Math. Struct. in Comp. Science (2005), *vol.* 15, *pp.* 433–451. © 2005 Cambridge University Press
doi:10.1017/S0960129505004743 Printed in the United Kingdom

Monads of coalgebras: rational terms and term graphs

NEIL GHANI[†], CHRISTOPH LÜTH[‡] and FEDERICO DE MARCHI[†§]

[†] *Department of Mathematics and Computer Science, University of Leicester*
[‡] *FB 3 – Mathematik und Informatik, Universität Bremen*

Received 11 July 2003

This paper introduces guarded and strongly guarded monads as a unified model of a variety of different term algebras covering fundamental examples such as initial algebras, final coalgebras, rational terms and term graphs. We develop a general method for obtaining finitary guarded monads that allows us to define and prove properties of the rational and term graph monads. Furthermore, our treatment of rational equations extends the traditional approach to allow right-hand sides of equations to be infinite terms, term graphs or other such coalgebraic structures. As an application, we use these generalised rational equations to sketch part of the correctness of the term graph implementation of functional programming languages.

1. Introduction

Initial algebra semantics has long been regarded as one of the cornerstones of the semantics of programming languages. Within this paradigm, the syntax of a language is modelled as an initial algebra consisting of the finite terms of the language, while the semantics of the language is given as the unique map from the initial algebra into any other algebra. A similar situation arises in the theory of datatypes, where the initial algebra consists of the finite terms built from the constructors of the datatype while initiality allows functions to be defined over the datatype via structural recursion.

Categorically, one regards such an initial algebra as the initial algebra of an endofunctor F, which represents the language or datatype. In order to incorporate fundamental features such as *variables* and *substitution* into the framework, one considers not a single initial algebra, but rather, for each object X, the initial F-algebra *over* X, that is, the initial $X + F$-algebra. The mapping sending an object X, thought of as an object of variables, to (the carrier of) the initial $X + F$-algebra defines the free *monad* over F. The multiplication of the monad provides an abstract representation of substitution, and the unit models the variables, while the freeness of the monad models the inductive nature of the initial algebra. Applications to base categories other than Set have proved fruitful in many situations, for example, the study of S-sorted algebraic theories as monads over Set^S, the study of categories with structure using monads over **Graph** or **Cat** (Dubuc and Kelly

[§] Research supported by EPSRC grant GRM96230/01: *Categorical Rewriting: Monads and Modularity*

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

1983), the study of rewriting using monads over **Pre** or **Cat** (Lüth and Ghani 1997) and the study of higher order abstract syntax using monads over $\mathsf{Set}^{\mathbb{F}}$ (Fiore *et al.* 1999) (where $\mathbb{F}$ is the skeleton of the category of finite sets).

Of course, there are other term algebras other than the initial algebra of finite terms. For example, instead of finite terms, what about infinite terms? From a computational perspective one may argue that one should consider only those infinite terms that are computable in some sense. Or, from a recursion-theoretic perspective, what about rational terms, which are those terms definable by recursive equations of a certain form? From the perspective of the implementation of functional languages, what about term graphs, which are the dominant model of syntax within that field? These questions are of clear practical importance and the papers cited above suggest the generalisation of these term algebras beyond the category of Set.

There has already been some work in this direction within the coalgebra community. Indeed, Moss (Moss 2001) gave what amounts to a rather full answer in the case of the term algebra of finite and infinite terms by showing that they form the final $X + F$-coalgebra, thereby elegantly dualising the initial algebra characterisation of finite syntax trees. In addition, he showed that the collection of these coalgebras also forms a monad, thereby meeting our requirement for substitution to be taken into account. These results were discovered independently in Aczel *et al.* (2001) and in Ghani *et al.* (2001), though the latter paper uses a more restrictive setting. Independently of us, Adamek *et al.* (2003) considers the specific term algebra of rational terms. This paper encodes the traditional definition of rational equations categorically, shows that the resulting terms form a rational monad, and then characterises this monad as the free iterative monad.

Nevertheless, these results deal only with the specific term algebras and we do not want to tackle each term algebra on a case-by-case basis. To answer explicitly a question we have been asked on a number of occasions, our intent is to go beyond the work cited above that deals with infinite terms and rational terms and to lay the foundations of a meta-theory encompassing a wide variety of term algebras. Thus this paper introduces *guarded monads* as a uniform framework for such term algebras. In detail, we:

— introduce the notion of a guarded monad and demonstrate that this definition captures a number of important examples;
— provide a general theorem for building monads from coalgebras, and use this theorem to prove that the key examples of rational terms and term graphs are guarded;
— solve rational equations whose right-hand sides are not just finite terms but belong to any strongly guarded monad. This underpins part of the correctness of the implementation of functional programming languages via term graphs.

The term graph monad is possibly the most important contribution of this work. While the denotational semantics of functional languages provides an elegant framework for reasoning, the semantics of the implementation of functional languages has remained relatively low level. While some attempts to model term graphs using abstract techniques have been made (Corradini and Gadducci 1997), they have yet to make a significant impact within the functional programming community. Our impression is that the mathematical structures underlying these models, for example, 2-categories, double categories, *etc.*, are

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

significantly more complex than our approach of modelling term graphs as particular coalgebras. We hope the simplicity of the coalgebraic model of term graphs will help bridge this gap between the semantics and implementation of functional languages.

This paper has undergone some changes since its first draft, and we would like to acknowledge the referees for their constructive comments. In particular, previous versions used a concept called a *coalgebraic monad*, which has now been renamed as a *strongly guarded monad*. A number of colleagues pointed out that the isomorphism inherent in that definition was too strong for certain examples and, after some thought, we have come to agree with them, and so introduced the weaker notion of a guarded monad. In addition, De Marchi (Marchi 2003) has now completed his thesis, which contains detailed proofs of all the theorems stated here.

The paper is structured as follows: Section 2 introduces the notion of a coalgebraic monad. Section 3 contains a general theorem for proving that a given class of term algebras forms a monad and, then, applies this to the cases of the rational monad and the term graph monad. Section 4 extends Section 3 to cover coalgebraic monads, and Section 5 contains our partial correctness result.

2. Coalgebraic monads

Let $F : \mathscr{C} \longrightarrow \mathscr{C}$ be a functor, which we may think of as arising from a signature of some form. If TX is some set of terms built from F using a set of variables X, we take the following properties to be desirable:

— TX should contain all the variables X and be closed under applications of term constructors from F. Thus TX should be an $X + F$-algebra.
— In order to have a well-behaved notion of substitution, the map sending X to TX should be a monad

Thus we shall define a monad T to be F-guarded iff T is an $(\mathsf{Id}+F\circ-) : [\mathscr{C},\mathscr{C}] \longrightarrow [\mathscr{C},\mathscr{C}]$-algebra. In certain situations, every term $t \in TX$ should either be a variable or start with a term constructor from F, which is captured by asking T to be an $(\mathsf{Id}+F\circ-)$-coalgebra. If these algebra and coalgebra structure maps are inverse, we call T a strongly F-guarded monad[†]. Although this is our central intuition, we formally introduce these concepts by first noting the following lemma.

Lemma 2.1. Let (T,η,μ) be a monad and F be an endofunctor on a category $\mathscr{C}$. There is a bijection between natural transformations $\tau : F \longrightarrow T$ and natural transformations $\alpha : FT \longrightarrow T$ making the following diagram commute

$$\begin{array}{ccc} FTT & \xrightarrow{F\mu} & FT \\ {\scriptstyle \alpha_T}\downarrow & & \downarrow{\scriptstyle \alpha} \\ TT & \xrightarrow[\mu]{} & T \end{array} \qquad (1)$$

[†] We ignore size issues here since we intend to work with lfp categories and finitary functors.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

Proof. Given τ, define $\alpha = \mu.\tau_T$. Commutation of (1) follows immediately by naturality of τ and the associativity of μ:

$$\begin{array}{ccccc} FTT & \xrightarrow{\tau_{TT}} & TTT & \xrightarrow{\mu_T} & TT \\ {\scriptstyle F\mu}\downarrow & & \downarrow{\scriptstyle T\mu} & & \downarrow{\scriptstyle \mu} \\ FT & \xrightarrow[\tau_T]{} & TT & \xrightarrow[\mu]{} & T \end{array}$$

Conversely, given α, define $\tau = \alpha.F\eta$. It is easy to show that the two mappings are mutually inverse. □

Condition (1) just says that if we think of α as transforming F terms over T into T terms, it does not make any difference if we multiply two terms under the F context and then transform, or rather transform the upper term and then multiply it with the second. In other words, (1) implies that $\mu : \alpha_T \longrightarrow \alpha$ is an F-algebra homomorphism.

