Under consideration for publication in Math. Struct. in Comp. Science

Monads of Coalgebras: Rational Terms and Term Graphs

NEIL GHANI[1], CHRISTOPH LÜTH[2],

FEDERICO DE MARCHI[1 †]

[1] *Department of Mathematics and Computer Science, University of Leicester*

[2] *FB 3 — Mathematik und Informatik, Universität Bremen*

Received 12 May 2003

This paper introduces coalgebraic 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 coalgebraic monads which allows us to generalise the notion of rational term and term graph to categories other than Set. 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 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,

† Research supported by EPSRC grant GRM96230/01: *Categorical Rewriting: Monads and Modularity*

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, ie 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, 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, e.g. 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, 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 the computational perspective one may argue that one should consider only those infinite terms which 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 within the coalgebra community in this direction. Indeed, Moss (Moss,) 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 and thereby meeting our requirement for substitution to be taken into account. These results were independently discovered in (Aczel et al., 2001) and in (Ghani et al., 2001) although the latter paper uses a more restrictive setting. In dependently of ourselves, (Adamek et al., 2001) 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 don't want to tackle each term algebra on a case-by-case basis. This paper introduces *coalgebraic monads* as a uniform framework for such term algebras. In detail, we

— introduce the notion of a coalgebraic 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 coalgebraic monads;
— solve rational equations whose right-hand sides are not just finite terms but belong to

any coalgebraic 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, eg 2-categories, double categories etc are 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.

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 while Section 5 contains our partial correctness result.

2. Coalgebraic Monads

Let $F : \mathcal{C} \longrightarrow \mathcal{C}$ 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, then we take the following properties as 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 a $(\mathsf{Id} + F \circ -) : [\mathcal{C}, \mathcal{C}] \longrightarrow [\mathcal{C}, \mathcal{C}]$-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 a $(\mathsf{Id} + F \circ -)$-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:

Lemma 2.1. Let (T, η, μ) be a monad and F an endofunctor on a category $\mathcal{C}$. There is a bijection between natural transformations $\tau : F \longrightarrow T$ and natural transformations

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

$\alpha : FT \longrightarrow T$ making the following diagram commute

$$\begin{tikzcd} FTT \arrow[r, "F\mu"] \arrow[d, "\alpha_T"'] & FT \arrow[d, "\alpha"] \\ TT \arrow[r, "\mu"'] & T. \end{tikzcd} \tag{1}$$

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

$$\begin{tikzcd} FTT \arrow[r, "\tau_{TT}"] \arrow[d, "F\mu"'] & TTT \arrow[r, "\mu_T"] \arrow[d, "T\mu"] & TT \arrow[d, "\mu"] \\ FT \arrow[r, "\tau_T"'] & TT \arrow[r, "\mu"'] & T. \end{tikzcd}$$

Conversely, given α, define $\tau = \alpha.F\eta$. The two mappings are easily shown to be mutually inverse. □

Condition (1) says just that, if we think of α as transforming F terms over T into T terms, it doesn't 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 $\mathcal{C}$ is a 4-tuple (T, η, μ, τ) such that (T, η, μ) is a monad on $\mathcal{C}$ 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 (**?**).

Lemma 2.3. Let (T, η, μ, τ) be an F-guarded monad on $\mathcal{C}$. 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, TX is naturally an $X + F$-algebra. We shall call *strong* those monads for which this algebra structure is invertible.

Definition 2.4. Let F be an endofunctor on a category $\mathcal{C}$. A *strongly F-guarded monad* on $\mathcal{C}$ 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.

Now for some examples.

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

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 $\psi : T \longrightarrow 1 + FT$. That ϕ and ψ are mutually inverse are diagram chases 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 $(\mathsf{Id} + F \circ -) : [\mathcal{C}, \mathcal{C}] \longrightarrow [\mathcal{C}, \mathcal{C}]$ 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 $F : \mathcal{C} \longrightarrow \mathcal{C}$ be and let T^ν be the final $(\mathsf{Id} + F \circ -)$-coalgebra. Then T^ν is the final strongly F-guarded monad.

Proof. 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, 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 a F-guarded monad morphism, as one can prove with a bit of diagram chasing (**?**) 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 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. This way, one can show that the following are all strongly F-guarded monads over Set: i) infinite terms which contain only a finite number of variables; ii) locally finite terms (Courcelle, 1983), i.e. finite and infinite terms which 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 which are not subalgebras of T^ν, eg term graphs which model recursion and sharing via 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.

