Theoretical Computer Science 412 (2011) 4969–4988

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

On second-order iterative monads

Jiří Adámek [a], Stefan Milius [a,*], Jiří Velebil [b,1]

[a] Institut für Theoretische Informatik, Technische Universität Braunschweig, Germany
[b] Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

ARTICLE INFO

Keywords:
Algebraic trees
Recursive program schemes
Ideal theory
Monads

ABSTRACT

B. Courcelle studied algebraic trees as precisely the solutions of all recursive program schemes for a given signature in Set. He proved that the corresponding monad is iterative. We generalize this to recursive program schemes over a given finitary endofunctor H of a "suitable" category. A monad is called second-order iterative if every guarded recursive program scheme has a unique solution in it. We construct two second-order iterative monads: one, called the second-order rational monad, S^H, is proved to be the initial second-order iterative monad. The other one, called the context-free monad, C^H, is a quotient of S^H and in the original case of a polynomial endofunctor H of Set we prove that C^H is the monad studied by B. Courcelle. The question whether these two monads are equal is left open.

1. Introduction

Recursive program schemes formalize the construction of new programs from the given ones by solving a recursive system of second-order equations. Building on the classical work of Bruno Courcelle [11] we introduce, for every finitary endofunctor H of a locally finitely presentable category, the *context-free monad* C^H of H. In the case where H is the polynomial endofunctor of a signature Σ in Set we prove that C^H is Courcelle's monad of *algebraic trees*, i.e., those Σ-trees that are solutions of recursive program schemes. This monad C^H is a quotient monad of the *second-order rational monad* S^H defined as the colimit of the diagram of all recursive program schemes. This is analogous to our previous construction of the rational monad R^H characterizing solutions of first-order recursive equations of type H; see [4]. In the case of a polynomial functor $H = H_\Sigma$ on Set the monad R^H is given by all rational Σ-trees, i.e., Σ-trees having (up to isomorphism) only a finite set of subtrees; see [19].

Recall from [11] the language $L(t)$ associated to every tree t: this is the language consisting of all words $n_1 \dots n_k f$ where $n_1 \dots n_k$ is a word over ω denoting a path from the root of t to a node (of depth k), and $f \in \Sigma$ is the label of that node. The tree t is rational iff $L(t)$ is a regular language, and, as proved in [13], the tree t is algebraic iff $L(t)$ is a deterministic context-free language. For this reason we call C^H the context-free monad for H.

This paper is part of our research program to provide a new and conceptionally clear approach to algebraic semantics (see e.g. [11,20]) which is a topic at the heart of theoretical computer science. In this new approach we use category theoretic methods and tools instead of general algebraic ones. So in lieu of sets, signatures and trees we consider categories, functors, final coalgebras and monads formed by them. Algebraic trees for a signature are a very important concept in classical

* Corresponding author. Tel.: +49 531 391 9524.
E-mail address: mail@stefan-milius.eu (S. Milius).

1 The third author was supported by the grant MSM 6840770014 of the Ministry of Education of the Czech Republic.

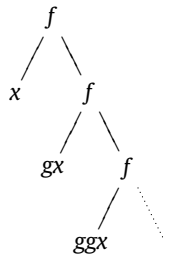
0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.04.027

algebraic semantics providing a semantic domain for uninterpreted solutions of recursive program schemes. So it is an important question how algebraic trees can be captured in our more general category theoretic approach to algebraic semantics, and we present the context-free monad as an answer. In addition, the move from signatures to endofunctors also allows to extend the notion of recursive program scheme. So in our new semantics we can capture recursive equations that the classical work cannot deal with; for example, equational laws between given operations of a program scheme may be considered directly in our approach—this is discussed in [24].

Let us now explain the results of this paper a bit more in detail. Recall that a *recursive program scheme* (or *rps* for short) defines new operations $\varphi_1, \ldots, \varphi_k$ of given arities $n_1, \ldots, n_k$ recursively, using given operations represented by symbols from a signature Σ. An rps is *guarded* if the right-hand sides of the equations have the leading symbol in Σ. Here is an example:

$$\varphi(x) = f(x, \varphi(gx)) \tag{1.1}$$

is a recursive program scheme defining a unary operation φ from the givens in $\Sigma = \{f, g\}$ with f binary and g unary. Here we are interested in the so-called uninterpreted semantics, which treats a recursive program scheme as a purely syntactic construct, and so its solution is given by Σ-trees over the given variables. For example, the uninterpreted solution of φ above is the Σ-tree



(1.2)

(here we simply put the terms x, gx, ggx, etc. for the corresponding subtrees).

Observe that if $\Phi = \{\varphi_1, \ldots, \varphi_k\}$ denotes the finite signature of the newly defined operations of arities n_i and

$$H_\Phi X = X^{n_1} + \cdots + X^{n_k}$$

is the corresponding polynomial endofunctor of Set, then algebras for H_Φ are just the classical general algebras for the signature Φ. We denote by F^H the free monad on H, thus F^{H_Φ} is the monad of finite Φ-trees. A recursive program scheme can be formalized as a natural transformation

$$e : H_\Phi \to F^{H_\Sigma + H_\Phi}.$$

In fact, $F^{H_\Sigma + H_\Phi}$ is the monad of all finite $(\Sigma + \Phi)$-trees. Since X^{n_i} is a functor representable by n_i, a natural transformation from X^{n_i} into $F^{H_\Sigma + H_\Phi}$ is, by the Yoneda Lemma, precisely an element of $F^{H_\Sigma + H_\Phi}(n_i)$, i.e., a finite $(\Sigma + \Phi)$-tree on n_i variables. Thus, to give a natural transformation e as above means precisely to give k equations, one for each operation symbol φ_i from Φ,

$$\varphi_i(x_0, \ldots, x_{n-1}) = t_i \quad (i = 1, \ldots, k) \tag{1.3}$$

where t_i is a $(\Sigma + \Phi)$-term on $\{x_0, \ldots, x_{n-1}\}$. This is the definition of a recursive program scheme used in [11].

An uninterpreted solution of $e : H_\Phi \to F^{H_\Sigma + H_\varphi}$ is a k-tuple of Σ-trees $t_1^\dagger, \ldots, t_k^\dagger$ such that the above formal equations (1.3) become identities under the simultaneous second-order substitution[2] of t_i for f_i, for $i = 1, \ldots, k$. For example, the tree $t^\dagger(x)$ from (1.2) satisfies the corresponding equality of trees

$$t^\dagger(x) = g(x, t^\dagger(fx)).$$

This concept of solutions was formalized in [24] by means of the free completely iterative monad T^H on a functor H; in the case $H = H_\Sigma$ this is the monad of all (finite and infinite) Σ-trees. We recall this in Section 2. The uninterpreted solution is a natural transformation $e^\dagger : H_\Phi \to T^{H_\Sigma}$ and this leads us to the following reformulation of the concept of an algebraic tree of Courcelle [11]:

A Σ-tree is called *algebraic* if there exists a recursive program scheme (1.3) such that $t = t_1^\dagger$. (Every rational tree is algebraic, and (1.2) shows an algebraic tree that is not rational.)

Courcelle proved that the monad C^{H_Σ} of all algebraic Σ-trees as a submonad of T^{H_Σ} is iterative in the sense of Calvin Elgot [12]. Furthermore, algebraic trees are closed under second-order substitution. In this paper we study, for general finitary endofunctors H, solutions of recursive program schemes in an arbitrary H-pointed monad, i.e., a monad B together with a natural transformation from H to B.

[2] Recall that, in general, a simultaneous second-order substitution replaces in a tree over a signature Γ all operation symbols by trees over another signature; see [11] for classical second-order substitution or [24] for a category theoretic description.

Definition 1.1. An H-pointed finitary monad is called *second-order iterative* provided that every guarded recursive program scheme has a unique solution in it.

The aim of this paper is a construction of two second-order iterative monads. The first one, denoted S^H, is given by colimits of all guarded recursive program schemes. This is analogous to the rational monad we introduced in [4], and therefore we call S^H the *second-order rational monad*. We prove that S^H is the initial second-order iterative monad. The other monad we introduce is the *context-free monad* C^H. We prove that this monad agrees with the monad of algebraic Σ-trees for the polynomial endofunctors $H = H_\Sigma$ of Set. We also prove that the monads S^H and C^H are always ideal in the sense of Elgot [12], i.e., they can be seen as a coproduct of variables and non-variables—this is a desired property that simplifies working with a monad; see e.g. [24,7,18]. However, at this moment we leave as open problems the proofs that C^H is closed under second-order substitution and it is iterative, in general. Our main open problem is whether $S^H = C^H$.

Related work. This paper is a completely revised and extended version of [6] containing full proofs. In addition, here the notion of second-order iterative monad is introduced. The result that S^H is the initial second-order iterative monad is new.

Our work is based on the pioneering paper by Bruno Courcelle [11]. As we mentioned already, Irène Guessarian [20] presents the classical algebraic semantics of recursive program schemes, for example, their uninterpreted solution as infinite Σ-trees and their interpreted semantics in ordered algebras. The realization that basic properties of Σ-trees stem from the fact that they form the final H_Σ-coalgebra goes back to Larry Moss [25] and also appears independently and almost at the same time in the work of Neil Ghani et al. [16] (see also [17]) and Peter Aczel et al. [2] (see also [1]). Ghani et al. [14] were the first to present a semantics of uninterpreted recursive program schemes in the coalgebraic setting. Their paper contains a solution theorem for uninterpreted (generalized) recursive program schemes. Here we derive from that the result that all "guarded" recursive program schemes have a unique solution, i.e., a unique fixed point w. r. t. second-order substitution. The ideas of [14] were taken further in [24]; this fundamental study contains a comprehensive category theoretic version of algebraic semantics in the coalgebraic setting: the paper provides an uninterpreted as well as interpreted semantics of recursive program schemes and the relation of the two semantics (this is a fundamental theorem in algebraic semantics).

Here we build on ideas in [14,24]. Our constructions of the second-order rational and the context-free monads are new. They are inspired by the constructions of the rational monad in [4,15].