Definition 2.2. An *F-guarded monad* on $\mathscr{C}$ is a 4-tuple (T, η, μ, τ) such that (T, η, μ) is a monad on $\mathscr{C}$ and $\tau : F \longrightarrow T$ is a natural transformation. A *morphism of F-guarded monads* between (T, η, μ, τ) and (T', η', μ', τ') is a monad morphism ϕ from (T, η, μ) and (T', η', μ') such that $\phi\tau = \tau'$.

We argued earlier that F-guarded monads should have a $(\mathsf{Id} + F \circ -)$-algebra structure on them. This is verified by the following lemma, which is a simple case of diagram chasing (Marchi 2003).

Lemma 2.3. Let (T, η, μ, τ) be an F-guarded monad on $\mathscr{C}$. Let $\alpha : FT \longrightarrow T$ be the natural transformation induced by τ, such that (1) commutes. Define the natural transformations $\overline{\eta} = \mathsf{inl} : \mathsf{Id} \longrightarrow \mathsf{Id} + FT$; and

$$\overline{\mu} : \mathsf{Id} + FT + FT(\mathsf{Id} + FT) \xrightarrow{\mathsf{Id}+FT+FT[\eta,\alpha]} \mathsf{Id} + FT + FT^2 \xrightarrow{[\mathsf{Id}+FT,F\mu]} \mathsf{Id} + FT.$$

Then, the triple $(\mathsf{Id} + FT, \overline{\eta}, \overline{\mu})$ is a monad, and the map $[\eta, \alpha] : \mathsf{Id} + FT \longrightarrow T$ is a monad morphism. Moreover, given another F-guarded monad (S, η', μ', τ') and a morphism of guarded monads $\psi : T \longrightarrow S$, there is a a monad morphism $\mathsf{Id} + F\psi$ such that $[\eta', \alpha'](\mathsf{Id} + F\psi) = \psi[\eta, \alpha]$.

So, for any F-guarded monad T, we have that TX is naturally an $X + F$-algebra. We shall say that those monads for which this algebra structure is invertible are *strong*.

Definition 2.4. Let F be an endofunctor on a category $\mathscr{C}$. A *strongly F-guarded monad* on $\mathscr{C}$ is an F-guarded monad $T = (T, \eta, \mu, \tau)$ such that the induced monad morphism $[\eta, \mu.\tau_T] : \mathsf{Id} + FT \longrightarrow T$ is an isomorphism. Strongly guarded monads form a full subcategory of the category of F-guarded monads.

We will now give some examples.

Proposition 2.5 (Initial F-guarded monad). Let (T, η, μ) be the free monad over an endofunctor $F : \mathscr{C} \longrightarrow \mathscr{C}$. Then, T is the initial (strongly) F-guarded monad.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

Proof. Freeness gives a natural transformation $\tau : F \longrightarrow T$. Verifying the conditions of Lemma 2.3 shows that $\phi = [\eta, \mu.\tau T] : 1 + FT \longrightarrow T$ is a monad morphism. The transformation $\mathsf{inr} \circ F\eta : F \longrightarrow \mathsf{Id} + FT$ induces, by freeness of T, a monad morphism $\psi : T \longrightarrow 1 + FT$. Diagram chases show that ϕ and ψ are mutually inverse, and hence T is a (strongly) F-guarded monad. Given any other (strongly) F-guarded monad (T', τ'), freeness and the transformation $\tau' : F \longrightarrow T'$ gives a unique monad morphism $! : T \longrightarrow T'$ such that $\tau' = !\tau$. □

A slicker proof is possible when, for each object X, the action of the free monad is given by the (underlying object of the) initial $X + F$-algebra. In such a setting, T is the initial algebra of the endofunctor $(\mathsf{Id} + F \circ -) : [\mathscr{C}, \mathscr{C}] \longrightarrow [\mathscr{C}, \mathscr{C}]$ and, since all initial algebras are isomorphisms, T is isomorphic to $1 + FT$. Initiality follows since every (strongly) F-guarded monad is a $(\mathsf{Id} + F \circ -)$-algebra. This proof covers the canonical case of finitary functors over the category Set.

Lemma 2.6 (Final strongly F-guarded monad). Let $F : \mathscr{C} \longrightarrow \mathscr{C}$ be and let T^ν be the final $(\mathsf{Id} + F \circ -)$-coalgebra. Then T^ν is the final strongly F-guarded monad.

Proof. The fact that T^ν is a monad such that there is an isomorphism $[\eta, \alpha] : \mathsf{Id} + FT^\nu \longrightarrow T^\nu$ is proved in Aczel *et al.* (2001). Furthermore, the fact that α satisfies condition (1) also follows from their substitution theorem. Thus T^ν is strongly F-guarded. Any other strongly F-guarded monad S is an $(\mathsf{Id} + F \circ -)$-coalgebra, and hence there is a unique $(\mathsf{Id} + F \circ -)$-coalgebra homomorphism between S and T^ν. This is also an F-guarded monad morphism, as one can prove with a bit of diagram chasing (Marchi 2003) and using the finality of T^ν. Uniqueness follows from finality. □

Returning to our motivations concerning term algebras, the monad T^ν is of practical importance as it captures the set of finite and infinite terms just as the free monad captures the set of finite terms. A number of other subalgebras of T^ν are strongly F-guarded monads over Set, as can be seen by showing that the multiplication of T^ν restricts to the subalgebra. In this way one can show that the following are all strongly F-guarded monads over Set:

(i) infinite terms that contain only a finite number of variables;

(ii) locally finite terms (Courcelle 1983), that is, finite and infinite terms that have the property that from every node, there is a finite path to a leaf;

(iii) rational terms are terms with a finite number of subterms, or, more formally, the free iterative theory over a signature (Elgot *et al.* 1978).

However, guarded monads also capture other syntactic structures that are not subalgebras of T^ν, for example, term graphs that model recursion and sharing through the use of cycles and multiple edges.

We now develop a general theorem for deriving monads as pointwise colimits and use this result to define the rational and term graph monads.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

3. When is a pointwise colimit a monad?

3.1. *Lfp categories and Kleisli monoids*

Inherent in the notions of signature, terms, substitution *etc*., is the concept of *arity*, which categorically means the representing monad has a rank. To understand this condition, the initial algebra $T_\Sigma X$ built over a signature satisfies

$$T_\Sigma(X) = \bigcup_{X_0 \subset X \text{ is finite}} T_\Sigma(X_0). \tag{2}$$

This equation holds because all the operators in Σ have a finite arity, and thus a term built over X can only contain a finite number of variables. Such monads are *finitary*, and we restrict our attention to them as they capture most of the key examples, and the theory generalises straightforwardly to functors with higher rank. The relationship between signatures and their representing monads can be generalised to *lfp-categories*, that is, cocomplete categories generated by a set of *finitely presentable* objects, where an object is finitely presentable if its covariant hom functor preserves filtered colimits. See Adamek and Rosický (1994) for more details on lfp categories and Kelly and Power (1993) for a more detailed description of the generalisation of algebraic theories to lfp categories.

If $\mathscr{C}$ is an lfp-category, let $\mathscr{C}_{\text{fp}}$ be the full subcategory of finitely presentable objects with inclusion $J : \mathscr{C}_{\text{fp}} \longrightarrow \mathscr{C}$. A functor $F : \mathscr{C} \longrightarrow \mathscr{D}$ is *finitary* if it preserves filtered colimits, or, equivalently, if it is isomorphic to the left Kan extension along J of its restriction to $\mathscr{C}_{\text{fp}}$. Hence, the category $Fin[\mathscr{C}, \mathscr{D}]$ of finitary functors from $\mathscr{C}$ to $\mathscr{D}$ is equivalent to the functor category $[\mathscr{C}_{\text{fp}}, \mathscr{D}]$.

When $\mathscr{D} = \mathscr{C}$, the category of finitary endofunctors is monoidal with the identity as unit and multiplication given by composition. Similarly, $[\mathscr{C}_{\text{fp}}, \mathscr{C}]$ is also monoidal with the inclusion J as unit and multiplication given by $F \Box G = \mathsf{Lan}_J F \circ G$. Now, to give a finitary monad on $\mathscr{C}$ is to give a monoid in $Fin[\mathscr{C}, \mathscr{C}]$, which is thus a monoid in the equivalent category $[\mathscr{C}_{\text{fp}}, \mathscr{C}]$ (Kelly and Power 1993). Thus we can prove that rational terms, term graphs or some other term algebra form a finitary monad by proving their restriction to $[\mathscr{C}_{\text{fp}}, \mathscr{C}]$ is a monoid. Such a monoid can be obtained from what we call a *Kleisli monoid*, which is nothing more than a Kleisli triple reformulated appropriately for the monoidal category $[\mathscr{C}_{\text{fp}}, \mathscr{C}]$.