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 straight forwardly to functors with higher rank. The relationship between signatures and their representing monads can be generalised to *lfp-categories*, i.e. 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 $\mathcal{C}$ is an lfp-category, let $\mathcal{C}_{\mathrm{fp}}$ be the full subcategory of finitely presentable objects with inclusion $J : \mathcal{C}_{\mathrm{fp}} \longrightarrow \mathcal{C}$. A functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ is *finitary* if it preserves filtered colimits, or equivalently if it is isomorphic to the left Kan extension along J of its restriction to $\mathcal{C}_{\mathrm{fp}}$. Hence, the category $\mathit{Fin}[\mathcal{C}, \mathcal{D}]$ of finitary functors from $\mathcal{C}$ to $\mathcal{D}$ is equivalent to the functor category $[\mathcal{C}_{\mathrm{fp}}, \mathcal{D}]$.

In the case of $\mathcal{D} = \mathcal{C}$, the category of finitary endofunctors is monoidal with unit the identity and multiplication given by composition. Similarly $[\mathcal{C}_{\mathrm{fp}}, \mathcal{C}]$ is also monoidal with unit the inclusion J and with multiplication given by $F \Box G = \mathsf{Lan}_J F \circ G$. Now, to give a finitary monad on $\mathcal{C}$ is to give a monoid in $\mathit{Fin}[\mathcal{C}, \mathcal{C}]$ which is thus a monoid in the equivalent category $[\mathcal{C}_{\mathrm{fp}}, \mathcal{C}]$ (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 $[\mathcal{C}_{\mathrm{fp}}, \mathcal{C}]$ 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 fro the monoidal category $[\mathcal{C}_{\mathrm{fp}}, \mathcal{C}]$.

Definition 3.1. A *Kleisli monoid* for an lfp category $\mathcal{C}$ consists of
— a function T assigning to each object X in $\mathcal{C}_{\mathrm{fp}}$ an object TX in $\mathcal{C}$;
— for each X in $\mathcal{C}_{\mathrm{fp}}$ a map $\eta_X : X \longrightarrow TX$ in $\mathcal{C}$;
— for objects X, Y in $\mathcal{C}_{\mathrm{fp}}$, a lifting function $\mathrm{s}_{X,Y} : \mathcal{C}(X, TY) \longrightarrow \mathcal{C}(TX, TY)$;
satisfying the following conditions:

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

In the rest of the paper, we shall omit the subscript to the function s, whenever possible. The definition doesn't substantially differ from that of a Kleisli triple for a category $\mathcal{C}$ and, not surprisingly, defines a monoid in the category $[\mathcal{C}_{\mathrm{fp}}, \mathcal{C}]$ and hence a monad on $\mathcal{C}$. In doing so we make use of the well known formula for Kan extensions in terms of coends and wedges. See (**?**) for a reference to this material.

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

Proof. Firstly, T extends to a functor $T : \mathcal{C}_{\mathrm{fp}} \longrightarrow \mathcal{C}$ 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

$$(T \Box T)S = \mathsf{Lan}_J T(TS) = \int_{n \in \mathcal{C}_{\mathrm{fp}}} \mathcal{C}(n, TS) \otimes Tn \longrightarrow TS$$

we therefore require a wedge $w_n : \mathcal{C}(n, TS) \otimes Tn \longrightarrow TS$ which, via the universal property of the tensor, is a family of maps $\mathcal{C}(n, TS) \longrightarrow \mathcal{C}(Tn, TS)$ 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 () for more details. $\Box$

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 systemize the proofs that rational terms and term graphs form monads. Consequently the interested reader may prefer to skim this section, consider the examples and then reread this section in greater depth.

First, we abstract the data we have. Firstly, for every object X, we are interested in a subcategory of the category of $X + F$-coalgebras. This we model by as a functor $\mathcal{I} : \mathcal{C} \longrightarrow \mathsf{Cat}$. Of key importance is the forgetfull functors U_X from the (sub)-category of $X + F$-coalgebras to $\mathcal{C}$. If we denote by $\mathsf{K}_{\mathcal{C}} : \mathcal{C} \longrightarrow \mathsf{Cat}$ the constant functor mapping each object to $\mathcal{C}$, then the collection of forgetfull functors forms a natural transformation $U : \mathcal{I} \Rightarrow \mathsf{K}_{\mathcal{C}} : \mathcal{C} \longrightarrow \mathsf{Cat}$. 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 $\mathcal{C}_{\mathrm{fp}}$. However, first we introduce some notation concerning colimits since colimits are central in what follows.