2. Construction of the monads S^H and C^H

Throughout the paper we assume that a finitary (i.e., filtered colimit preserving) endofunctor H of a category $\mathscr{A}$ is given, and that H preserves monomorphisms. We assume that $\mathscr{A}$ is locally finitely presentable, coproduct injections

$$\mathsf{inl} : X \to X + Y \quad \text{and} \quad \mathsf{inr} : Y \to X + Y$$

are always monic, and a coproduct of two monomorphisms is also monic. Recall that local finite presentability means that $\mathscr{A}$ is cocomplete and has a set $\mathscr{A}_{\mathsf{fp}}$ of finitely presentable objects (meaning those whose hom-functors are finitary) such that $\mathscr{A}$ is the closure of $\mathscr{A}_{\mathsf{fp}}$ under filtered colimits.

Example 2.1.

1. Sets, posets and graphs form locally finitely presentable categories, and our assumptions about monomorphisms hold in these categories. Finite presentability of objects means precisely that they are finite.
2. If $\mathscr{A}$ is locally finitely presentable, then so is $\mathsf{Fun}_f(\mathscr{A})$, the category of all finitary endofunctors of $\mathscr{A}$ and natural transformations. In the case $\mathscr{A} = \mathsf{Set}$, the polynomial endofunctor
$$H_\Sigma X = \coprod_{\sigma \in \Sigma} X^n \qquad n = \text{arity of } \sigma \tag{2.1}$$
is a finitely presentable object of $\mathsf{Fun}_f(\mathsf{Set})$ iff Σ is a finite signature. This is easily seen using the Yoneda Lemma. In fact, the finitely presentable objects of $\mathsf{Fun}_f(\mathsf{Set})$ are precisely quotients $H_\Sigma/\sim$ of the polynomial functors with Σ finite, where $\sim$ is a congruence on H_Σ; see [5].

   Notice that our assumptions concerning monomorphisms carry over to $\mathsf{Fun}_f(\mathscr{A})$ since coproducts are formed object wise and natural transformations are monic iff their components are.

Remark 2.2. We shall need to work with categories that are locally finitely presentable but where the assumptions on monomorphisms above need not hold:

1. The category
$$\mathsf{Mnd}_f(\mathscr{A})$$
of all finitary monads on $\mathscr{A}$ and monad morphisms. This is a locally finitely presentable category. Indeed, the forgetful functor
$$\mathsf{Mnd}_f(\mathscr{A}) \to \mathsf{Fun}_f(\mathscr{A})$$
is finitary and monadic (see e.g. [21]), thus, the local finite presentability of $\mathsf{Fun}_f(\mathscr{A})$ implies that of $\mathsf{Mnd}_f(\mathscr{A})$; see [9], 2.78. It follows that $\mathsf{Mnd}_f(\mathcal{A})$ is cocomplete and filtered colimits of finitary monads are formed object wise on the level of $\mathscr{A}$.
2. The coslice category $A/\mathscr{A}$ of all morphisms with domain A is locally finitely presentable; see [9], Corollary 2.44.

Remark 2.3. Recall that a countably filtered colimit is a colimit whose scheme has for every countable subcategory a cocone. A functor is called *countably accessible* if it preserves countably filtered colimits. We denote by $\mathsf{Fun}_c(A)$ the category of all countably accessible endofunctors of A, and by $\mathsf{Mnd}_c(\mathcal{A})$ the category of all countably accessible monads. These two categories are cocomplete (in fact, locally $\aleph_1$-presentable).

Free Monad. Recall from [3] that since H is a finitary endofunctor, free H-algebras $\varphi_X : H(F^H X) \to F^H X$ exist for all objects X of $\mathcal{A}$. Denote by $\widehat{\eta}_X : X \to F^H X$ the universal arrow. As proved by Barr [10] the corresponding monad on $\mathcal{A}$ of free H-algebras, denoted by

$$F^H,$$

is a free monad on H. It follows that F^H is a finitary monad, and its unit

$$\widehat{\eta} : Id \to F^H$$

together with the natural transformation

$$\varphi : HF^H \to F^H$$

given by the above algebra structures φ_X yield the universal arrow

$$\widehat{\kappa} = (H \xrightarrow{H\widehat{\eta}} HF^H \xrightarrow{\varphi} F^H). \tag{2.2}$$

The universal property states that for every monad S and every natural transformation $f : H \to S$ there exists a unique monad morphism $\bar{f} : F^H \to S$ such that the triangle below commutes:

$$\begin{array}{ccc} H & \xrightarrow{\widehat{\kappa}} & F^H \\ & \searrow^{f} & \downarrow{\scriptstyle \bar{f}} \\ & & S \end{array} . \tag{2.3}$$

Moreover, from [3] we have

$$F^H = HF^H + Id \quad \text{with injections } \varphi \text{ and } \widehat{\eta}. \tag{2.4}$$

Remark 2.4. The category $\mathsf{Mnd}_f(\mathcal{A})$, being locally finitely presentable, has coproducts. We use the notation $\oplus$.

Given finitary endofunctor H and K, since the free monad on $H + K$ is the coproduct of the corresponding free monads, we have

$$F^{H+K} = F^H \oplus F^K. \tag{2.5}$$

We shall use the same notation φ, $\widehat{\eta}$ and $\widehat{\kappa}$ for other endofunctors than H, e.g. $\widehat{\kappa} : H + K \to F^{H+K}$.

Free Completely Iterative Monad. For every object X the functor $H(-) + X$, being finitary, has a terminal coalgebra

$$T^H X \to H\left(T^H X\right) + X. \tag{2.6}$$

By Lambek's Lemma [22], this morphism is invertible, and we denote the components of the inverse by

$$\tau_X : H\left(T^H X\right) \to T^H X \quad \text{and} \quad \eta_X : X \to T^H X.$$

respectively.

Notation 2.5. Since $T^H X$ is only used for the given functor H throughout the paper, we omit the upper index H, and write from now on simply

$$TX.$$

As proved in [1], T is the underlying functor of a monad (T, η, μ) with the unit $\eta : Id \to T$ above. This monad is, moreover, the free completely iterative monad on H; see [1,23]. The above natural transformation $\tau : HT \to T$ yields the universal arrow

$$\kappa = (H \xrightarrow{H\eta} HT \xrightarrow{\tau} T). \tag{2.7}$$

Moreover, in analogy to (2.4) above, we have

$$T = HT + Id \qquad \text{with injections } \tau \text{ and } \eta. \tag{2.8}$$

Also recall from loc. cit. that the monad multiplication $\mu : TT \to T$ is a homomorphism of H-algebras (we drop objects in the square below as all arrows are natural transformations):

$$\begin{array}{ccc} HTT & \xrightarrow{\tau T} & TT \\ {\scriptstyle H\mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ HT & \xrightarrow[\tau]{} & T \end{array} \tag{2.9}$$

Lemma 2.6. *The functor T is countably accessible, i.e., it preserves countably filtered colimits.*

Proof. In [4] we constructed the countably accessible monad $R^{\aleph_1}$ by forming, for every object Z of $\mathscr{A}$, the colimit $R^{\aleph_1}Z$ of the diagram of all coalgebras of $H(-)+Z$ carried by countably presentable objects of $\mathscr{A}$. In Proposition 5.16 of [4] we proved that $T = R^{\aleph_1}$. □

Notation 2.7. (i) We denote by

$$H/\mathsf{Mnd}_c(\mathcal{A})$$

the category of *H-pointed countably accessible monads*, i. e., pairs (B, β) where B is a countably accessible monad on $\mathscr{A}$ and $\beta : H \to B$ is a natural transformation. This is isomorphic to the coslice category of F^H:

$$H/\mathsf{Mnd}_c(\mathscr{A}) \cong F^H/\mathsf{Mnd}_c(\mathscr{A}).$$

Since $\mathsf{Mnd}_c(\mathscr{A})$ is cocomplete, so is $H/\mathsf{Mnd}_c(\mathscr{A})$.

(ii) For example, F^H and T are H-pointed monads (via the universal arrows). We shall often simply write the monad B when we refer to an object (B, β) of $\mathsf{Mnd}_c(\mathscr{A})$.

(iii) For every H-pointed monad (B, β) we write

$$b^+ = [\mu^B \cdot \beta B, \eta^B] : HB + Id \to B.$$

Lemma 2.8 (*Ghani et al. [15]*)**.** *For every H-pointed monad (B, β) the endofunctor $HB + Id$ carries a canonical monad structure whose unit is the coproduct injection* $\mathsf{inr} : Id \to HB + Id$ *and whose multiplication is given by*

$$\begin{array}{c} (HB+Id)(HB+Id) \\ \Vert \\ HB(HB+Id)+HB+Id \\ \downarrow{\scriptstyle HBb^+ + HB + Id} \\ HBB + HB + Id \\ \downarrow{\scriptstyle [H\mu^B, HB] + Id} \\ HB + Id \end{array} \tag{2.10}$$

Remark 2.9. For $HB + Id$ we also have an obvious H-pointing

$$\mathsf{inl} \cdot H\eta^B : H \to HB + Id. \tag{2.11}$$

This defines an endofunctor $\mathcal{H} : H/\mathsf{Mnd}_c(\mathcal{A}) \to H/\mathsf{Mnd}_c(\mathcal{A})$ on objects by

$$\mathcal{H}(B, \beta) = (HB + Id, \mathsf{inl} \cdot H\eta^B);$$

see [15] or [24], Lemma 5.2, for details.

Example 2.10. For every finitary endofunctor V we consider F^{H+V} as an H-pointed monad via

$$H \xrightarrow{\mathsf{inl}} H + V \xrightarrow{\widehat{\kappa}} F^{H+V}. \tag{2.12}$$

And $\mathcal{H}(F^{H+V}) = HF^{H+V} + Id$ is then an H-pointed monad via (2.11) which yields the pointing

$$\psi = (H \xrightarrow{H\widehat{\eta}} HF^{H+V} \xrightarrow{\mathsf{inl}} HF^{H+V} + Id). \tag{2.13}$$

Notation 2.11. Analogously to the category $H/\mathsf{Mnd}_c(\mathcal{A})$ in Notation 2.7 we write

$$H/\mathsf{Mnd}_f(\mathscr{A})$$

for the category of all H-pointed *finitary* monads. Observe that $H/\mathsf{Mnd}_f(A) \cong F^H/\mathsf{Mnd}_f(A)$ is locally finitely presentable; see Remark 2.2(ii). Thus, it is cocomplete.