Definition 3.1. A *Kleisli monoid* for an lfp category $\mathscr{C}$ consists of

— a function T assigning to each object X in $\mathscr{C}_{\text{fp}}$ an object TX in $\mathscr{C}$,
— for each X in $\mathscr{C}_{\text{fp}}$ a map $\eta_X : X \longrightarrow TX$ in $\mathscr{C}$, and
— for objects X, Y in $\mathscr{C}_{\text{fp}}$, a lifting function $\mathrm{s}_{X,Y} : \mathscr{C}(X, TY) \longrightarrow \mathscr{C}(TX, TY)$,

satisfying the following conditions:

$$\mathrm{s}_{X,X}(\eta_X) = 1_{TX} \quad \mathrm{s}_{X,Y}(f).\eta_X = f \quad \mathrm{s}_{X,Z}(\mathrm{s}_{Y,Z}(g).f) = \mathrm{s}_{Y,Z}(g).\mathrm{s}_{X,Y}(f).$$

In the rest of the paper, we shall omit the subscript to the function s, whenever possible. The definition does not substantially differ from that of a Kleisli triple for a category $\mathscr{C}$ and, not surprisingly, defines a monoid in the category $[\mathscr{C}_{\text{fp}}, \mathscr{C}]$, and hence a finitary

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

monad on $\mathscr{C}$. In doing so we make use of the well-known formula for Kan extensions in terms of coends and wedges. See Mac Lane (1971) for a reference to this material.

Lemma 3.2. If $\mathscr{C}$ is an lfp category, every Kleisli monoid for $\mathscr{C}$ defines a monoid in the category $[\mathscr{C}_{\mathrm{fp}}, \mathscr{C}]$.

Proof. First, T extends to a functor $T : \mathscr{C}_{\mathrm{fp}} \longrightarrow \mathscr{C}$ and a natural transformation of the form $\eta : J \longrightarrow T$ by setting $T(f) = s(\eta.f)$. Multiplication requires maps of the form $\mu_S : (T\Box T)S \longrightarrow TS$. Since $(T\Box T)S$ is defined via the Kan extension,

$$(T\Box T)S = \mathsf{Lan}_J T(TS) = \int^{n \in \mathscr{C}_{\mathrm{fp}}} \mathscr{C}(n, TS) \bullet Tn \longrightarrow TS.$$

We therefore require a wedge $w_n : \mathscr{C}(n, TS) \bullet Tn \longrightarrow TS$, which, through the universal property of the tensor, is a family of maps $\mathscr{C}(n, TS) \longrightarrow \mathscr{C}(Tn, TS)$, which is precisely what is given by the lifting of the Kleisli monoid. That these maps do indeed form a wedge follows from the laws of the Kleisli monoid. Naturality follows from the parametricity theorem for coends, while the laws of a monoid again boil down to the laws of the Kleisli monoid. See Marchi (2003) for more details. □

3.2. *Some general properties of colimits*

In the next section we will take a finitary functor F and define the rational monad at an object X to be the colimit of the inclusion of the full subcategory of $X + F$-coalgebras with finitely presentable carrier. A similar pointwise colimit construction will be used to define the term graph monad. Rather than prove these results directly, we consider the more general question of when is a pointwise colimit a monad? Note that this is a fairly technical section and its inclusion is merely to systematise the proofs that rational terms and term graphs form monads. Consequently, the reader may prefer to skim this section, consider the examples and then reread this section in greater depth later.

We begin by abstracting the data we have. First, for every object X, we are interested in a subcategory of the category of $X + F$-coalgebras. This we model by a functor $\mathscr{I} : \mathscr{C} \longrightarrow \mathsf{Cat}$. Of key importance are the forgetful functors U_X from the (sub)-category of $X + F$-coalgebras to $\mathscr{C}$. If we use $\mathsf{K}_{\mathscr{C}} : \mathscr{C} \longrightarrow \mathsf{Cat}$ to denote the constant functor mapping each object to $\mathscr{C}$, the collection of forgetful functors forms a natural transformation $U : \mathscr{I} \Rightarrow \mathsf{K}_{\mathscr{C}} : \mathscr{C} \longrightarrow \mathsf{Cat}$. Thus, we can formalise our question mathematically as to when the assignment $X \mapsto \mathrm{colim} U_X$ defines a finitary monad. We answer this question by finding a Kleisli monoid structure for its restriction to $\mathscr{C}_{\mathrm{fp}}$. However, we first introduce some notation concerning colimits since colimits are central in what follows.

Definition 3.3. Let $\mathscr{C}$ be a 2-category and X be an object of $\mathscr{C}$. For the *lax slice 2-category* Lax_X:

— Objects of Lax_X are maps $f : Y \longrightarrow X$ in $\mathscr{C}$.
— Arrows $\alpha : f \longrightarrow g$ of Lax_X are pairs $\langle h, \alpha \rangle$ where $\alpha : f \Rightarrow gh$ in $\mathscr{C}$. If $\alpha : f \longrightarrow g$ and $\beta : g \longrightarrow h$ are morphisms in Lax_X, we write their composite as $\beta \diamond \alpha : f \longrightarrow h$.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

— Given maps $\langle h, \alpha\rangle : f \longrightarrow g$ and $\langle h', \alpha'\rangle : f \longrightarrow g$, a 2-cell $\langle h, \alpha\rangle \longrightarrow \langle h', \alpha'\rangle$ consists of a 2-cell $\theta : h \longrightarrow h'$ in $\mathscr{C}$ such that $\alpha' = g\theta.\alpha$.

The usual definition of a slice category $\mathscr{C}/X$ is the lax slice category on the 2-category obtained by adding only the identity 2-cells to $\mathscr{C}$. Henceforth, given a diagram D with colimit X, we shall use $\overline{d}$ to indicate the colimiting map $\overline{d} : d \longrightarrow X$ for any object d in D. Using this notation, lax slice categories allow us to state the properties of colimits we require as follows.

Lemma 3.4. Let $\mathscr{C}$ be a cocomplete category. Consider the lax slice category $\mathsf{Lax}_{\mathscr{C}}$ built over the 2-category Cat. The colimit operation defines a 2-functor $\mathrm{colim} : \mathsf{Lax}_{\mathscr{C}} \longrightarrow \mathscr{C}$ where we regard $\mathscr{C}$ as the 2-category obtained from $\mathscr{C}$ by adding identity 2-cells.

Unwinding Lemma 3.4 means that if $\langle H, \alpha\rangle \in \mathsf{Lax}_{\mathscr{C}}(F, G)$, the family of arrows $\overline{Hd}.\alpha_d$ defines a cocone over F and hence a map $\mathrm{colim}\alpha : \mathrm{colim}F \longrightarrow \mathrm{colim}G$. In addition, $\mathrm{colim}1_F = 1 : \mathrm{colim}F \longrightarrow \mathrm{colim}F$ and $\mathrm{colim}(\beta \diamond \alpha) = \mathrm{colim}\beta.\mathrm{colim}\alpha$. Finally, given a 2-cell $\alpha \longrightarrow \alpha'$, we have $\mathrm{colim}\alpha = \mathrm{colim}\alpha'$.

3.3. *Obtaining the data for a Kleisli monoid*

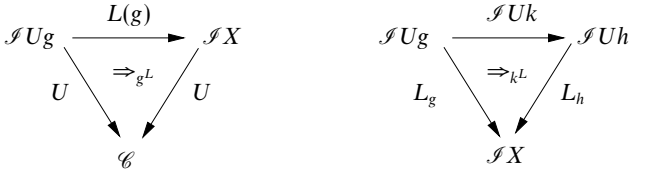
We define $TX = \mathrm{colim}U_X$ for X finitely presentable and consider what is required for TX to be a Kleisli monoid. To obtain a unit for T, we assume that for each finitely presentable object $X \in \mathscr{C}_{\mathrm{fp}}$ there is an object $i_X \in \mathscr{I}X$ such that $U_X(i_X) = X$. Thus we can define η_X to be the inclusion $\overline{i_X} : X = U_X(i_X) \longrightarrow \mathrm{colim}U_X = TX$. In our examples, i_X will be the coalgebra $\mathsf{inl} : X \longrightarrow X + FX$, whose inclusion embeds the variables into the colimit.