Definition 3.3. Let $\mathcal{C}$ be a 2-category and X an object of $\mathcal{C}$. The *lax slice 2-category* Lax_X has

- Object of Lax_X are maps $f : Y \longrightarrow X$ in $\mathcal{C}$.
- Arrows $\alpha : f \longrightarrow g$ of Lax_X are pairs $\langle h, \alpha \rangle$ where $\alpha : f \Rightarrow gh$ in $\mathcal{C}$. If $\alpha : f \longrightarrow g$ and $\beta : g \longrightarrow h$ are morphisms in Lax_X, we write their composite as $\beta \Diamond \alpha : f \longrightarrow h$.
- 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 $\mathcal{C}$ such that $\alpha' = g\theta.\alpha$.

The usual definition of a slice category $\mathcal{C}/X$ is the lax slice category on the 2-category obtained by adding only the identity 2-cells to $\mathcal{C}$. Henceforth, given a diagram D with colimit X, we shall indicate with $\overline{d}$ 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 $\mathcal{C}$ be a cocomplete category and consider the lax slice category $\mathsf{Lax}_{\mathcal{C}}$ built over the 2-category Cat. The colimit operation defines a 2-functor $\mathrm{colim} : \mathsf{Lax}_{\mathcal{C}} \longrightarrow \mathcal{C}$ where we regard $\mathcal{C}$ as the 2-category obtained from $\mathcal{C}$ by adding identity 2-cells.

Unwinding Lemma 3.4 means that if $\langle H, \alpha \rangle \in \mathsf{Lax}_{\mathcal{C}}(F, G)$, then 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 $\alpha \longrightarrow \alpha'$, then $\mathrm{colim}\alpha = \mathrm{colim}\alpha'$.

3.3. *Obtaining the Data for a Kleisli Monoid*

So, 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 $X \in \mathcal{C}_{\mathrm{fp}}$ there is an object $i_X \in \mathcal{I}X$ 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 accociated cocone into TY. Since X is finitely presentable, X can be actually mapped to TY_0 for some subobject Y_0 of Y and then adding Y_0 to the carrier of each $X+F$-coalgebra, thereby making it into a $Y+F$ one. Formally, this procedure is captured by what we call a lifting:

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

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

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

$$
\begin{tikzcd}
\mathcal{I}Ug \arrow[rr, "L(g)"] \arrow[dr, "U"'] & \arrow[d, phantom, "\Rightarrow_{g^L}"] & \mathcal{I}X \arrow[dl, "U"] \\
& \mathcal{C} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{I}Ug \arrow[rr, "\mathcal{I}Uk"] \arrow[dr, "L_g"'] & \arrow[d, phantom, "\Rightarrow_{k^L}"] & \mathcal{I}Uh \arrow[dl, "L_h"] \\
& \mathcal{I}X &
\end{tikzcd}
$$

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 $\mathsf{Lax}_\mathcal{C}$. Thus we may form the lax slice category Lax_{U_X} which (by ignoring the 2-cells) defines a functor $\mathsf{Lax}_{U\_} : \mathcal{C} \longrightarrow \mathsf{Cat}$. A lifting is then a natural transformation $\mathcal{I} \longrightarrow \mathsf{Lax}_{U\_}$. As usual, see (**?**) for more details.

Lemma 3.6. Let X and Y be finitely presentable, $\mathcal{I}Y$ is a filtered category and $\mathcal{I}$ 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 $\mathcal{I}Y$ is filtered, f factors as $f = \overline{i_f}.f^+$ for some $i_f \in \mathcal{I}Y$. This defines a functor $Lf = L(i_f)\mathcal{I}(f^+)$ and also a natural transformation $f^L = i_f^L \lozenge 1$

$$
\begin{tikzcd}
X \arrow[rr, "f"] \arrow[drr, "f^+"'] & & \mathrm{colim} U_Y \\
& & U_Y(i_f) \arrow[u, "\overline{i_f}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{I}X \arrow[r, "\mathcal{I}(f^+)"] \arrow[dr, "U"'] & \mathcal{I}U(i_f) \arrow[r, "L_{i_f}"] \arrow[d, "U"] \arrow[dl, phantom, "\Rightarrow_1"] & \mathcal{I}Y \arrow[dl, "U"] \arrow[dll, phantom, "\Rightarrow_{i_f^L}"] \\
& \mathcal{C} &
\end{tikzcd}
$$