Lemma 2.12. *Let V be a finitely presentable endofunctor as an object of* $\mathsf{Fun}_f(\mathscr{A})$*. Then F^{H+V} is a finitely presentable object of $H/\mathsf{Mnd}_f(\mathscr{A})$.*

Proof. The category $H/\mathsf{Mnd}_f(\mathscr{A})$ is locally finitely presentable and its object $B = F^H$ fulfills $F^{H+V} = B \oplus F^V$; see (2.5). The forgetful functor $U : B/(H/\mathsf{Mnd}_f(\mathscr{A})) \to H/\mathsf{Mnd}_f(\mathscr{A})$ has the left adjoint $X \mapsto B \oplus X$, and since U is finitary its left adjoint preserves finitely presentable objects. If V is finitely presentable in $\mathsf{Fun}_f(A)$, then F^V is finitely presentable in $H/\mathsf{Mnd}_f(\mathscr{A})$, thus, $F^{H+V} = B \oplus F^V$ is also finitely presentable. □

The proof of the following theorem is similar to the proof of Lemma 2.6 in [15]. The precise statement using the category $H/\mathsf{Mnd}(\mathscr{A})$ can be found in [24], Theorem 5.4.

Theorem 2.13. *The terminal coalgebra for $\mathcal{H}$ is given by the free completely iterative monad T, H-pointed as in (2.7), with the coalgebra structure $T \xrightarrow{\sim} \mathcal{H}T$ from (2.6).*

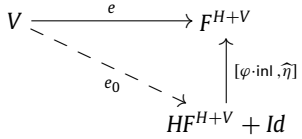
Definition 2.14. A *recursive program scheme* (or *rps* for short) of type H is a natural transformation

$$e : V \to F^{H+V}$$

from an endofunctor V which is a finitely presentable object of $\mathsf{Fun}_f(\mathscr{A})$ to the free monad on $H + V$. It is called *guarded* provided that it factorizes through the summand $HF^{H+V} + Id$ of the coproduct (2.4):

$$F^{H+V} = (H + V)F^{H+V} + Id = HF^{H+V} + VF^{H+V} + Id,$$

that is, we have a commutative triangle



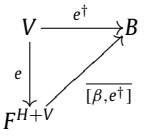
(2.14)

Observe that e_0 is unique since the vertical arrow, being a coproduct injection, is monic. This implies that e_0 and e are in bijective correspondence, which is the reason for our assumption that $\mathscr{A}$ have monic coproduct injections.

Example 2.15. In the case of a polynomial endofunctor $H = H_\Sigma : \mathsf{Set} \to \mathsf{Set}$ every recursive program scheme (1.3) yields a natural transformation $e : H_\Phi \to F^{H_\Phi + H_\Sigma}$, as explained in the introduction. This is a special case of Definition 2.14: in lieu of a general finitely presentable endofunctor V, which is a quotient of H_Φ (cf. Example 2.1(ii)), we just take $V = H_\Phi$.

The system (1.3) is guarded iff every right-hand side term is either just a variable or it has an operation symbol from Σ at the head of the term. Such a recursive program scheme is said to be in *Greibach normal form*. All reasonable rps, e.g. (1.1), are guarded. The unguarded ones such as $f(x) = f(x)$ are to be avoided if we want to work with unique solutions.

Definition 2.16. By a *solution* of a recursive program scheme $e : V \to F^{H+V}$ in an H-pointed monad (B, β) is meant a natural transformation $e^\dagger : V \to B$ such that the unique monad morphism $[\beta, e^\dagger]$ extending $H + V \to B$ (see (2.3)) makes the triangle below commutative:



(2.15)

Remark 2.17. Every guarded recursive program scheme (2.14) turns F^{H+V} into a coalgebra for $\mathcal{H}$. Indeed, $e_0 : V \to \mathcal{H}(F^{H+V})$ together with the pointing ψ (see (2.13)) yield a natural transformation $[\psi, e_0] : H + V \to \mathcal{H}(F^{H+V})$ which, by the universal property of the free monad F^{H+V}, provides a unique monad morphism

$$\overline{[\psi, e_0]} : F^{H+V} \to \mathcal{H}(F^{H+V}). \tag{2.16}$$

It preserves the pointing: we have

$$\overline{[\psi, e_0]} \cdot (\widehat{\kappa} \cdot \mathsf{inl}) = [\psi, e_0] \cdot \mathsf{inl} = \psi.$$

Thus, F^{H+V} is a coalgebra for $\mathcal{H}$.

(2) Conversely, every coalgebra for $\mathcal{H}$ carried by F^{H+V}, where V is a finitely presentable endofunctor, stems from a guarded recursive program scheme: the coalgebra structure $r : F^{H+V} \to \mathcal{H}(F^{H+V})$ is uniquely determined by $r \cdot \widehat{\kappa} : H + V \to \mathcal{H}(F^{H+V})$, and since the left-hand component of $r \cdot \widehat{\kappa}$ is the pointing ψ, we see that r is determined by $e_0 = r \cdot \widehat{\kappa} \cdot \mathsf{inr} : V \to \mathcal{H}(F^{H+V})$ defining a (unique) recursive program scheme.

Notation 2.18. For every guarded recursive program scheme $e : V \to F^{H+V}$ we denote by

$$e^* : F^{H+V} \to T \tag{2.17}$$

the unique coalgebra homomorphism for $\mathcal{H}$ (included by $\overline{[\psi, e_0]}$ above), and by

$$\widehat{e} : F^{H+V} \to F^{H+V}$$

the unique monad morphism extending $[\widehat{\kappa} \cdot \mathsf{inl}, e] : H + V \to F^{H+V}$; i.e., we have

$$\widehat{e} \cdot \widehat{\kappa} = [\widehat{\kappa} \cdot \mathsf{inl}, e]. \tag{2.18}$$

Remark 2.19. Our concept of a recursive program scheme is a special case of the algebraic systems studied by Neil Ghani et al. [14]. Let us recall from that paper that

1. an H-pointed monad is called *coalgebraic* if it is isomorphic to the monad $HB + Id$ of Lemma 2.8 via $b^+ : HB + Id \to B$ in Notation 2.7(ii),
2. examples of coalgebraic monads include F^H (see (2.4)) and T (see (2.8))
3. T is the final coalgebraic monad; we denote by $u_B : B \to T$ the unique morphism for a coalgebraic monad (B, β),
4. an *algebraic system* is given by a finitary monad E, a finitary coalgebraic monad (B, β) and a monad morphism
$$e : E \to H(B \oplus E) + Id,$$
5. a *solution* of e is a monad morphism $e^\ddagger : E \to T$ such that the square below commutes:

$$\begin{array}{ccc} E & \xrightarrow{\quad e^\ddagger \quad} & T \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle [\tau,\eta]^{-1}} \\ H(B \oplus E) + Id & \xrightarrow[H([u_B, e^\ddagger]) + Id]{} & HT + Id \end{array}$$

Theorem 2.20 (*Ghani et al. [14]*)**.** *Every algebraic system has a unique solution.*

This gives a solution theorem for recursive program schemes as follows: due to (2.5) we have the morphism $e_0 : V \to H(F^H \oplus F^V) + Id$ in (2.14) yielding an algebraic system via (2.3):

$$\overline{e_0} : F^V \to H(F^H \oplus F^V) + Id. \tag{2.19}$$

Indeed, take $E = F^V$ and $B = F^H$. Thus, a unique solution $e^\ddagger : F^V \to T$ exists.

Theorem 2.21. *Every guarded recursive program scheme of type H has a unique solution $e^\dagger$ in T. It can be computed from the unique coalgebra homomorphism $e^* : F^{H+V} \to T$ (see (2.17)) by*

$$e^\dagger = (V \xrightarrow{\text{inr}} H + V \xrightarrow{\widehat{\kappa}} F^{H+V} \xrightarrow{e^*} T). \tag{2.20}$$

Indeed, for the unique solution $e^\ddagger : F^V \to T$ of the algebraic system $\overline{e_0}$ in (2.19) above we obtain a solution $e^\dagger$ in the sense of Definition 2.14 by composing with $\widehat{\kappa} : V \to F^V$:

$$e^\dagger = (\ V \xrightarrow{\widehat{\kappa}} F^V \xrightarrow{e^\ddagger} T\ ).$$

The proof that (2.15) commutes is performed using some diagram chasing. A somewhat subtle point is that for $u_B : B \to T$ (see Remark 2.19(iii)) we have the equality

$$[u_B, e^\ddagger] = \overline{[\widehat{\kappa}, e^\dagger]} : F^{H+V} \to T.$$

Here the square brackets on the left refer to the coproduct of F^H and F^V in $H/\mathsf{Mnd}_f(\mathscr{A})$ and those on the right to $H + V$ in $\mathsf{Fun}_f(\mathscr{A})$. The verification uses the universal property of the free monad on $H + V$ and is not difficult. The fact that (2.20) holds follows from the same diagram.

To prove that $e^\dagger$ is unique use the fact that for any solution $e^\dagger$ in the sense of Definition 2.14 its extension $\overline{e^\dagger} : F^V \to T$ is a solution of the corresponding algebraic system $\overline{e_0}$.

Remark 2.22. It is our goal to define a submonad C^H of T formed by all solutions of recursive program schemes of type H. We do this in two steps.

1. A finitary monad S^H together with a monad morphism $s^* : S^H \to T$ is constructed by forming a colimit of coalgebras for the endofunctor $\mathcal{H}$ obtained from all recursive program schemes. We prove later that S^H is the initial second-order iterative monad for H.
2. The (strong epi, mono)-factorization of s^* is formed to obtain the desired submonad C^H:

$$\begin{array}{ccc} S^H & \xrightarrow{\quad s^* \quad} & T \\ & {\scriptstyle k}\searrow \quad \nearrow{\scriptstyle c} & \\ & C^H & \end{array}$$

Proposition 2.23. *The category $\mathsf{Mnd}_c(\mathscr{A})$ has as monomorphisms precisely the monad morphisms with monic components. It has (strong epi, mono)-factorizations and $\mathsf{Mnd}_f(\mathscr{A})$ is closed in $\mathsf{Mnd}_c(\mathscr{A})$ under subobjects and strong quotients.*

Proof. (1) The category $\mathsf{Fun}_c(\mathscr{A})$ of all countably accessible endofunctors has as monomorphisms precisely the morphisms with monic components. The first statement of our proposition follows from the fact that every monomorphism $m : P \to Q$ in $\mathsf{Mnd}_c(\mathscr{A})$ is monomorphic in $\mathsf{Fun}_c(\mathscr{A})$. Indeed, consider $u, v : K \to P$ with $m \cdot u = m \cdot v$ where K is a countably accessible endofunctor. Then free K-algebras exist; see [3]. Therefore a free monad F^K exists; see [10]. The corresponding monad morphisms $\overline{u}, \overline{v} : F^K \to P$ (cf. (2.3)) fulfill $m \cdot \overline{u} = m \cdot \overline{v}$. This implies $\overline{u} = \overline{v}$ since m is monic as a monad morphism. Thus, $u = \overline{u} \cdot \widehat{\kappa} = \overline{v} \cdot \widehat{\kappa} = v$, as desired.