Finally, we turn to the lifting functions for a Kleisli monoid. Since $TX = \mathrm{colim}U_X$, we can extend a map $X \longrightarrow TY$ to a map $TX \longrightarrow TY$ by mapping each $X + F$-coalgebra into a $Y + F$-coalgebra and then constructing the associated cocone into TY. Since X is finitely presentable, X can actually be mapped to TY_0 for some subobject Y_0 of Y and then we can add Y_0 to the carrier of each $X + F$-coalgebra, thereby making it into a $Y + F$-coalgebra. Formally, this procedure is captured by what we call a lifting.

Definition 3.5. $\mathscr{I}$ is said to have a *lifting* L when:

— For each object $X \in \mathscr{C}$ and each $g \in \mathscr{I}X$, there is an arrow $\langle L(g), g^L\rangle : U_{Ug} \longrightarrow U_X$ in $\mathsf{Lax}_{\mathscr{C}}$, that is, a functor $L(g) : \mathscr{I}Ug \longrightarrow \mathscr{I}X$ and a natural transformation $g^L : U \Rightarrow U.L(g)$.

— For each arrow $k : g \longrightarrow h$ in $\mathscr{I}X$, there exists a natural transformation of the form $k^L : L(g) \longrightarrow L(h).\mathscr{I}Uk$ such that $h^L.\mathscr{I}Uk = Uk^L.g^L$

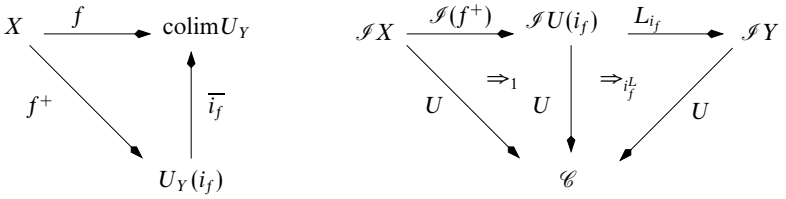


https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

A more abstract definition of a lifting is given as follows. First note that for each object X, U_X is an object of the 2-category $\mathsf{Lax}_{\mathscr{C}}$. Thus we may form the lax slice category Lax_{U_X}, which (by ignoring the 2-cells) defines a functor $\mathsf{Lax}_{U_} : \mathscr{C} \longrightarrow \mathsf{Cat}$. A lifting is then a natural transformation $\mathscr{I} \longrightarrow \mathsf{Lax}_{U_}$. As usual, see Marchi (2003) for more details.

Lemma 3.6. Let X and Y be finitely presentable, $\mathscr{I}Y$ be a filtered category and $\mathscr{I}$ have a lifting. Then, any map $f : X \longrightarrow \mathrm{colim}U_Y$, induces a map $\mathrm{s}(f) : \mathrm{colim}U_X \longrightarrow \mathrm{colim}U_Y$.

Proof. Since X is finitely presentable and $\mathscr{I}Y$ is filtered, f factors as $f = \overline{i_f}.f^+$ for some $i_f \in \mathscr{I}Y$. This defines a functor $Lf = L(i_f).\mathscr{I}(f^+)$ and also a natural transformation $f^L = i_f^L \diamond 1$



and hence, by Lemma 3.4, a map $\mathrm{s}(f) = \mathrm{colim}f^L : \mathrm{colim}U_X \longrightarrow \mathrm{colim}U_Y$.

The construction of $\mathrm{s}(f)$ appears to depend upon the factorisation $f = \overline{i_f}.f^+$, but this is not actually the case. In fact, suppose $f = \overline{i_h}.h$ is another factorisation, inducing the functor and natural transformation

$$L(i_h).\mathscr{I}(h) : \mathscr{I}X \longrightarrow \mathscr{I}Y \text{ and } i_h^L \diamond 1 : U \Rightarrow UL(i_h).\mathscr{I}(h).$$

By Lemma 3.4, we have a map $\mathrm{colim}(i_h^L \diamond 1) : TX \longrightarrow TY$. Because $\mathscr{I}Y$ is filtered, there exist i, $k : i_f \longrightarrow i$ and $k' : i_h \longrightarrow i$ such that f factorises as $f = \overline{i}.g$, for a given map $g : X \longrightarrow U_Y(i)$, with $Uk.f^+ = g = Uk'.h$. Let us focus on $k : i_f \longrightarrow i$. The lifting L determines $k^L : L(i_f) \Rightarrow L(i).\mathscr{I}Uk$. Then

$$i^L \diamond 1 = i^L.\mathscr{I}(Uk).\mathscr{I}(f^+) = (Uk^L.i_f^L).\mathscr{I}(f^+) = Uk^L.\mathscr{I}(f^+).i_f^L.\mathscr{I}(f^+),$$

where the first equality comes from $\mathscr{I}(g) = \mathscr{I}(Uk).\mathscr{I}(f^+)$, the second is from the lifting of k^L and the third is the distribution of horizontal over vertical composition of natural transformations. Thus $i^L \diamond 1 = Uk^L.\mathscr{I}(f^+).i_f^L \diamond 1$, and by Lemma 3.4, we have $\mathrm{colim}(i^L \diamond 1) = \mathrm{colim}(i_f^L \diamond 1)$. Analogously, $\mathrm{colim}(i^L \diamond 1) = \mathrm{colim}(i_h^L \diamond 1)$, hence $\mathrm{s}(f)$ does not depend on the factorisation. □

3.4. *Verifying the laws for a Kleisli monoid*

Reusing the notation above, we define for the canonical factorisation $f = \overline{i_f}.f^+$ the functor $L(f) = L(i_f).\mathscr{I}(f^+)$ and $f^L = i_f^L.\mathscr{I}(f^+)$. Having obtained our candidates for a Kleisli triple, we now verify the three laws for which we need further assumptions. For the lifting of the unit, note first that there is a factorisation $\eta_X = \overline{i_X}.1$. Thus, by Lemma 3.4, $\mathrm{s}(\eta) = \mathrm{colim}(i_X^L \diamond 1) = \mathrm{colim}i_X^L \mathrm{colim}1 = \mathrm{colim}i_X^L$. This verifies the first law for a Kleisli monoid.

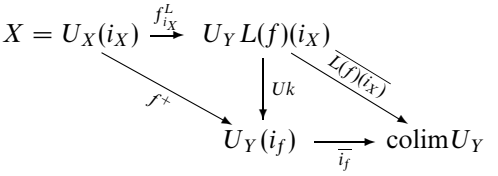
Lemma 3.7. Let $X \in \mathscr{C}$. If $\mathrm{colim}i_X^L = 1$, then $\mathrm{s}(\eta_X) = 1_{UX}$.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

Now we establish the second law for a Kleisli monoid.

Lemma 3.8. Let X and Y be finitely presentable and $f : X \longrightarrow \mathrm{colim} U_Y$. If there is a map $k : L(f)(i_X) \longrightarrow i_f$ in $\mathscr{I}Y$ such that $Uk.f^L_{i_X} = f^+$, then $\mathrm{s}(f).\eta_X = f$.

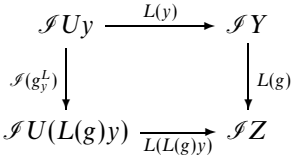
Proof. $\mathrm{s}(f)$ is determined by a factorisation $f = \overline{i_f}.f^+$. Since η_X is the inclusion from $U_X(i_X)$ to $\mathrm{colim} U_X$, $\mathrm{s}(f).\eta$ is the $U_X(i_X)$-th component of the cocone determined by f as in Lemma 3.6, that is, $\mathrm{s}(f).\eta = \overline{L(f)(i_X)}.f^L_{i_X}$. Thus:



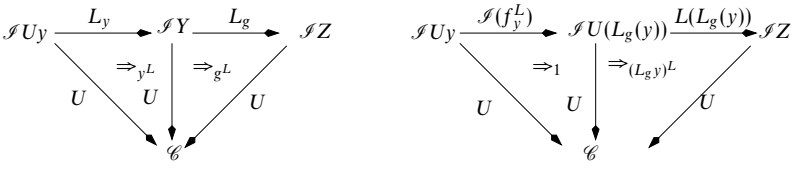
where the left triangle commutes by assumption and the right since it is part of the universal cocone over U_Y. □

To establish the third and final law of a Kleisli monoid, we require some extra assumptions as follows.