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 $f = \overline{i_f}.f^+$. This is actually not the case. In fact, suppose $f = \overline{i_h}.h$ is another factorisation, inducing the functor and natural transformation

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

By lemma 3.4, we have a map $\mathrm{colim}(i_h^L \lozenge 1) : TX \longrightarrow TY$. Because $\mathcal{I}Y$ 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's focus on $k : i_f \longrightarrow i$. The lifting L determines $k^L : L(i_f) \Rightarrow L(i).\mathcal{I}Uk$. Then

$$i^L \lozenge 1 = i^L \mathcal{I}(Uk)\mathcal{I}(f^+) = (Uk^L.i_f^L)\mathcal{I}(f^+) = Uk^L\mathcal{I}(f^+).i_f^L\mathcal{I}(f^+)$$

where the first equality comes from $\mathcal{I}(g) = \mathcal{I}(Uk)\mathcal{I}(f^+)$, the second is from the lifting of k^L and the third is the distribution of horizontal over vertical composition of natural transformations. Thus $i^L \lozenge 1 = Uk^L\mathcal{I}(f^+).i_f^L \lozenge 1$, and by lemma 3.4, we have that $\mathrm{colim}(i^L \lozenge 1) = \mathrm{colim}(i_f^L \lozenge 1)$. Analogously, $\mathrm{colim}(i^L \lozenge 1) = \mathrm{colim}(i_h^L \lozenge 1)$, hence $\mathrm{s}(f)$ doesn't 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 $L(f) = L(i_f).\mathcal{I}(f^+)$ and $f^L = i_f^L\mathcal{I}(f^+)$. 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, $\mathrm{s}(\eta) = \mathrm{colim}(i_X^L \lozenge 1) = \mathrm{colim} i_X^L \mathrm{colim} 1 = \mathrm{colim} i_X^L$. This verifies the first law for a Kleisli monoid.

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

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 $\mathcal{I}Y$ such that $Uk.f_{i_X}^L = f^+$, 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, i.e. $\mathrm{s}(f).\eta = \overline{L(f)(i_X)}.f_{i_X}^L$. Thus:

$$\begin{tikzcd}
X = U_X(i_X) \arrow[r, "f_{i_X}^L"] \arrow[dr, "f^+"'] & U_Y L(f)(i_X) \arrow[d, "Uk"] \arrow[dr, "\overline{L(f)(i_X)}"] & \\
& U_Y(i_f) \arrow[r, "\overline{i_f}"'] & \mathrm{colim} U_Y
\end{tikzcd}$$

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. Consider two maps $f : X \longrightarrow \mathrm{colim} U_Y$ and $g : Y \longrightarrow \mathrm{colim} U_Z$. Assume, for all $y \in \mathcal{I}Y$,

$$
\begin{tikzcd}
\mathcal{I}Uy \arrow[r, "L(y)"] \arrow[d, "\mathcal{I}(g^L_y)"'] & \mathcal{I}Y \arrow[d, "L(g)"] \\
\mathcal{I}U(L(g)y) \arrow[r, "L(L(g)y)"'] & \mathcal{I}Z
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
\mathcal{I}Uy \arrow[r, "L_y"] \arrow[dr, "U"'] \arrow[r, phantom, shift right=3ex, "\Rightarrow_{y^L}"] & \mathcal{I}Y \arrow[r, "L_g"] \arrow[d, "U"] \arrow[r, phantom, shift right=3ex, "\Rightarrow_{g^L}"] & \mathcal{I}Z \arrow[dl, "U"] \\
& \mathcal{C} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{I}Uy \arrow[r, "\mathcal{I}(f^L_y)"] \arrow[dr, "U"'] \arrow[r, phantom, shift right=3ex, "\Rightarrow_1"] & \mathcal{I}U(L_g(y)) \arrow[r, "L(L_g(y))"] \arrow[d, "U"] \arrow[r, phantom, shift right=3ex, "\Rightarrow_{(L_g y)^L}"] & \mathcal{I}Z \arrow[dl, "U"] \\
& \mathcal{C} &
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, "f^+"'] & TY \arrow[r, "\mathrm{s}(g)"] & TZ \\
& Ui_f \arrow[u, "\overline{i_f}"'] \arrow[r, "g^L_{i_f}"'] & U(L(g)(i_f)) \arrow[u, "\overline{L(g)(i_f)}"']
\end{tikzcd}
$$

