of the closed euclidean interval, in: Proc. of 16th Ann. IEEE Symp. on Logic in Computer Science, LICS'01 (Boston, June 2001), IEEE CS Press, 2001, pp. 115–128.

[12] P. J. Freyd, Algebraically complete categories, in: A. Carboni, M. C. Pedicchio, G. Rosolini (Eds.), Proc. of Int. Conf. on Category Theory '90, CT'90 (Como, July 1990), Vol. 1488 of Lecture Notes in Math., Springer-Verlag, 1991, pp. 95–104.

[13] P. J. Freyd, Recursive types reduced to inductive types, in: Proc. of 5th IEEE Ann. Symp. on Logic in Computer Science, LICS'90 (Philadelphia, PA, June 1990), IEEE CS Press, 1990, pp. 498–507.

[14] 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, Vol. 177 of LMS Lecture Note Series, Cambridge Univ. Press, 1992, pp. 95–106.

[15] E. Giménez, Codifying guarded definitions with recursion schemes, in: P. Dybjer, B. Nordström (Eds.), Selected Papers from 2nd Int. Wksh. on Types for Proofs and Programs, TYPES'94 (Båstad, June 1994), Vol. 996 of Lecture Notes in Comput. Sci., Springer-Verlag, 1995, pp. 39–59.

[16] E. Giménez, Structural recursive definitions in type theory, in: K. G. Larsen, S. Skyum, G. Winskel (Eds.), Proc. of 25th Int. Coll. on Automata, Languages and Programming, ICALP'98 (Aalborg, July 1998), Vol. 1443 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1998, pp. 397–408.

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

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

[19] S. Milius, Working with the CIA—completely iterative monads revisited, manuscript (Nov. 2003).

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

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

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

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

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

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

[26] D. Turi, G. D. Plotkin, Towards a mathematical operational semantics, in: Proc. of 12th Ann. IEEE Symp. on Logic in Computer Science, LICS'97 (Warsaw, June/July 1997), IEEE CS Press, 1997, pp. 280–291.

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

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