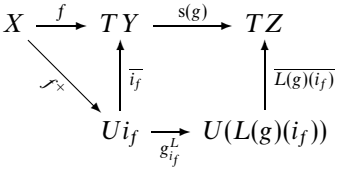
Lemma 3.9. Let X, Y and Z be finitely presentable, and consider the two maps $f : X \longrightarrow \mathrm{colim} U_Y$ and $g : Y \longrightarrow \mathrm{colim} U_Z$. Assume, for all $y \in \mathscr{I}Y$,



and $g^L \diamond y^L = (L(g)y)^L \diamond 1$. Then $\mathrm{s}(\mathrm{s}(g).f) = \mathrm{s}(g).\mathrm{s}(f)$



Proof. The map $\mathrm{s}(g).f$ can be factorised as follows:



where $f = \overline{i_f}.f^+$ is the factorisation of f used in the construction of $\mathrm{s}(f)$ and the square commutes by the construction of $\mathrm{s}(g)$. Thus we have a factorisation of $\mathrm{s}(g).f$, and the functor $L(\mathrm{s}(g).f)$ can be calculated as

$$L(\mathrm{s}(g).f)) = L(g^L i_f).\mathscr{I}(g^L_{i_f}).\mathscr{I}(f^+) = L(g).L(i_f).\mathscr{I}(f^+) = L(g).L(f),$$

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

where the third equality is by the definition of $L(f)$. From the assumptions, the equality $(L(g)i_f)^L \diamond 1 = g^L \diamond i_f^L : \mathscr{I}U_X(i_f) \longrightarrow \mathscr{I}Z$ is also derivable. By precomposing with $\mathscr{I}f^+$, we get that the two 2-cells associated to $L(\mathrm{s}(g).f)$ and $L(g).L(f)$ are equal, and thus $\mathrm{s}(\mathrm{s}(g).f) = \mathrm{s}(g).\mathrm{s}(f)$. □

Collecting all the results so far, we have established the following theorem.

Theorem 3.10. Let $\mathscr{C}$ be an lfp category, and $\mathscr{I} : \mathscr{C}_{\mathrm{fp}} \longrightarrow \mathsf{Cat}$ be a functor such that for every X in $\mathscr{C}_{\mathrm{fp}}$, $\mathscr{I}X$ is filtered. Let $U : \mathscr{I} \Rightarrow \mathsf{K}_{\mathscr{C}} : \mathscr{C}_{\mathrm{fp}} \longrightarrow \mathsf{Cat}$ be a natural transformation with a lifting L. For arbitrary objects X, Y and Z in $\mathscr{C}_{\mathrm{fp}}$ and maps $f : X \longrightarrow \mathrm{colim}U_Y$ and $g : Y \longrightarrow \mathrm{colim}U_Z$ in $\mathscr{C}$, assume that:

— There is an object i_X in $\mathscr{I}X$ such that $U_X(i_X) = X$ and $\mathrm{colim}i_X^L = 1$.
— There is a map $k : L(f)(i_X) \longrightarrow i_f$ such that $Uk.f_{i_X}^L = f^+$.
— For all y in $\mathscr{I}Y$, $g^L \diamond y^L = (L(g)y)^L \diamond 1$ and

$$\begin{array}{ccc} \mathscr{I}Uy & \xrightarrow{L(y)} & \mathscr{I}Y \\ {\scriptstyle \mathscr{I}(g_y^L)}\downarrow & & \downarrow{\scriptstyle L(g)} \\ \mathscr{I}U(L(g)y) & \xrightarrow[L(L(g)y)]{} & \mathscr{I}Z \end{array}$$

Then the assignment $TX = \mathrm{colim}U_X$ carries a Kleisli monoid structure.

Putting all of the work in this section together, we have the following corollary stating the conditions under which pointwise colimits form a monad.

Corollary 3.11. Under the assumptions of the previous theorem, the left Kan extension of T under the inclusion of $\mathscr{C}_{\mathrm{fp}}$ into $\mathscr{C}$ is a finitary monad on $\mathscr{C}$.

4. Rational and term graph monads

We now use our general theorem to prove that term graphs and rational terms form monads. First we state some general properties of categories of coalgebras. We write $U_X : (X+F)\text{–}\mathsf{coalg} \longrightarrow \mathscr{C}$ for the forgetful functor sending each coalgebra to its carrier. U_X creates colimits, and since $\mathscr{C}$ is lfp, and hence cocomplete, any functor $J : D \longrightarrow (X+F)\text{–}\mathsf{coalg}$ from a small category $\mathscr{D}$ has a colimit and $U_X\mathrm{colim}J = \mathrm{colim}U_X J$. Next, we have the following lemma.

Lemma 4.1. Let $F : \mathscr{C} \longrightarrow \mathscr{C}$ be a functor. The assignment sending an object X of $\mathscr{C}$ to the category $(X+F)\text{–}\mathsf{coalg}$ defines a functor $\Phi : \mathscr{C} \longrightarrow \mathsf{Cat}$. In addition, $U : \Phi \Rightarrow \mathsf{K}_{\mathscr{C}}$ is a natural transformation.

Proof. If $f : X \longrightarrow Y$ is a map in $\mathscr{C}$, then $\Phi(f)$ sends an $X + F$-coalgebra (S, h) to the $X + F$-coalgebra $(S, (f + 1).h)$. The action of $\Phi(f)$ on coalgebra morphisms and functoriality is easily proved. Naturality holds since $U_X = U_Y\Phi(f)$. □

In general, we shall instantiate Theorem 3.10 with functors that are variously related to Φ and whose properties follow from those of Φ. Similarly, we define a lifting for Φ from which liftings for other functors can be obtained.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

Lemma 4.2. Let (S, g) be an $X+F$-coalgebra. The map sending any $S+F$-coalgebra (A, m) to the following $X+F$-coalgebra defines a lifting for Φ:

$$A + S \xrightarrow{[m,\mathsf{inl}]} S + FA \xrightarrow{g+1} X + FS + FA \xrightarrow{1+[F\mathsf{inr},F\mathsf{inl}]} X + F(A + S)$$

Proof. We have defined the object part of a functor $L(g) : \Phi Ug \longrightarrow \Phi X$ and the action of $L(g)$ on morphisms is also easily defined. We have that the (A, m)-th component of $g^L : U_S \longrightarrow U_X.L(g)$ is the inclusion $g^L_m = \mathsf{inl} : A \longrightarrow A + S$. Naturality of g^L is the naturality of inl. Given any $X + F$-coalgebra map $k : (S, g) \longrightarrow (S', h)$, define k^L to be the natural transformation whose component on an $(S+F)$-coalgebra $m : A \longrightarrow S + FA$ is the following $X + F$-coalgebra morphism:

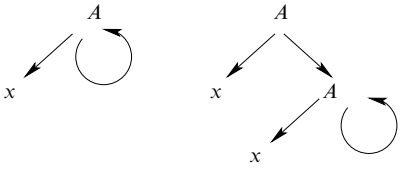
$$\begin{array}{ccc} A + S & \xrightarrow{L(g)(m)} & X + F(A + S) \\ {\scriptstyle 1+k}\downarrow & & \downarrow{\scriptstyle 1+F(1+Uk)} \\ A + S' & \xrightarrow[L(h)(\Phi k(m))]{} & X + F(A + S') \end{array}$$

The components of the natural transformations representing the 2-cell condition are $(1+k).\mathsf{inl} : A \longrightarrow A + S'$ and $\mathsf{inl} : A \longrightarrow A + S'$, which are clearly equal. □

4.1. *The rational monad*

Given a signature Σ, rational terms are the set of finite and infinite terms that arise as solutions of systems of equations of the form $e_1 = t_1, \ldots, e_n = t_n$ where each t_i is a finite term built from the signature Σ and whose variables are from the union of a fixed set X and $E = \{e_1, \ldots, e_n\}$. In order to get a solution, one insists that each term t_i is not an element of E. Rephrasing this categorically, a rational equation is a function $\phi : E \longrightarrow X + F_\Sigma T_\Sigma(X + E)$ where F_Σ is the polynomial endofunctor associated to the signature and T_Σ is the initial $(X + E) + F$-algebra, as in (2). As we shall see in Section 6, all such equations have unique solutions in $T^\nu(X)$. For example, the equation `e = A(e)` has as its solution the rational term `A(A(...))`.