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

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

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

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

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

— there is an object i_X in $\mathcal{I}X$ such that $U_X(i_X) = X$ and $\mathrm{colim} i^L_X = 1$;
— there is a map $k : L(f)(i_X) \longrightarrow i_f$ such that $Uk.f^L_{i_X} = f^+$;

— for all y in $\mathcal{I}Y$, $g^L \Diamond y^L = (L(g)y)^L \Diamond 1$ and

$$\begin{tikzcd}
\mathcal{I}Uy \arrow[r, "L(y)"] \arrow[d, "\mathcal{I}(g_y^L)"'] & \mathcal{I}Y \arrow[d, "L(g)"] \\
\mathcal{I}U(L(g)y) \arrow[r, "L(L(g)y)"'] & \mathcal{I}Z.
\end{tikzcd}$$

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 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 $\mathcal{C}_{\mathrm{fp}}$ into $\mathcal{C}$ is a finitary monad on $\mathcal{C}$.

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 $U_X : (X+F){-}\mathsf{coalg} \longrightarrow \mathcal{C}$ for the forgetful functor sending each coalgebra to its carrier. U_X creates colimits and since $\mathcal{C}$ is lfp, and hence cocomplete, any functor $J : D \longrightarrow (X{+}F){-}\mathsf{coalg}$ from a small category $\mathcal{D}$ has a colimit and $U_X \mathrm{colim} J = \mathrm{colim} U_X J$. Next,

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

Proof. If $f : X \longrightarrow Y$ is a map in $\mathcal{C}$, $\Phi(f)$ 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 are easily proved. Naturality holds since $U_X = U_Y \Phi(f)$. □

In general, we shall instantiate theorem 3.10 with functors which 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.

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 $L(g) : \Phi U g \longrightarrow \Phi X$ and the action of $L(g)$ on morphisms is easily defined. 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

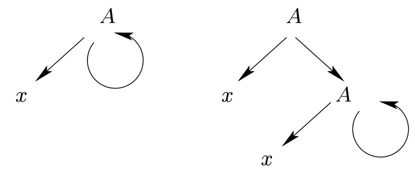
$$\begin{tikzcd}
A + S \arrow[r, "L(g)(m)"] \arrow[d, "1+k"'] & X + F(A + S) \arrow[d, "1+F(1+Uk)"] \\
A + S' \arrow[r, "L(h)(\Phi k(m))"'] & X + F(A + S')
\end{tikzcd}$$

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

4.1. *The Rational Monad*

Given a signature Σ, rational terms are the set of finite and infinite terms which 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 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 to either 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 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 $\mathcal{I}_R : \mathcal{C} \longrightarrow \mathsf{Cat}$ 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 $U_X : \mathcal{I}_R(X) \longrightarrow \mathcal{C}$ provides the functor whose pointwise colimit we shall take.

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

Proof. Since this functor is finitary, we use theorem 3.10 to prove rational terms form a monad. Functoriality of $\mathcal{I}_R$ is inherited from that of Φ as introduced in lemma 4.1. Similarly, a lifting on $\mathcal{I}_R$ 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 $\mathcal{I}_R$.
In order to apply corollary 3.11, we still need to verify the three conditions expressed in our main theorem 3.10. The existence of a natural transformation $\alpha = \mathsf{inl} : \mathsf{Id}_{\mathcal{I}X} \Rightarrow i^L_X$ ensures, by lemma 3.4, that $\text{colim} i^L_X = \text{colim}\mathsf{Id} = 1$.
Now, given a map $f : X \longrightarrow RY$ in $\mathcal{C}$, where X and Y are finitely presentable, the coalgebra $L(f)(i_X)$ is defined by the composite