(2) The existence of (strong epi, mono)-factorizations follows from the local presentability of the category $\mathsf{Mnd}_c(\mathscr{A})$; see [9], Proposition 1.16.

(3) The category $\mathsf{Mnd}_f(\mathscr{A})$ of all finitary monads is closed under subobjects in $\mathsf{Mnd}_c(\mathscr{A})$ since (by the same argument above) monomorphisms in $\mathsf{Mnd}_f(\mathscr{A})$ are precisely the morphisms that are component wise monic. And $\mathsf{Mnd}_f(\mathscr{A})$ is closed under strong quotients in $\mathsf{Mnd}_c(\mathscr{A})$ since this subcategory is coreflective; indeed, all left adjoints preserve strong epimorphisms; see [8]. □

Corollary 2.24. *The functor $\mathcal{H}$ preserves monomorphisms.*

Indeed, given a monomorphism $m : (B, \beta) \to (B', \beta')$ in $H/\mathsf{Mnd}(\mathscr{A})$, then m is component wise monic, thus, so is Hm (since H preserves monomorphisms), and so is also $\mathcal{H}m = Hm + id$ (since coproducts of monomorphisms are monic in $\mathscr{A}$).

Remark 2.25. The endofunctor $\mathcal{H}$ takes finitary monads to finitary monads: if B preserves filtered colimits, so does $H \cdot B + Id$. Thus, it restricts to an endofunctor $\mathcal{H}_f$ of $H/\mathsf{Mnd}_f(\mathscr{A})$. The latter category is cocomplete (see Notation 2.11) hence, the category of all coalgebras of $\mathcal{H}_f$ is also cocomplete.

Construction 2.26. The H-pointed monad S^H, called the *second-order rational monad* of H. For every guarded recursive program scheme (2.14) consider F^{H+V} as a coalgebra for the functor $\mathcal{H}$; see (2.16).

We denote by

$$\mathsf{EQ} \subseteq \mathsf{Coalg}\,\mathcal{H}$$

the full subcategory of all these coalgebras. The inclusion functor is an essentially small diagram since $\mathsf{Fun}_f(\mathscr{A})$ has only a set of finitely presentable objects up to natural isomorphism. We denote the colimit of this small diagram by

$$S^H = \operatorname{colim} \mathsf{EQ} \quad (\text{in } \mathsf{Coalg}\,\mathcal{H}).$$

(This colimit exists in Notation 2.7(i).)

Thus, we have a finitary monad S^H with an H-pointing and a coalgebra structure denoted by

$$\sigma : H \to S^H \quad \text{and} \quad s : S^H \to \mathcal{H}(S^H) \tag{2.21}$$

respectively, forming a coalgebraic structure for $\mathcal{H}$. And we also have a colimit cocone

$$e^\sharp : F^{H+V} \to S^H \quad \text{for all rps } e : V \to F^{H+V}, \tag{2.22}$$

formed by coalgebra homomorphisms for $\mathcal{H}$ preserving the pointing (2.13), i.e. with

$$\sigma = e^\sharp \cdot (\widehat{\kappa} \cdot \mathsf{inl}) \quad \text{for every } e. \tag{2.23}$$

Lemma 2.27. EQ *is closed under finite coproducts in* $\mathsf{Coalg}\,\mathcal{H}$.

Proof. Consider two objects e and e' of EQ determined by

$$e_0 : V \to HF^{H+V} + Id \quad \text{and} \quad e'_0 : V' \to HF^{H+V'} + Id.$$

The coproduct injections $i : H + V \to H + V + V'$ and $i' : H + V' \to H + V + V'$ yield corresponding monad morphisms $\overline{i} : F^{H+V'} \to F^{H+V+V'}$ and $\overline{i'} : F^{H+V'} \to F^{H+V}$; see (2.3). Denote by

$$k = (HF^{H+V} + Id) + (HF^{H+V'} + Id) \xrightarrow{[H\overline{i}+Id, H\overline{i'}+Id]} HF^{H+V+V'} + Id$$

the canonical morphism. We prove that the object $f : V + V' \to F^{H+V+V'}$ of EQ determined by

$$f_0 = k \cdot (e_0 + e'_0) : V + V' \to HF^{H+V+V'} + Id \tag{2.24}$$

is the coproduct of the two given objects.

Consider a coalgebra X for $\mathcal{H}$ given by a pointed monad $\beta : H \to B$ and a coalgebra structure $b : B \to \mathcal{H}B$. We know from Remark 2.17 that a morphism from the above object (2.24) into this coalgebra is given by a natural transformation

$$t = [q, q'] : V + V' \to B$$

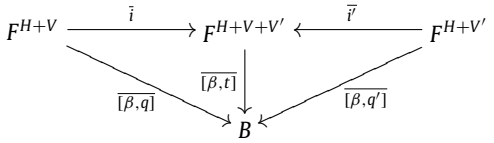
such that the extension $\overline{[\beta, t]} : F^{H+V+V'} \to B$ to a monad morphism (see (2.3)) fulfills

$$b \cdot \overline{[\beta, t]} = (H\overline{[\beta, t]} + Id) \cdot \overline{[\psi, f_0]}.$$

We claim that this holds for $t : V + V' \to B$ iff

(i) the left-hand component $q : V \to B$ of t yields a morphism $\overline{[\beta, q]} : F^{H+V} \to B$ of $\mathsf{Coalg}\,\mathcal{H}$ from the object determined by e_0 into X and
(ii) the right-hand component $q' : V' \to B$ yields a morphism $\overline{[\beta, q']} : F^{H+V} \to B$ from the object determined by e'_0 into X.

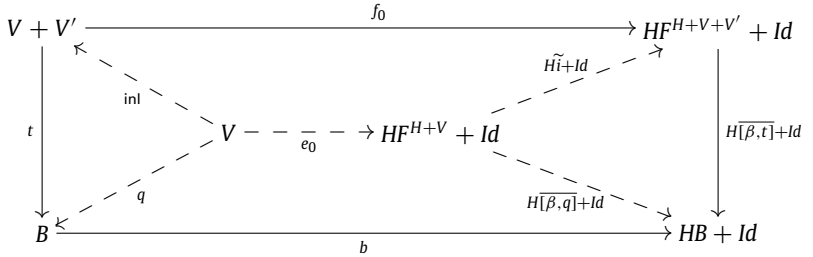
For that observe first that the diagram



commutes: indeed, all these morphisms are monad morphisms. The left-hand triangle commutes since $\bar{i} \cdot \widehat{\kappa}^{H+V} = \widehat{\kappa}^{H+V+V'} \cdot i$, therefore,

$$(\overline{[\beta, t]} \cdot \bar{i}) \cdot \widehat{\kappa} = [\beta, t] \cdot i = [\beta, q] = \overline{[\beta, q]} \cdot \widehat{\kappa}$$

and analogously for the right-hand triangle. Thus, the square



commutes iff $\overline{[\beta, q]}$ and $\overline{[\beta, q']}$ are morphisms of $\mathsf{Coalg}\,\mathcal{H}$ into X: in the diagram we indicated the left-hand component (commuting iff $b \cdot q = (H\overline{[\beta, q]} + Id) \cdot e_0$, that is, $\overline{[\beta, q]}$ is a homomorphism), analogously for the right-hand one. This proves that f is a coproduct of e and e' in EQ. □

Corollary 2.28. *S^H is a filtered colimit of the closure EQ_1 of EQ under coequalizers in $\mathsf{Coalg}\,\mathcal{H}$.*

Indeed, since EQ is closed under finite coproducts, EQ_1 is closed under finite colimits, thus, it is filtered. Observe that since the forgetful functors

$$\mathsf{EQ} \to \mathsf{Coalg}\,\mathcal{H} \to H/\mathsf{Mnd}_c(\mathscr{A}) \to \mathsf{Mnd}_c(\mathscr{A}) \to \mathsf{Fun}_c(\mathscr{A})$$

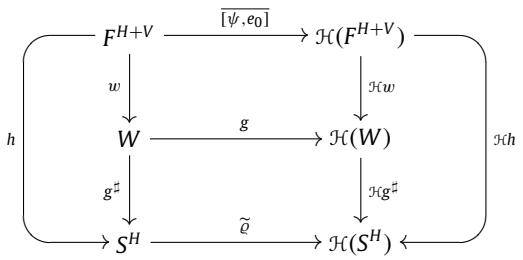
clearly preserve connected colimits, the above cocone $e^\sharp : F^{H+V} \to T$ is also a colimit cocone in $\mathsf{Fun}_c(\mathscr{A})$. In other words, the colimit

$$S^H = \operatorname{colim} \mathsf{EQ}_1$$

is performed object wise on the level of $\mathscr{A}$.

Lemma 2.29. *For every guarded recursive program scheme $e : V \to F^{H+V}$ the colimit injection $e^\sharp : F^{H+V} \to S^H$ is the unique coalgebra homomorphism from the $\mathcal{H}$-coalgebra F^{H+V} of Remark 2.16 into S^H.*

Proof. We have already seen in Construction 2.26 that $e^\sharp$ is an $\mathcal{H}$-coalgebra homomorphism. We now prove its uniqueness. Suppose that $h : F^{H+V} \to S^H$ is an $\mathcal{H}$-coalgebra homomorphism. Since F^{H+V} is a finitely presentable object by Lemma 2.12 and S^H a filtered colimit of EQ_1, there exists some object $g : W \to \mathcal{H}(W)$ in EQ_1 and a morphism $w : F^{H+V} \to W$ of $H/\mathsf{Mnd}_f(\mathscr{A})$ such that $g^\sharp \cdot w = h$. We may assume, without loss of generality, that w is an $\mathcal{H}$-coalgebra homomorphism:



Indeed, since the outside square commutes, we see that the upper square does when extended by $\mathcal{H}g^\sharp$. Because $\mathcal{H}$ is finitary, $\mathcal{H}g^\sharp$ is a colimit injection for $\mathcal{H}(S^H) = \operatorname{colim} \mathcal{H}\cdot \mathsf{EQ}_1$ merging the two parallel morphisms $F^{H+V} \to \mathcal{H}(W)$ given by the upper square. By the finite presentability of F^{H+V} there exists an object $g' : W' \to \mathcal{H}(W)$ in EQ_1 with a connecting morphism $\alpha : W \to W'$ such that $\mathcal{H}(\alpha)$ also merges the upper square. Now replace W (and g) by W' (and g') and replace also w by $\alpha \cdot w$ to obtain the desired commutative upper square.