We can think of an equation of the form $e = t(e, \vec{x})$ as a finite tree with variables $\vec{x}$ and e pointing to the root of the system. For example, the equations `e=A(x,e)` and `e=A(x,A(x,e))` can be represented as follows:



Such trees are $X + F_\Sigma$-coalgebras on a finite set whose states are the nodes in the graph and whose structure map sends each state either to the term constructor labelling it and the child states, or to the variable labelling the node. Since equations are, intuitively, coalgebras with finitely presentable carrier, one may guess that the rational monad should be defined as the coproduct of all such coalgebras. However, the two coalgebras above have the same solution and hence define the same rational term. Categorically, these

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

coalgebras are bisimilar and hence the rational monad is the colimit of the inclusion of the full subcategory of $X + F$-coalgebras with finitely presentable carrier. Taking the full subcategory includes all coalgebra morphisms and hence quotients by bisimulation. Formally, set $\mathscr{I}_R : \mathscr{C} \longrightarrow \mathsf{Cat}$ to be the functor mapping an object X to the full subcategory of $X + F$-coalgebras whose underlying object is finitely presentable. The forgetful functor $U_X : \mathscr{I}_R(X) \longrightarrow \mathscr{C}$ provides the functor whose pointwise colimit we shall take.

Proposition 4.3. The assignment $X \mapsto RX = \mathrm{colim} U_X$ defines a monad.

Proof. Since this functor is finitary, we use Theorem 3.10 to prove rational terms form a monad. Functoriality of $\mathscr{I}_R$ is inherited from that of Φ, as introduced in Lemma 4.1. Similarly, a lifting on $\mathscr{I}_R$ comes as a restriction of the lifting L for Φ. Crucially for this, the coproduct of finitely presentable objects is finitely presentable, which means that $L(f)$, when applied to a coalgebra on a finitely presentable object, returns a coalgebra on a finitely presentable object. Since we use the full subcategory, all the components of the required natural transformations in the lifting L are still available. Finally, all the equational properties of the lifting still hold, hence we have a lifting for $\mathscr{I}_R$.

In order to apply Corollary 3.11, we still need to verify the three conditions expressed in our main theorem (Theorem 3.10). The existence of a natural transformation $\alpha = \mathsf{inl} : \mathsf{Id}_{\mathscr{I}X} \Rightarrow i_X^L$ ensures, by Lemma 3.4, that $\mathrm{colim} i_X^L = \mathrm{colim} \mathsf{Id} = 1$.

Now, given a map $f : X \longrightarrow RY$ in $\mathscr{C}$, where X and Y are finitely presentable, the coalgebra $L(f)(i_X)$ is defined by the composite

$$X + Y_0 \xrightarrow{\mathsf{inl}[f^+, Y_0]} Y_0 + FX \xrightarrow{i_f + FX} Y + FY_0 + FX \xrightarrow{Y + [F\mathsf{inl}, F\mathsf{inr}]} Y + F(Y_0 + X),$$

where f factors as $f = \overline{i_f} f^+$. If we take the map $k : L(f)(i_X) \longrightarrow i_f$ to be $[f^+, Y_0] : X + Y_0 \longrightarrow Y_0$, it is now clear that $Uk.f^L_{i_X} = f^+$.

Finally, let Z be finitely presentable, $g : Y \longrightarrow RZ$ be a map in $\mathscr{C}$, and $y : Y_0 \longrightarrow Y + FY_0$ be a $Y + F$-coalgebra. Then, for any $Y_0 + F$-coalgebra $\alpha : A \longrightarrow Y_0 + FA$ in $\mathscr{I}(Uy)$, the actions of the functors $L(g)L(y)$ and $L(L(g)y).\mathscr{I}(g^L_y)$ on α determine the same $Z + F$-coalgebra

$$\begin{array}{c} A + Y_0 + Z_0 \\ \downarrow{\scriptstyle [\alpha, Y_0, Z_0]} \\ Y_0 + Z_0 + FA \\ \downarrow{\scriptstyle (Y + Z_0 + [F\mathsf{inl}, F\mathsf{inr}])[y, Z_0, FA]} \\ Y + Z_0 + F(A + Y_0) \\ \downarrow{\scriptstyle (Z + [F\mathsf{inl}, F\mathsf{inr}])[i_g[g^+, Z_0], F(A + Y_0)]} \\ Z + F(A + Y_0 + Z_0) \end{array}$$

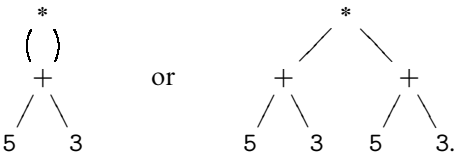
where g factors as $g = \overline{i_g} g^+$ with $U(i_g) = Z_0$, and we have omitted the obvious inclusions in the sums and deliberately rearranged the summands. The two corresponding components

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

of the natural transformations $g^L \diamond y^L$ and $(L(g)y)^L \diamond \mathsf{Id}$ turn out to be the inclusion $A \longrightarrow A + Y_0 + Z_0$, and hence are equal. □

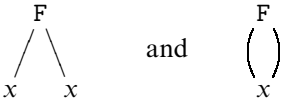
4.2. *The term graph monad*

In this section we shall work in the category $\mathscr{S}et$, and the functor F will be of the form F_Σ for some signature Σ, unless we explicitly say otherwise. Term graphs are widely used in computer science, as they are finite terms with loops and parallel edges, to model recursion and the sharing of subterms. For example, the term $(5+3)*(5+3)$ can be encoded as either



Clearly, in the first case the addition will be performed only once, whereas in the second case it will be evaluated twice.

When introducing them formally, people often ask for variables to be shared to the maximum possible extent. Here, we choose to relax this condition, allowing such terms as



to coexist. This allows users to choose for themselves how they handle the resources.

Formally, we define term graphs over a $\mathscr{S}et$-signature Σ as follows.

Definition 4.4. A *term graph* with variables in a set X is a 6-tuple (S, V, L_s, L_v, A, r) where:

— S and V are finite sets;
— L is a function from S to Σ;
— A is a function from S to C^* such that the length of $A(v)$ is the arity of $L(v)$, where $C = S + V$ and C^* is the set of finite words over C;
— L_v is a function from V to X;
— r is an element of C such that for any other state s in C there is a finite sequence $a_1, \ldots, a_n$ such that $a_1 = r$, $a_n = s$ and a_i is an element in the word $A(a_{i-1})$.

Here, C represents the set of *states* of the term graph, which naturally split as the union of those states labelled by term constructors, via L_s, and those labelled by variables, via L_v. The function A maps each state in S to the (ordered) set of its immediate child states and, clearly, there are as many children as the arity of the constructor labelling the state. Finally, the element $r \in C$ is a chosen *root* of the term graph, from which all the other states can be reached.

This data can be presented much more elegantly as an $X + F_\Sigma$-coalgebra with carrier C. The structure maps L_v, L_s and A clearly generate the structure map of the coalgebra

$$\gamma : C = V + S \longrightarrow X + F_\Sigma(V + S). \tag{3}$$

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

The information on the root transposes here to that of a generator.

Definition 4.5. Given an F-coalgebra (G,γ) for an endofunctor F on a category $\mathscr{C}$, the *subcoalgebra of* (G,γ) *generated by* S, written $< S >$, is the smallest subcoalgebra of (G,γ) containing S. Formally, if $i : S \longrightarrow G$ is the inclusion of S in G, then $< S >= (H,\xi)$ is such that there is a monic $j : S \longrightarrow H$ and a coalgebra monomorphism $h : (H,\xi) \longrightarrow (G,\gamma)$ for which $U(h)j = i$, and is universal amongst such.

In the particular situation we are dealing with now, the subcoalgebra generated by a subobject can be described explicitly in terms of reachable states. If (G,γ) is a coalgebra for the $\mathscr{S}et$-endofunctor F_Σ and $S \subset G$, then the coalgebra $< S >$ has as a carrier the set of all those states in G that can be reached starting from a state in S in the obvious sense. Thus the concept of root for a term graph has its categorical counterpart in the fact that (G,γ) is generated by r. In other words, the set of term graphs with variables from a set X is the set of $X + F$-coalgebras with a finite carrier, equipped with a choice of a specific generating element in the carrier. We shall call such a triple (g,G,γ) a *rooted coalgebra.*

Conversely, we can think of any rooted coalgebra as a term graph. Given (g,G,γ) with G finite, we can form the two pullbacks below:

$$\begin{array}{ccccc} G_v & \xrightarrow{\text{inl}} & G & \xleftarrow{\text{inr}} & G_s \\ \downarrow{\scriptstyle \gamma_v} & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \gamma_s} \\ X & \xrightarrow[\text{inl}]{} & X + FG & \xleftarrow[\text{inr}]{} & FG \end{array}$$

where, since $\mathscr{S}et$ is extensive, $G = G_v + G_s$. Note that, for the presence of roots, we cannot abstract from $\mathscr{S}et$ to any lfp extensive category (where the same construction could otherwise be performed).