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

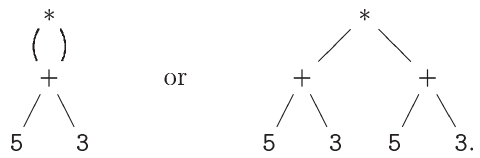
where f factors as $f = \overline{i_f} f^+$. If we take the map $k : L(f)(i_X) \longrightarrow i_f$ to be $[f^+, Y_0] : X + Y_0 \longrightarrow Y_0$, it is now clear that $Uk.f^L_{i_X} = f^+$.
Finally, let Z be finitely presentable, $g : Y \longrightarrow RZ$ be a map in $\mathcal{C}$, and $y : Y_0 \longrightarrow Y + FY_0$ be a $Y + F$-coalgebra. Then, for any $Y_0 + F$-coalgebra $\alpha : A \longrightarrow Y_0 + FA$ in $\mathcal{I}(Uy)$, the actions of the functors $L(g)L(y)$ and $L(L(g)y)\mathcal{I}(g^L_y)$ on α determine the same $Z + F$-coalgebra

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

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 summonds. The two corresponding components of the natural transformations $g^L \lozenge y^L$ and $(L(g)y)^L \lozenge \mathsf{Id}$ turn out to be the inclusion $A \longrightarrow A + Y_0 + Z_0$, hence being equal. $\square$

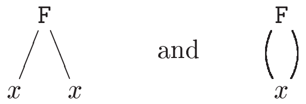
4.2. *The Term Graph Monad*

In this section we shall work in the category $\mathcal{S}et$, and the functor F will be of the form F_Σ for some signature Σ, unless otherwise explicitly said. 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 the user to choose for oneself how to handle the resources.

Formally, we define term graphs over a $\mathcal{S}et$-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

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

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 $\mathcal{C}$, 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 $\mathcal{S}et$-endofunctor F_Σ and $S \subset G$, then the coalgebra $< S >$ has as a carrier the set of all those states in G which 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:

$$\begin{tikzcd}
G_v \arrow[r, "\mathsf{inl}"] \arrow[d, "\gamma_v"'] & G \arrow[d, "\gamma"] & G_s \arrow[l, "\mathsf{inr}"'] \arrow[d, "\gamma_s"] \\
X \arrow[r, "\mathsf{inl}"'] & X + FG & FG \arrow[l, "\mathsf{inr}"]
\end{tikzcd}$$

where, since $\mathcal{S}et$ is extensive, $G = G_v + G_s$. Note that, for the presence of roots, we can not abstract from $\mathcal{S}et$ 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, $\mathcal{I}X$ should have as objects rooted $X+F$-coalgebras with a finite carrier which the forgetful functor U_X maps to its root. Since we don't want to identify bisimilar term graphs, maps in $\mathcal{I}X$ should just be isomorphisms of coalgebras. Unfortunately, this is not a filtered category but, fortunately, there is an easy fix. Let $\mathcal{I}'X$ be the category we just described. Then, choose $\mathcal{I}X$ to be the completion of $\mathcal{I}'X$ under finite coproducts in $(X{+}F){-}\mathsf{coalg}$. In particular, the objects of $\mathcal{I}X$ 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 copairing 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 $(X{+}F){-}\mathsf{coalg}$ The functor U_X will map such a Γ to the set $\{g_1,\dots,g_n\}$. In less categorical terms, we are now considering *term graph forests.*

That $\mathcal{I}X$ is filtered can easily be verified (**?**). To define the lifting, note that because $\mathcal{I}X$ is the free completion under finite coproducts of $\mathcal{I}'X$ in $(X+F)-\mathsf{coalg}$, we can define the functor $L(\Gamma)$ as the free extension of a functor from $\mathcal{I}'(\{g_1,\dots,g_n\})$ to $\mathcal{I}X$, where $\Gamma=(\{g_1,\dots,g_n\},G,\gamma)$ is an object of $\mathcal{I}X$. 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.

If $a\in A_s$, then 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 $(\{a_1,\dots,a_n\},A,\alpha)$ has to be a map of the form

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

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 $i=1,\dots,n$). Naturality is easily checked.

Finally, given a morphism $\phi:\Gamma\longrightarrow\Delta$ in $\mathcal{I}X$, we need a natural transformation $\phi^L:L(\Gamma)\longrightarrow L(\Delta)\mathcal{I}U_X\phi$ such that $U_X(\phi^L)\Gamma^L=\Delta\Diamond j^{U_X\phi}$. Infact, unravelling the definitions means that we can take ϕ^L to be the identity. That these data form a lifting is left either as an exercise or can be found in (**?**). Thus we have

Proposition 4.6. The functor sending X to $\mathrm{colim}\mathcal{I}X$ 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 preliminary results showing that the term graph and rational monads are indeed guarded and the rational monad is strongly guarded.