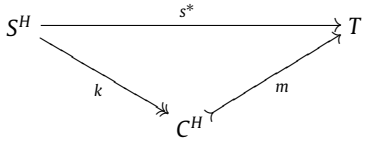
We now conclude that w is a connecting morphism from F^{H+V} to W in EQ_1, thus

$$h = g^\sharp \cdot w = e^\sharp. \quad \square$$

Definition 2.30. For the coalgebra S^H of (2.21) denote by

$$s^* : S^H \to T$$

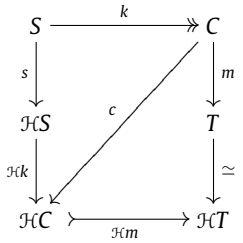
the unique coalgebra homomorphism (see Theorem 2.13). Define the *context-free monad of H* as the submonad C^H of T obtained by the following (strong epi, mono)-factorization of s^* in $\mathsf{Mnd}_c(\mathscr{A})$:



Remark 2.31.

1. Since S^H is finitary and T countably accessible (see Lemma 2.6) we have the desired factorization by Proposition 2.23.
2. The context-free monad is pointed: The pointing $\sigma : H \to S^H$ of S^H yields the pointing
$$\gamma = k \cdot \sigma : H \to C^H$$
of C^H. Observe that the morphism m preserves this pointing (because s^* is a morphism of $H/\mathsf{Mnd}_c(\mathscr{A})$):
$$\kappa = m \cdot \gamma : H \to T.$$
3. Analogously to T we shall write C and S without the upper index H from now on.

Observation 2.32. The functor $\mathcal{H}$ preserves monomorphisms by Corollary 2.24, thus, C carries a canonical structure c of an $\mathcal{H}$-coalgebra derived from the structure s for S:



(2.25)

Indeed, recall that $m \cdot k = s^*$ is an $\mathcal{H}$-coalgebra homomorphism; so the outside of the above square commutes, and we can use the unique diagonalization property of the factorization system to obtain c.

Remark 2.33. For the diagram EQ of all recursive program schemes we observed that the second-order rational monad S is the filtered colimit of

$$\mathsf{EQ}_1 = \text{ closure of } \mathsf{EQ} \text{ under coequalizers in } \mathsf{Coalg}\,\mathcal{H}.$$

We now observe that, analogously, the context-free monad C is the filtered colimit of

$$\mathsf{EQ}_2 = \text{ closure of } \mathsf{EQ} \text{ under strong quotients in } \mathsf{Coalg}\,\mathcal{H}.$$

In fact, for every recursive program scheme e factorize e^* of (2.17) as

$$F^{H+V} \xrightarrow{q_e} Q_e \xrightarrow{m_e} T$$

where q_e is a strong epimorphism and m_e a monomorphism. Then these objects Q_e of EQ_2, where e ranges through all recursive program schemes, form a cofinal subcategory of EQ_2, thus, the colimit $C' = \operatorname{colim} \mathsf{EQ}_2$ is also a colimit of the subdiagram formed by all Q_e. The cocone of monomorphisms $m_e : Q_e \to T$ then yields a monomorphism $m : C' \to T$ as the factorizing map; see [9], Proposition 1.62. The natural transformation with components $q_e : F^{H+V} \to Q_e$ yields a strong epimorphism $q = \operatorname{colim} q_e : S \to C'$. Since T is a terminal $\mathcal{H}$-coalgebra by Theorem 2.13, we conclude $s^* = m \cdot q$. Thus, $C' = C$, since C was obtained by a factorization of s^*.

3. Ideal monads

Under the assumptions of Section 2 we prove that the monads S and C are ideal in the sense of Elgot [12] for every finitary endofunctor H. Elgot's concept was defined for monads (S, η, μ) in Set: the monad is ideal if the complement of $\eta : Id \to S$ is a subfunctor $\sigma : S' \hookrightarrow S$ of S (thus, $S = S' + Id$) and μ restricts to a natural transformation $\mu' : S'S \to S'$. For general categories "ideal" is not a property but a structure:

Definition 3.1 (*[1]*)**.** An *ideal monad* is a six-tuple $(B, \eta, \mu, B', i, \mu')$ where (B, η, μ) is a monad,

$$i : B' \to B \quad \text{(called the ideal)}$$

is a subfunctor such that $B = B' + Id$ with injection i and η, and

$$\mu' : B'B \to B'$$

is a natural transformation restricting μ in the sense that

$$\mu \cdot iB = i \cdot \mu'. \tag{3.1}$$

Example 3.2.

1. The free monad F^H is ideal: its ideal is $\varphi^H : HF^H \to F^H$; see (2.4).
2. The free completely iterative monad T is ideal: its ideal is $\tau : HT^H \to T^H$; see (2.8).
3. Coproducts of ideal monads are ideal and have a nice construction; see [18].

Remark 3.3. It is our goal to prove that the context-free monad (C, η^C, μ^C) is ideal. The $\mathcal{H}$-coalgebra structure $\gamma : C \to HC + Id$, see Observation 2.32, is (analogously to the two examples F^H and T above) invertible, as we prove below: its inverse is the morphism

$$b^+ \equiv HC + Id \xrightarrow{\gamma C + Id} CC + Id \xrightarrow{[\mu^C, \eta^C]} C, \tag{3.2}$$

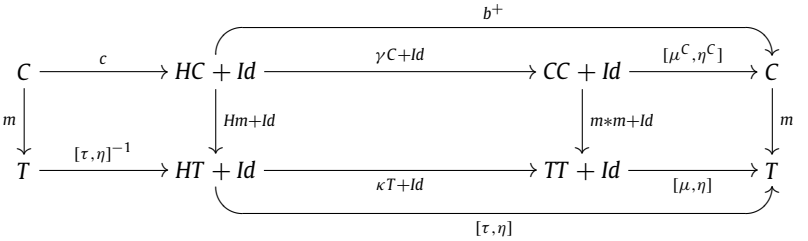
cf. Notation 2.7(ii). From that we will derive that C is an ideal monad with the ideal

$$b^+ \cdot \mathsf{inl} : HC \to C.$$

Theorem 3.4. *The context-free monad C is an ideal monad for every finitary functor H.*

Proof. We first prove that c is inverse to b^+.

(1) The proof of $b^+ \cdot c = id$ follows, since m is a monomorphism, from the commutativity of the following diagram (here $m * m$ denotes the parallel composition of natural transformations):



Indeed, the right-hand square commutes since $m : C \to T$ is a monad morphism, the left-hand one does because m is a coalgebra homomorphism for $\mathcal{H}$ (see (2.25)), and the middle square follows from the fact that, by Remark 2.31, m preserves the pointing, i.e., $m \cdot \gamma = \tau \cdot H\eta$. Finally, the lower part follows from (2.9):

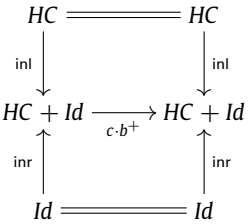
$$\mu \cdot \tau T \cdot H\eta T = \tau \cdot H\mu \cdot H\eta T = \tau.$$

So the outside of the diagram commutes:

$$m \cdot b^+ \cdot c = m$$

and since m is a monomorphism, we see that $b^+ \cdot c = id$.

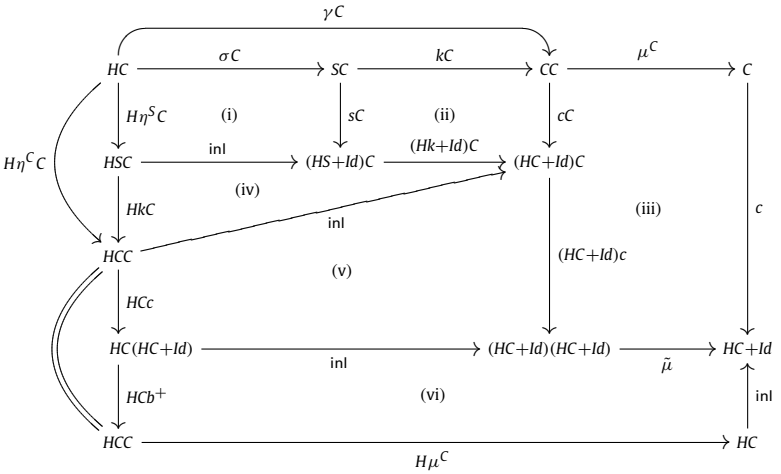
(2) To prove that $c \cdot b^+ = id$ we show that the diagram below commutes:



For the commutativity of the lower square we have, since c is a monad morphism and the unit of $\mathcal{H}C$ is, by Lemma 2.8, inr, that

$$c \cdot b^+ \cdot \mathsf{inr} = c \cdot \eta^C = \mathsf{inr}.$$

Since in (3.2) $b^+ \cdot \mathsf{inl} = \mu^C \cdot \gamma C = \mu^C \cdot (kC \cdot \sigma C)$, the commutativity of the upper square boils down to showing that the outside of the following diagram commutes:



Here $\widetilde{\mu}$ denotes the monad multiplication (2.10) of Lemma 2.8, where $B = C$ and $\beta = \gamma$. Indeed, all inner parts commute: the two left-hand parts commute since $k \cdot \eta^B = \eta^C$ and $b^+ \cdot c = id$, for part (i) recall that the coalgebra structure s is a morphism in $H/\mathsf{Mnd}(\mathscr{A})$, part (ii) commutes since k is a coalgebra homomorphism for $\mathcal{H}$ (cf. (2.25)), for (iii) use that c is a monad morphism, (iv) and (v) are trivial, and part (vi) commutes by (2.10). The remaining upper part commutes since k preserves the H-pointing. Finally, using the monad law $\mu^C \cdot \eta^C C = id$, we get $c \cdot \mu^C \cdot \gamma C = \mathsf{inl} : HC \to HC + Id$, and this completes the proof.