To define the term graph monad via Theorem 3.10, we need to get $G(X)$ as a colimit. Intuitively, $\mathscr{I}X$ should have as objects rooted $X + F$-coalgebras with a finite carrier that the forgetful functor U_X maps to its root. Since we do not want to identify bisimilar term graphs, maps in $\mathscr{I}X$ should just be isomorphisms of coalgebras. Unfortunately, this is not a filtered category, but, fortunately, there is an easy fix. Let $\mathscr{I}'X$ be the category we have just described. Then, choose $\mathscr{I}X$ to be the completion of $\mathscr{I}'X$ under finite coproducts in $(X+F)$—coalg. In particular, the objects of $\mathscr{I}X$ are of the form $\Gamma = \sum_{i=1}^{n}(g_i, G_i, \gamma_i)$ where each (g_i, G_i, γ_i) is a rooted $X + F$-coalgebra with a finite carrier, and arrows are copairings of injections. Sometimes, we shall write a Γ as above in the form $\Gamma = (\{g_1,\ldots,g_n\}, G, \gamma)$, where $(G,\gamma) = \sum_{i=1}^{n}(G_i,\gamma_i)$ is a coproduct in $(X+F)$—coalg The functor U_X will map such a Γ to the set $\{g_1,\ldots,g_n\}$. In less categorical terms, we are now considering *term graph forests.*

That $\mathscr{I}X$ is filtered can easily be verified (Marchi 2003). To define the lifting, note that because $\mathscr{I}X$ is the free completion under finite coproducts of $\mathscr{I}'X$ in $(X+F)$—coalg, we can define the functor $L(\Gamma)$ as the free extension of a functor from $\mathscr{I}'(\{g_1,\ldots,g_n\})$ to $\mathscr{I}X$, where $\Gamma = (\{g_1,\ldots,g_n\}, G, \gamma)$ is an object of $\mathscr{I}X$. The action of this functor is

$$L(\Gamma)(a, A, \alpha) = (\widetilde{a}, \widetilde{A}, \widetilde{\alpha}),$$

where the triple $(\widetilde{a}, \widetilde{A}, \widetilde{\alpha})$ depends on whether a belongs to A_s or to A_v.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

If $a \in A_s$, we put $\widetilde{a} = a$,

$$\widetilde{A} = \sum_{a \in A_v} G_a + A_s,$$

where for each $a \in A_v$ $\alpha(a) = g_i$ for some i and G_a is the coalgebra G_i, and

$$\widetilde{\alpha} = \sum_{a \in A_v} G_a + A_s \xrightarrow{\sum \gamma_a + \alpha_s} X + F(\sum_{a \in A_v} G_a) + FA \xrightarrow{X + [F\mathsf{inl}, F(\widetilde{\alpha_v} + \mathsf{id})]} X + F(\sum_{a \in A_v} G_a + A_s),$$

where the function $\widetilde{\alpha_v} : A_v \longrightarrow \sum_{a \in A_v} G_a$ maps an element a to the element $\alpha(a) = \alpha_v(a)$ considered as an element of G_a. If $a \in A_v$, then $L(\Gamma)(a, A, \alpha) = (g_a, G_a, \gamma_a)$ where the subscript a is the specific i for which $\alpha(a) = g_i$.

The component of the natural transformation Γ^L corresponding to $(\{a_1, \ldots, a_n\}, A, \alpha)$ has to be a map of the form

$$\{a_1, \ldots, a_n\} \longrightarrow \{g_{a_i} \mid a_i \in A_v\} \cup \{a_i \mid a_i \in A_s\}.$$

This will map the element a_i to itself if $a_i \in A_s$, and to $g_{a_i} = \alpha(a_i)$ if $a_i \in A_v$ (where $i = 1, \ldots, n$). Naturality is easily checked.

Finally, given a morphism $\phi : \Gamma \longrightarrow \Delta$ in $\mathscr{I}X$, we need a natural transformation $\phi^L : L(\Gamma) \longrightarrow L(\Delta)\mathscr{I}U_X\phi$ such that $U_X(\phi^L)\Gamma^L = \Delta \diamond j^{U_X\phi}$. In fact, unravelling the definitions means that we can take ϕ^L to be the identity. That these data form a lifting is left as an exercise or can be found in Marchi (2003). Thus we have the following proposition.

Proposition 4.6. The functor sending X to $\mathrm{colim}\mathscr{I}X$ defines a monad.

4.3. *Coalgebraicity results*

We have constructed a general theorem for defining monads as pointwise colimits. Since our overall interest lies in guarded monads, the question arises as to whether it is possible to extend Theorem 3.10 to produce guarded monads. Here we provide some preliminary results showing that the term graph and rational monads are indeed guarded and the rational monad is strongly guarded.

Lemma 4.7. Let $F : \mathscr{C} \longrightarrow \mathscr{C}$ be a finitary functor and $\mathscr{I}X$ be a filtered category, and assume there is a functor $H : \mathscr{I}X \longrightarrow \mathscr{I}X$. If there is a natural transformation $\alpha : FU_X \Rightarrow U_XH$, then $\mathrm{colim}U_X$ is an F-algebra. Similarly, if there is a natural transformation $\alpha : U_X \Rightarrow FU_XH$, then $\mathrm{colim}U_X$ is an F-coalgebra.

Proof. Lemma 3.4 gives the map in the lax slice category $(H, \alpha) \in \mathsf{Lax}_C(FU_X, U_X)$, and hence $\mathrm{colim}\alpha : \mathrm{colim}FU_X = F\mathrm{colim}U_X \longrightarrow \mathrm{colim}U_X$, where the first equality is the finitariness of F. The second half of the theorem is proved analogously. □

We now prove that the rational and term graph monads are pointwise fixed points. To do this, we use the following property of coalgebras.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

Lemma 4.8. Let $F : \mathscr{C} \longrightarrow \mathscr{C}$ be a functor and X be an object of $\mathscr{C}$. Then, the mapping sending an $X + F$-coalgebra $h : S \longrightarrow X + FS$ to the $X + F$-coalgebra

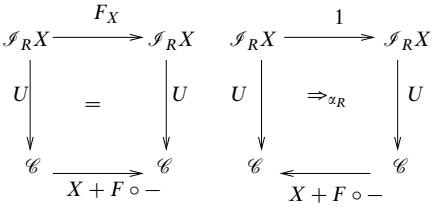
$$X + Fh : X + FS \longrightarrow X + F(X + FS)$$

defines a functor $F_X : (X+F)\text{–}\mathsf{coalg} \longrightarrow (X+F)\text{–}\mathsf{coalg}$. Also, $(X + F \circ -).U_X = U_X F_X$, there is a natural transformation $\alpha : U_X \Rightarrow (X + F \circ -).U_X$ and another natural transformation $\beta : 1 \Rightarrow F_X$ such that $\alpha \diamond 1 = U\beta$.

Proof. For the transformation α, for an $X + F$-coalgebra $\langle A, h\rangle$, set α_h to be the map h, which is clearly a map of the right form. Naturality of α is precisely the condition on a map $k : S \longrightarrow S'$ such that $k : (S, h) \longrightarrow (S', h')$ is a coalgebra homomorphism. Similarly, set $\beta_h = h$. □

Lemma 4.9. If F preserves finitely presentable objects, the rational monad is strongly F-guarded monad and the term graph monad is guarded.

Proof. We consider the rational monad, with the case of the term graph monad being analogous. The functor F_X preserves finite coalgebras and hence restricts to a functor $\mathscr{I}_R X \longrightarrow \mathscr{I}_R X$, which we also denote F_X. Similarly, α and β restrict to natural transformations α_R and β_R. Since the functor $X + F \circ -$ is finitary, by Lemma 4.7, RX is both an $X + F$-coalgebra via $\mathrm{colim}\alpha_R : RX \longrightarrow X + FRX$, and an $X + F$-algebra, with structure map $\mathrm{colim}1 : X + FRX \longrightarrow RX$.



We use Lemma 3.4 to show that these maps are mutually inverse. In one direction, $(\mathrm{colim}\alpha).(\mathrm{colim}1) = \mathrm{colim}(\alpha \diamond 1) = \mathrm{colim}(U\beta) = \mathrm{colim}1 = 1$, where the second equality is from Lemma 4.8 and all the others are from Lemma 3.4. In the reverse direction, calculating the pasting gives $1 \diamond \alpha_R = \alpha_R F_X = (X + F)U\beta$, where the last equality holds because both natural transformations have, as component for a coalgebra h, the arrow $X + Fh$. Thus

$$\begin{aligned}(\mathrm{colim}1).(\mathrm{colim}\alpha) &= \mathrm{colim}1 \diamond \alpha = \mathrm{colim}\alpha F_X = \mathrm{colim}(X + F)U\beta \\ &= X + F(\mathrm{colim}U\beta) = X + F(1) = 1,\end{aligned}$$

where the fourth equality holds because $X + F$ is finitary.