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

Proof. By lemma 3.4, we have 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 finitaryness 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

Lemma 4.8. Let $F : \mathcal{C} \longrightarrow \mathcal{C}$ be a functor and X and object of $\mathcal{C}$. 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 $F_X : (X+F){-}\mathsf{coalg} \longrightarrow (X+F){-}\mathsf{coalg}$. Furthermore, $(X+F\circ -).U_X = U_X F_X$, 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 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 $\mathcal{I}_R X \longrightarrow \mathcal{I}_R X$ 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 $\mathrm{colim}1 : X + FRX \longrightarrow RX$.

$$\begin{tikzcd}
\mathcal{I}_R X \arrow[r, "F_X"] \arrow[d, "U"'] \arrow[dr, phantom, "="] & \mathcal{I}_R X \arrow[d, "U"] & \mathcal{I}_R X \arrow[r, "1"] \arrow[d, "U"'] \arrow[dr, phantom, "\Rightarrow\alpha_R"] & \mathcal{I}_R X \arrow[d, "U"] \\
\mathcal{C} \arrow[r, "X + F \circ -"'] & \mathcal{C} & \mathcal{C} & \mathcal{C} \arrow[l, "X + F \circ -"]
\end{tikzcd}$$

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 $1\lozenge\alpha_R = \alpha_R F_X = (X+F)U\beta$, where the last equality holds since both natural transformations have, as component for a coalgebra h, the arrow $X + Fh$. Thus

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

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 $f^R = i_Y^R\lozenge 1$. Thus, commutation of the naturality square amounts to proving that $\alpha\lozenge f^R = X + F(f^R)\lozenge\alpha$. At an $X + F$-coalgebra (S, h), $(\alpha\lozenge f^R)_h = L_{i_Y}.\mathsf{inl}$ while $(X + F(f^R)\lozenge\alpha)_h = X + F(\mathsf{inl}).h$. The definition of L_{i_Y} shows that these are equal. The 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 $\mathrm{colim}1.\mathsf{inl} = \overline{F_X(i_x)} = \overline{i_X} = \eta$. 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 $\phi^\dagger : E \longrightarrow T^\nu(X)$ 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 a H-rational equation ϕ is a map $\phi^\dagger : E \longrightarrow T^\nu(X)$ for which

$$\begin{tikzcd} E \arrow[rr, "\phi"] \arrow[dr, "\phi^\dagger"'] & & X + FH(X+E) \arrow[dl, "\psi"] \\ & T^\nu(X) & \end{tikzcd}$$

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

$$FH(X+E) \xrightarrow{F!} FT^\nu(X+E) \xrightarrow{FT^\nu[\eta,\phi]} FT^{\nu 2}(X) \xrightarrow{F\mu} FT^\nu \longrightarrow T^\nu$$

Notice that each H-rational equation $\phi : E \xrightarrow{\phi} X + FH(X + E)$ determines a $X + F$-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 $E \longrightarrow T^{\nu}(X)$ and $X+F$-coalgebra morphisms $H(X+E) \longrightarrow T^{\nu}(X)$. In one direction, we simple restrict a coalgebra morphism with the canonical inclusion $E \longrightarrow H(X+E)$, while given a map $\psi : E \longrightarrow T^{\nu}(X)$ we can construct a map $H(X+E) \xrightarrow{H[\eta,\psi]} HT^{\nu}(X) \longrightarrow T^{\nu}(X)$. It's 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 $T^{\nu}(X)$ is the final $X+F$-coalgebra we have proved:

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

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, $\phi^{\dagger} = (\psi^*(\phi))^{\dagger}.\psi$

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 $T^{\nu}(X)$, 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)` which 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 infact 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 et al., C., editor, *Proceedings of CMCS'01*, volume 44(1) of *ENTCS*. Elsevier, Amsterdam.

Adamek, J., Milius, S., and Velebil, J. (2001). Free iterative theories: a coalgebraic view. *submitted.*

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

Corradini, A. and Gadducci, F. (1997). A 2-categorical presentation of term graph rewriting. In *Proceedings CTCS'97*, volume 1290 of *LNCS*. Springer.

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 for 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 et al., C., editor, *Proceedings of CMCS'01*, volume 44(1) of *ENTCS*. Elsevier, Amsterdam.

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*, number 1290 in LNCS, pages 69– 86. Springer.

Moss, L. Parametric corecursion. preprint, available at `http://math.indiana.edu/home/moss/parametric.ps`.