(3) The monad C is ideal with respect to the ideal

$$i^C = (HC \xrightarrow{\gamma C} CC \xrightarrow{\mu^C} C)$$

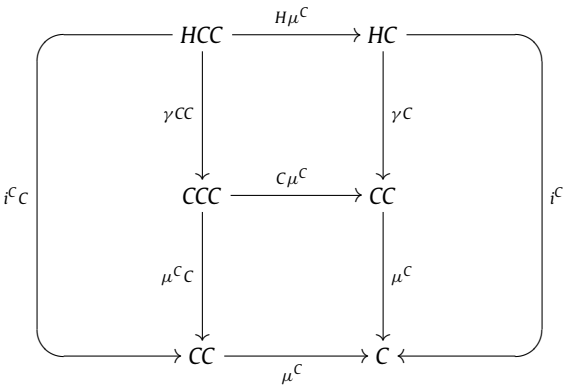
and the restriction of μ^C to

$$(\mu^C)' = H\mu^C : (HC)C \to HC.$$

Indeed, by the definition of b^+ (see (3.2)) we have

$$b^+ \cdot \mathsf{inl} = i^C \quad \text{and} \quad b^+ \cdot \mathsf{inr} = \eta^C$$

so that C is in fact a coproduct of HC and Id with injections i^C and η^C. And (3.1) is obvious from the following diagram



whose lower part is a monad axiom and the upper one is the naturality of γ. □

Remark 3.5. Next we prove that the second-order rational monad (S, η^S, μ^S) is ideal. Analogously to the preceding case we prove that the coalgebra structure $s : S \to HS + Id$ is invertible. For that we are going to use the following natural transformation

$$\xi : \mathcal{H} \to Id$$

introduced in [24]: given an object $\beta : H \to B$ of $H/\mathsf{Mnd}_c(A)$ with the monad structure $B = (B, \eta^B, \mu^B)$ define an algebra structure for $\mathcal{H}$

$$\xi_B : \mathcal{H}B \to B$$

as the following composite

$$\xi_B \equiv HB + Id \xrightarrow{\beta B + Id} BB + Id \xrightarrow{[\mu^B, \eta^B]} B. \tag{3.3}$$

As proved in [24], Lemma 5.2, this makes $\mathcal{H}$ a well-copointed endofunctor of $H/\mathsf{Mnd}_c(\mathscr{A})$, i.e.,

$$H\xi_B = \xi_{HB} \quad \text{for all } (B, \beta) \text{ in } H/\mathsf{Mnd}_c(\mathcal{A}). \tag{3.4}$$

Moreover, for every guarded recursive program scheme e the morphism $\widehat{e}$ of (2.18) makes the triangle

$$\begin{array}{ccc} F^{H+V} & \xrightarrow{\overline{[\psi, e_0]}} & \mathcal{H}F^{H+V} \\ {\scriptstyle \widehat{e}}\downarrow & \swarrow {\scriptstyle \xi_{F^{H+V}}} & \\ F^{H+V} & & \end{array} \tag{3.5}$$

commutative. This is proved analogously to Lemma 6.12 in [24]. In the next proof we verify that, analogously to $c^{-1} = b^+$ above, we have

$$s^{-1} = \xi_S. \tag{3.6}$$

Theorem 3.6. *The second-order rational monad S is ideal for every finitary functor H.*

Proof. We first prove (3.6).

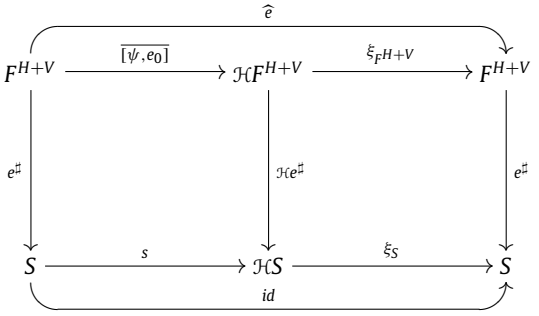
(1) The equation

$$\xi_S \cdot s = id \tag{3.7}$$

follows from the fact that the morphisms $e^\sharp$ of (2.22) fulfill

$$\xi_S \cdot s \cdot e^\sharp = e^\sharp \quad \text{for every } e \in \mathsf{EQ}_0. \tag{3.8}$$

This follows from the following diagram



It is our task to show that the left-hand vertical arrow merges the two lower endomorphisms of S. For that we observe that both the inner squares commute: recall that $e^\sharp$ is a coalgebra homomorphism and ξ is natural. The upper part commutes by (3.5) and the outside square does:

$$e^\sharp \cdot \widehat{e} = e^\sharp : F^{H+V} \to F^{H+V} \tag{3.9}$$

because $\widehat{e}$ is a coalgebra homomorphism for $\mathcal{H}$ (see Notation 2.18) thus so is $e^\sharp \cdot \widehat{e}$, and we apply Lemma 2.29. This proves (3.8).

(2) Next we prove

$$s \cdot \xi_S = id.$$

This is analogous to Part (2) of the proof in Theorem 3.4: replace b^+ with ξ_S, and also replace C (and c) with S (and s). We thus obtain

$$s \cdot \xi_S \cdot \mathsf{inr} = \mathsf{inr},$$

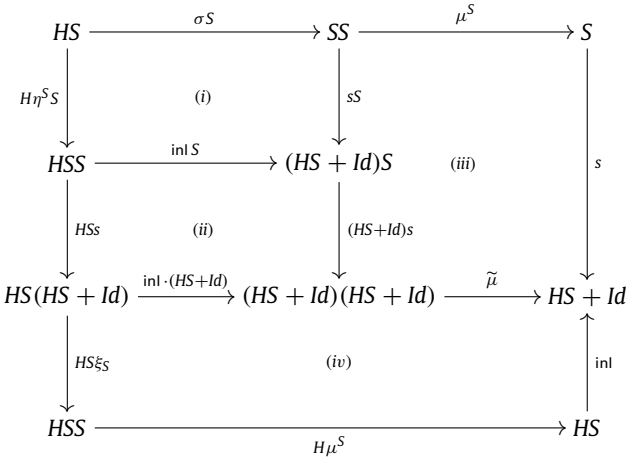
and it remains to prove

$$s \cdot \xi_S \cdot \mathsf{inl} = \mathsf{inl}.$$

From the definition (3.3) we see that $\xi_S \cdot \mathsf{inl} = \mu^S \cdot \sigma S$, thus, our task is to prove

$$s \cdot \mu^S \cdot \sigma S \cdot \mathsf{inl} = \mathsf{inl}. \tag{3.10}$$

We use the following diagram analogous to the diagram in the proof of Theorem 3.4:



(3.11)

Here $\widetilde{\mu}$ denotes the monad multiplication of Lemma 2.8, where $B = S$ and $\beta = \sigma$; thus (iv) commutes. Indeed, all inner parts commute: for part (i) recall that s is a morphism of $H/\mathsf{Mnd}_c(\mathcal{A})$, part (ii) triviality commutes, and part (iii) does since s is a monad morphism. Due to (3.7) the left-hand vertical arrows yield $H\eta^S S$. From this diagram we obtain, since $H\mu^S \cdot H\eta^S S = id$, the equality providing (3.10):

$$s \cdot \mu^S \cdot \sigma S = id.$$

(3) The proof that (S, μ^S, η^S) is an ideal monad with the ideal

$$i^S = (HS \xrightarrow{\sigma S} SS \xrightarrow{\mu^S} S)$$

and the restriction of μ^S to

$$(\mu^S)' = H\mu^S$$

is completely analogous to Theorem 3.4: replace γ with σ. □

4. Second-order iterative monads

In this section we introduce the concept of a second-order iterative monad as a monad where guarded recursive program schemes are uniquely solvable. And we prove that our monad S^H is the initial second-order iterative monad.

Definition 4.1. Given an endofunctor H of $\mathcal{A}$, we call an H-pointed monad (B, β) *second-order iterative* w.r.t. H provided that every guarded recursive program scheme $e : V \to F^{H+V}$ has a unique solution $e^\dagger : V \to B$; see (2.15).

Example 4.2. The free completely iterative monad T is second-order iterative by Theorem 2.21.

Theorem 4.3. *The context-free monad C is second-order iterative for every finitary functor H.*

Proof. Let $e : V \to F^{H+V}$ be a guarded recursive program scheme. We use $e^\ddagger$ for solutions in C and $e^\dagger$ for solutions in T throughout this proof. We are to prove that there exists a unique natural transformation $e^\ddagger : V \to C$ with $e^\ddagger = \overline{[\gamma, e^\ddagger]} \cdot e$.

Recall that the colimit injection $e^\sharp : F^{H+V} \to S$ in Construction 2.26 is a coalgebra homomorphism for $\mathcal{H}$, hence, so is $\widetilde{c} \cdot e^\sharp$. This proves

$$e^* = s^* \cdot e^\sharp,$$

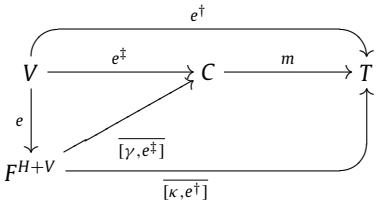
see Theorem 2.21 (because T is a terminal coalgebra by Theorem 2.13). Therefore, by (2.20) we have

$$e^\dagger = s^* \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr} = m \cdot k \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr}\,.$$

Thus for $e^\ddagger = k \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr}$ we obtain

$$e^\dagger = c \cdot e^\ddagger.$$

We conclude that $e^\ddagger$ is the desired solution in C: in the following diagram



the outside commutes, see (2.15) with $\sigma = \kappa$, and the right-hand part does since $\kappa = m \cdot \gamma$ (see Remark 2.31). Consequently, the left-hand triangle commutes: recall from Definition 2.30 that m is a monomorphism.

The uniqueness follows from the same diagram: if the left-hand triangle commutes, so does the outside one, and since $e^\dagger$ is uniquely determined (see Theorem 2.21), we conclude $e^\dagger = m \cdot e^\ddagger$. Finally, use again that m is monic. □

Theorem 4.4. *The second-order rational monad S is second-order iterative for every finitary functor H.*

Proof. For every recursive program scheme $e : V \to F^{H+V}$ we prove that a unique solution $e^\dagger : V \to S$ exists.