The family of maps making RX an $X + F$-coalgebra can be seen to be natural by noting that Rf arises via Lemma 3.4 as $\mathrm{colim}f^R$, and $X + Ff$ arises as $\mathrm{colim}X + F(f^R)$ where $f^R = i_Y^R \diamond 1$. Thus, commutation of the naturality square amounts to proving that $\alpha \diamond f^R = X + F(f^R) \diamond \alpha$. At an $X + F$-coalgebra (S, h), we have $(\alpha \diamond f^R)_h = L_{i_Y}.\mathsf{inl}$, while $(X + F(f^R) \diamond \alpha)_h = X + F(\mathsf{inl}).h$. The definition of L_{i_Y} shows that these are equal. The

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

naturality of the maps making RX an $X + F$-algebra follows from the naturality of the maps making RX an $X + F$-coalgebra and the fact that these maps are mutually inverse.

The first component of the algebra structure is the unit of R since we have a map of the form $\beta_{i_X} : i_X \longrightarrow F_X(i_X)$, which induces the second of the following equalities $\mathsf{colim1.inl} = \overline{F_X(i_x)} = \overline{i_X} = \eta$. Finally, equation (1) is another diagram chase. □

5. Recursion over coalgebraic monads

We have seen how rational terms arise as solutions of equations represented categorically as maps $\phi : E \longrightarrow T_\Sigma(X + E)$ for fixed sets X and E. A solution for ϕ is a map $\phi^\dagger : E \longrightarrow T^v(X)$ such that the appropriate diagram commutes. In this section we show that this ability to solve equations is possessed by all strongly F-guarded monads, not just the initial one.

Definition 5.1. Let H be a strongly F-guarded monad. An *H-rational equation* is a map $\phi : E \xrightarrow{\phi} X + FH(X + E)$.

A *solution* for an H-rational equation ϕ is a map $\phi^\dagger : E \longrightarrow T^v(X)$ for which

$$\begin{array}{ccc} E & \xrightarrow{\phi} & X + FH(X + E) \\ & \searrow^{\phi^\dagger} \quad \swarrow_{\psi} & \\ & T^v(X) & \end{array}$$

where $\psi.\mathsf{inl} = \eta$ and $\psi.\mathsf{inr}$ is obtained as the composite

$$FH(X + E) \xrightarrow{F!} FT^v(X + E) \xrightarrow{FT^v[\eta,\phi^\dagger]} FT^{v2}(X) \xrightarrow{F\mu} FT^v \longrightarrow T^v.$$

Notice that each H-rational equation $\phi : E \xrightarrow{\phi} X + FH(X + E)$ determines an $X + F$-coalgebra structure for $H(X + E)$:

$$H(X + E) \xrightarrow{[\eta,\alpha]^{-1}} X + E + FH(H + E) \xrightarrow{[\mathsf{inl},\phi,\mathsf{inr}]} X + FH(H + E).$$

Moreover, there is a bijective correspondence between solutions $E \longrightarrow T^v(X)$ and $X + F$-coalgebra morphisms $H(X + E) \longrightarrow T^v(X)$. In one direction, we simple restrict a coalgebra morphism with the canonical inclusion $E \longrightarrow H(X + E)$, while given a map $\psi : E \longrightarrow T^v(X)$, we can construct a map $H(X + E) \xrightarrow{H[\eta,\psi]} HT^v(X) \longrightarrow T^v(X)$. It is then simple diagram chasing to show that, under these constructions, being a solution corresponds to being a coalgebra map. We believe that the formulation of a solution via coalgebra maps is both cleaner and conceptually simpler that the usual formulation. Either way, since $T^v(X)$ is the final $X + F$-coalgebra, we have proved the following lemma.

Lemma 5.2. Every H-rational equation map $E \xrightarrow{\phi} X + FH(X + E)$ has a unique solution $\phi^\dagger : H(X + E) \longrightarrow T^v(X)$.

We can go further and build strongly F-guarded monad morphisms into the picture.

Lemma 5.3. Let $\psi : H \longrightarrow H'$ be a morphism of strongly F-guarded monads. Then there is a map ψ^* sending H-rational equations to H'-rational equations such that for all H-rational equations ϕ, we have $\phi^\dagger = (\psi^*(\phi))^\dagger.\psi$

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press

Proof. Given an H-rational equation $E \xrightarrow{\phi} X + FH(X + E)$, set $\psi^*(\phi)$ to be the H'-rational equation $E \xrightarrow{\phi} X + FH(X + E) \xrightarrow{X+F\psi} X + FH'(X + E)$. Next note that ψ defines a coalgebra morphism between the coalgebras $[\mathsf{inl}, \phi, \mathsf{inr}]$ and $[\mathsf{inl}, X + F\psi.\phi, \mathsf{inr}]$, which generate the solutions of ϕ and $\psi^*(\phi)$. By the finality of $T^v(X)$, we therefore have $\phi^\dagger = (\psi^*(\phi))^\dagger.\psi$ □

The importance of this result is that it gives a simple, yet precise, proof of part of the implementation of functional languages using term graphs. For example, suppose we write `c = A(x,c,c)` in a functional language. This is an equation written in the free monad, which has as solution an infinite term, which could be taken to be its semantics. However, a compiler would turn this equation into the equation over term graphs mapping x to the term graph for `A(x,c,c)` that shares the two copies of c. Thus we must ask how the solution to this term graph equation compares with the solution to the original equation. The solution theorems above, in fact, require a $1 + F \circ _$-coalgebra structure on the monad T, and such a coalgebra structure is possessed by the term graph monad. Thus Lemma 5.3 assures us that the compiler will behave correctly, in that solving the equation directly or solving the associated term graph equation produces the same result.

References

Aczel, P., Adámek, J. and Velebil, J. (2001) A coalgebraic view of infinite trees and iteration. In: Corradini, A., Lenisa, M. and Montanari, U. (eds.) Proceedings of CMCS'01. *ENTCS* **44** (1).

Adamek, J., Milius, S. and Velebil, J. (2003) Free iterative theories: a coalgebraic view. *Mathematical Structures in Computer Science* **13** 259–320.

Adamek, J. and Rosický, J. (1994) Locally Presentable and Accessible Categories. *London Mathematical Society Lecture Note Series* **189**, Cambridge University Press.

Corradini, A. and Gadducci, F. (1997) A 2-categorical presentation of term graph rewriting. In: Proceedings CTCS'97. *Springer-Verlag Lecture Notes in Computer Science* **1290**.

Courcelle, B. (1983) Fundamental properties of infinite trees. *Theoretical Computer Science* **25** (2) 95–169.

Dubuc, E. J. and Kelly, G. M. (1983) A presentation of topoi as algebraic relative to categories or graphs. *Journal of Algebra* **81** 420–433.

Elgot, C., Bloom, S. and Tindell, R. (1978) On the algebraic structure of rooted trees. *Journal of Computing System Sicences* **16** 361–399.

Fiore, M., Plotkin, G. and Turi, D. (1999) Abstract syntax and variable binding. In: *Proc. LICS'99*.

Ghani, N., Lüth, C., Marchi, F. D. and Power, J. (2001) Algebras, coalgebras, monads and comonads. In: Corradini, A., Lenisa, M. and Montanari, U. (eds.) Proceedings of CMCS'01. *ENTCS* **44** (1).

Kelly, G. M. and Power, A. J. (1993) Adjunctions whose counits are coequalizers, and presentations of finitary monads. *Journal for Pure and Applied Algebra* **89** 163–179.

Lüth, C. and Ghani, N. (1997) Monads and modular term rewriting. In: Category Theory in Computer Science CTCS'97. *Springer-Verlag Lecture Notes in Computer Science* **1290** 69–86.

Mac Lane, S. (1971) *Categories for the Working Mathematician*, Springer-Verlag.

Marchi, F. D. (2003) *Monads in Coalgebra.* Ph.D. Thesis, University of Leicester (submitted).

Moss, L. S. (2001) Parametric corecursion. *Theoretical Computer Science* **260** (1-2) 139–163.

https://doi.org/10.1017/S0960129505004743 Published online by Cambridge University Press