(1) Existence. For the colimit cocone (2.22) put

$$e^\dagger = (V \xrightarrow{\mathsf{inl}} H + V \xrightarrow{\widehat{\kappa}} F^{H+V} \xrightarrow{e^\sharp} S). \tag{4.1}$$

We are to prove that

$$e^\dagger = \overline{[\sigma, e^\dagger]} \cdot e.$$

Since $e^\sharp$ is a morphism of $H/\mathsf{Mnd}_c(\mathcal{A})$, we have

$$\sigma = e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inl} : H \to S.$$

From the universal property of the free monad F^{H+V} we conclude

$$e^\sharp = \overline{[\sigma \cdot e^\dagger]} : F^{H+V} \to S, \tag{4.2}$$

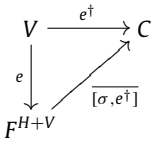
and thus it remains to prove

$$e^\dagger = e^\sharp \cdot e. \tag{4.3}$$

By the definition of $\widehat{e}$ (see (2.18)) we have $e = \widehat{e} \cdot \widehat{\kappa} \cdot \mathsf{inr}$ and since $e^\sharp \cdot \widehat{e} = e^\sharp$ by (3.3) we get

$$\begin{aligned} e^\dagger &= e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr} \\ &= e^\sharp \cdot \widehat{e} \cdot \widehat{\kappa} \cdot \mathsf{inr} \\ &= e^\sharp \cdot e. \end{aligned}$$

(2) To prove the uniqueness, let a solution



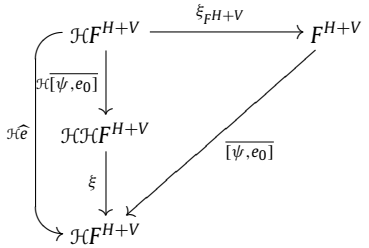
be given. We are going to verify that $\overline{[\sigma, e^\dagger]}$ is a coalgebra homomorphism, i.e.,

$$s \cdot \overline{[\sigma, e^\dagger]} = \mathcal{H}'\overline{[\sigma, s]} \cdot \overline{[\psi, e_0]} \tag{4.4}$$

(see (2.16)). This implies $\overline{[\sigma, e^\dagger]} = e^\sharp$ by Lemma 2.29, thus, the above triangle yields

$$e^\dagger = e^\sharp \cdot e$$

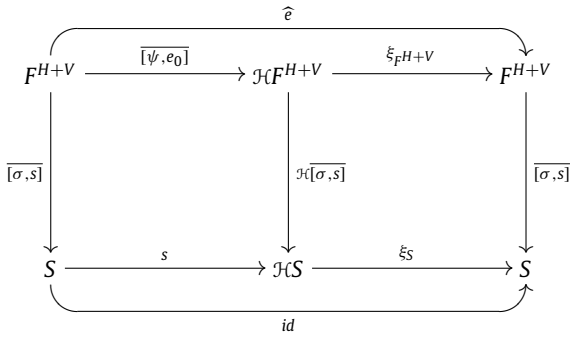
and so our $e^\dagger$ is the morphism of Part (1); see (4.3). From (3.5) and the naturality of ξ (see Remark 3.5) we deduce the commutativity of the diagram



By pasting this triangle and (3.5) along $\xi_{F^{H+V}}$ we obtain

$$\overline{[\psi, e_0]} \cdot \widehat{e} = \mathcal{H}\widehat{e} \cdot \overline{[\psi, e_0]}. \tag{4.5}$$

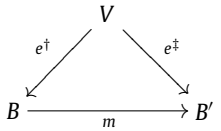
We are ready to prove (4.4): in the following diagram



the upper part is (3.5), the lower one (3.6), and the right-hand square is the naturality of ξ. Thus, the desired left-hand square commutes when extended by ξ_S. Since ξ_S is an isomorphism, the proof of (4.4) is complete. □

We now prove that the monad S is the initial second-order iterative monad w.r.t. H. We regard second-order iterative monads w.r.t. H as a full subcategory of $H/\mathsf{Mnd}_c(\mathscr{A})$. This is a "reasonable" choice of morphisms because each pointed monad morphism preserves solutions of recursive program schemes automatically:

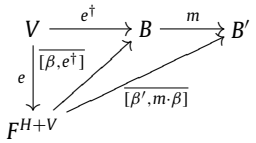
Lemma 4.5. *Every H-pointed monad morphism $m : (B, \beta) \to (B', \beta')$ between second-order iterative monads w.r.t. H preserves solutions: for every guarded recursive program scheme $e : V \to F^{H+V}$ the solution $e^\dagger$ in B is related to the solution $e^\ddagger$ in B' by the commutative triangle*



Proof. Since m is a monad morphism, we clearly have

$$m \cdot \overline{[\beta, e^\dagger]} = \overline{[m \cdot \beta, m \cdot e^\dagger]} : F^{H+V} \to B'.$$

Thus, using that $\beta' = m \cdot \beta$ we get a commutative diagram



which proves that $m \cdot e^\dagger$ is a solution of e in (B', β'). Since solutions are unique, the lemma is proved. □

Lemma 4.6. *Let (B, β) be a second-order iterative monad and $e : V \to F^{H+V}$ a recursive program scheme. Then $m = \overline{[\beta, e^\dagger]} : F^{H+V} \to B$ is the unique morphism of $H/\mathsf{Mnd}_c(\mathscr{A})$ with $m = m \cdot \widehat{e}$; see (2.18).*

Proof. (a) The monad morphism $m = \overline{[\beta, e^\dagger]}$ preserves the H-pointing, see Definition 2.16: $m \cdot \widehat{\kappa} \cdot \mathsf{inl} = \beta$. To prove

$$m = m \cdot \widehat{e} : F^{H+V} \to B$$

we use the freeness of F^{H+V} and verify that both sides are equal when precomposed with $\widehat{\kappa}$.

The left-hand side yields $[\beta, e^\dagger]$. Since $\widehat{e} \cdot \widehat{\kappa} = [\widehat{\kappa} \cdot \mathsf{inl}, e]$, the right-hand side yields $\overline{[\beta, e^\dagger]} \cdot [\widehat{\kappa} \cdot \mathsf{inl}, e]$ which has the left-hand component $[\beta, e^\dagger] \cdot \mathsf{inl} = \beta$ and the right-hand one $\overline{[\beta, e^\dagger]} \cdot e = e^\dagger$. Therefore $m = m \cdot \widehat{e}$.

(b) Conversely, given m with

$$m \cdot \widehat{e} = m \quad \text{and} \quad m \cdot \widehat{\kappa} \cdot \mathsf{inl} = \beta \tag{4.6}$$

(since m preserves the pointing), we are to prove $m = \overline{[\beta, e^\dagger]}$. Let us first prove

$$m \cdot e = e^\dagger. \tag{4.7}$$

In fact, the equality $m = m \cdot \widehat{e}$ yields by (2.18)

$$m \cdot \widehat{\kappa} = m \cdot [\widehat{\kappa} \cdot \mathsf{inl}, e]$$

which proves

$$m \cdot \widehat{\kappa} \cdot \mathsf{inr} = m \cdot e.$$

Consequently, together with (4.6) we get

$$m \cdot \widehat{\kappa} = [\beta, m \cdot e],$$

hence

$$m = \overline{[\beta, m \cdot e]}.$$

By precomposing both sides with e we get

$$m \cdot e = \overline{[\beta, m \cdot e]} \cdot e,$$

i.e., $m \cdot e$ is a solution of e. Thus, by the uniqueness of solutions, (4.7) holds. The desired equality $m = \overline{[\beta, e^\dagger]}$ now follows since m extends the morphism $m \cdot \widehat{\kappa} = [\beta, m \cdot e] = [\beta, e^\dagger]$. □

Theorem 4.7 (*Initial Second-Order Iterative Monad*)**.** *For every finitary endofunctor H the second-order rational monad S is the initial second-order iterative monad. That is, for every second-order iterative monad B w.r.t H there exists a unique H-pointed monad morphism from S to B.*

Proof. We know that S is second-order iterative by Theorem 4.4. Let (B, β) be a second-order iterative monad.

(1) A morphism from $S = \operatorname{colim} \mathsf{EQ}$ to B exists. To prove this, we form for every object $e : V \to F^{H+V}$ of EQ the solution $e^\dagger : V \to B$ in (B, β) and verify that the corresponding morphisms

$$\overline{[\beta, e^\dagger]} : F^{H+V} \to S \quad (e \in \mathsf{EQ})$$

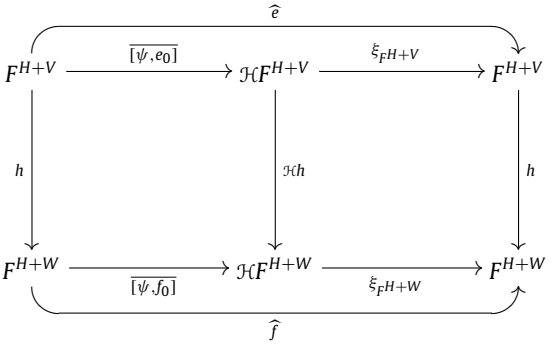
form a cocone of the diagram EQ. The unique factorization through the colimit cocone (2.22) is the desired morphism from S. First observe that $\overline{[\beta, e^\dagger]}$ preserves the pointing (2.12):

$$\overline{[\beta, e^\dagger]} \cdot \widehat{\kappa} \cdot \mathsf{inl} = [\beta, e^\dagger] \cdot \mathsf{inl} = \beta,$$

thus, we have a morphism of $H/\mathsf{Mnd}_f(A)$. For every morphism h of EQ (see the left-hand square in the diagram (4.9) below) we are to prove that the equation

$$\overline{[\beta, f^\dagger]} \cdot h = \overline{[\beta, e^\dagger]} \tag{4.8}$$

holds. The following diagram



(4.9)

commutes: the upper and lower parts commute by (3.5) and the right-hand square is the naturality of ξ. Thus, we proved

$$\widehat{f} \cdot h = h \cdot \widehat{e}. \tag{4.10}$$

Combining with Lemma 4.6 (for f in lieu of e) we conclude

$$\overline{[\beta, f^\dagger]} \cdot h \cdot \widehat{e} = \overline{[\beta, f^\dagger]} \cdot h.$$

From the uniqueness in Lemma 4.6 applied to e we now obtain (4.8). This concludes the proof that the morphisms $\overline{[\beta, e^\dagger]}$ form a cocone of EQ.

(2) Uniqueness. Given a morphism $m : S \to B$ in $H/\mathsf{Mnd}_f(A)$ we are to prove:

$$m \cdot e^\sharp = \overline{[\beta, e^\dagger]} \text{ for all } e \in \mathsf{EQ}.$$

By Lemma 4.6 all we need to check is

$$m \cdot e^\sharp = m \cdot e^\sharp \cdot \widehat{e}.$$

From (4.2) the unique solution $e^\ddagger$ in S satisfies $e^\sharp = \overline{[\sigma, e^\ddagger]}$, thus, Lemma 4.7 yields, when applied to S,

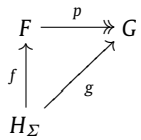
$$e^\sharp \cdot \widehat{e} = e^\sharp$$

which finishes the proof. □

5. Algebraic trees

We now return to the original concept of an algebraic Σ-tree on a given signature Σ, as studied by Bruno Courcelle (see Introduction). We prove that the context-free monad C^{H_Σ} of the polynomial endofunctor H_Σ of Set is indeed precisely the submonad $C^{H_\Sigma} \hookrightarrow T^{H_\Sigma}$ of the Σ-tree monad consisting of all algebraic Σ-trees.

Observation 5.1. Polynomial endofunctors are projective in $\mathsf{Fun}_f(\mathsf{Set})$. That is, for every epimorphism (which means a component wise surjective natural transformation) $p : F \to G$ and every natural transformation $g : H_\Sigma \to G$ there exists a natural transformation $f : H_\Sigma \to F$ with $g = p \cdot f$:



In the case where Σ consists of a single n-ary symbol, this follows from the Yoneda Lemma, since $H_\Sigma \cong \mathsf{Set}(n, -)$: the natural transformation g corresponds to an element of Gn, and we find its inverse image (under p_n) in Fn, giving us $f : H_\Sigma \to F$. If Σ has more symbols, apply the Yoneda Lemma to each of them separately.

Theorem 5.2. *For every signature Σ we have:*

C^{H_Σ} is the monad of algebraic Σ-trees.

Proof. Throughout the proof we write H in lieu of H_Σ and C in lieu of C^{H_Σ}.

(1) We prove that every element of CX lies in the image of $e^\ddagger$ for some guarded recursive program scheme

$$e : H_\Phi \to F^{H+H_\Phi}$$

where $e^\ddagger$ is the unique solution in C; see Theorem 4.3.

Indeed, since S is the filtered colimit of EQ_1 (see Corollary 2.28) and filtered colimits of finitary functors in $\mathsf{Mnd}_c(\mathscr{A})$ (and thus also in $H/\mathsf{Mnd}_c(\mathscr{A})$) are computed on the level of the underlying functors (in other words: filtered colimits are formed object wise in $\mathscr{A}$), we have for every set X a colimit cocone

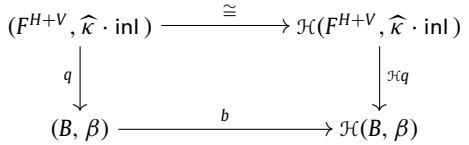
$$b^\sharp_X : BX \to SX$$

where $b : (B, \beta) \to \mathcal{H}(B, \beta)$ ranges over all coalgebras in EQ_1 and $b^\sharp : B \to S$ is the colimit cocone.

Since EQ_1 is a closure of EQ under coequalizers, every object of EQ_1 is a quotient of one in EQ. Thus, we have a guarded recursive program scheme

$$e : V \to F^{H+V} \tag{5.1}$$

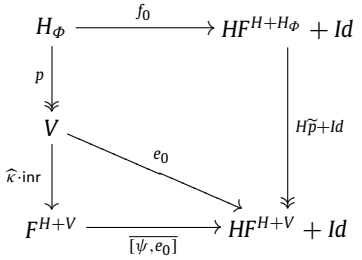
and an epimorphic coalgebra homomorphism for $\mathcal{H}$ (see (2.16)):



Since V is a finitely presentable endofunctor, there exists by Example 2.1(2) a finite signature Φ and an epimorphic natural transformation

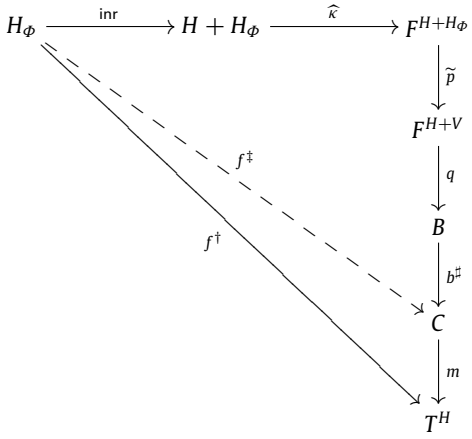
$$p : H_\Phi \to V.$$

The free-monad functor takes $H + p : H + H_\Phi \to H + V$ to a monad morphism $\widetilde{p} : F^{H+H_\Phi} \to F^{H+V}$ which is also an epimorphism (since the free-monad functor is a left adjoint). It follows that $\widetilde{p}$ is epimorphic as a natural transformation, thus, $H\widetilde{p} + Id$ is epimorphic. Due to the projectivity of H_Φ we obtain a natural transformation f_0 making the diagram



commutative (see Observation 5.1). Here f_0 is the guard of a "classical" guarded recursive program scheme $f : H_\Phi \to F^{H+H_\Phi}$ and for the corresponding $\mathcal{H}$-coalgebra on F^{H+H_Φ}, see Remark 2.17, the above monad morphism $\widetilde{p}$ is a coalgebra homomorphism.

Consider $f^\dagger$ (see Theorem 2.21) and $f^\ddagger$ (see Theorem 4.3). We shall now prove that the diagram below commutes:



Indeed, recall from (2.20) that the coalgebra homomorphism f^* fulfills

$$f^\dagger = f^* \cdot \widehat{\kappa} \cdot \mathsf{inr},$$

and so we only need to notice that the vertical arrow, being a coalgebra homomorphism, is equal to f^*. Since m is a monomorphism, the upper triangle also commutes. Thus, every element in the image of $b^\sharp_X$ lies in the image of $f^\ddagger_X$ for the above recursive program scheme f.

(2) We will verify that $m_X : CX \hookrightarrow TX$ consists precisely of the algebraic Σ-trees with leaves labeled by constant symbols and elements of X. Indeed, every context-free Σ-tree has the form

$$t = e^\dagger_X(x)$$

for some guarded recursive program scheme $e : H_\Phi \to F^{H+H_\Phi}$ and since $e^\dagger_X = m_X \cdot e^\ddagger_X$, the tree t lies in CX.

Conversely every element of CX has, by item (1) above, the form $e^\ddagger_X(x)$ for some guarded recursive program scheme $e : H_\Phi \to F^{H+H_\Phi}$. □

6. Conclusions and open problems

The aim of our paper was to construct for a finitary endofunctor H a monad expressing solutions of recursive program schemes of type H. We hoped originally to achieve what we managed to do for the first-order recursive equations of type H in previous work [4]: there we defined the rational monad R^H based on solutions of recursive equations, we proved that R^H is iterative (and, in particular, ideal) in the sense of Calvin Elgot, and we characterized R^H as the free iterative monad on H. From this we derived, in the case of endofunctors of Set, that R^H is closed under second-order substitution. Moreover, the construction worked for all locally finitely presentable base categories.

In this paper, we also exhibited a general construction: for every finitary endofunctor H we provided

(1) a second-order monad S^H defined via colimits of all recursive program schemes; this is the initial second-order iterative monad

and

(2) a context-free monad C^H, a quotient of S^H which is a submonad of the free completely iterative monad T^H. Also this monad C^H is second-order iterative: for all recursive program schemes the existence and uniqueness of solutions was derived from the corresponding more general solution theorem of Ghani et al. [14].

In the case where H is actually a polynomial endofunctor of Set associated to a signature Σ, our monad C^H coincides with the monad of algebraic trees of Bruno Courcelle [11]. However, whereas Courcelle proved that the monad of algebraic trees is iterative, we only proved that the monads S^H and C^H are ideal.

In fact, as soon as C^H would be proved to be iterative, the intuition says that this is not enough: the next open problem is, then, whether C^H is closed under second-order substitution in the sense of [24]. Again, this was, for context-free Σ-trees, proved by Bruno Courcelle.

It remains to be seen whether C^H can be characterized by some universal property, too. Unfortunately, algebraic trees cannot serve as a guiding example in this respect as no universal property of them is known.

References

[1] P. Aczel, J. Adámek, S. Milius, J. Velebil, Infinite trees and completely iterative theories: a coalgebraic view, Theoret. Comput. Sci. 300 (2003) 1–45.
[2] P. Aczel, J. Adámek, J. Velebil, A coalgebraic view of infinite trees and iteration, in: Proc. Coalgebraic Methods in Computer Science, CMCS'01, Electron. Notes Theor. Comput. Sci., vol. 44, 2001, pp. 1–26.
[3] J. Adámek, Free algebras and automata realizations in the language of categories, Comment. Math. Univ. Carolin. 15 (1974) 589–602.
[4] J. Adámek, S. Milius, J. Velebil, Iterative algebras at work, Math. Structures Comput. Sci. 16 (6) (2006) 1085–1131.
[5] J. Adámek, S. Milius, J. Velebil, Semantics of higher-order recursion schemes, Log. Methods Comput. Sci. 7 (1:15) (2011) 43.
[6] J. Adámek, S. Milius, J. Velebil, Recursive program schemes and context-free monads, in: Proc. Coalgebraic Methods in Computer Science, CMCS'10, Electron. Notes Theor. Comput. Sci. 264 (2010) 3–23.
[7] J. Adámek, S. Milius, J. Velebil, Iterative reflections of monads, Math. Structures Comput. Sci. 20 (3) (2010) 419–452.
[8] J. Adámek, S. Milius, J. Velebil, Some remarks on finitary and iterative monads, Appl. Categ. Structures 11 (6) (2003) 521–541.
[9] J. Adámek, J. Rosický, Locally Presentable and Accessible Categories, Cambridge University Press, 1994.
[10] M. Barr, Coequalizers and free triples, Math. Z. 116 (1970) 307–322.
[11] B. Courcelle, Fundamental properties of infinite trees, Theoret. Comput. Sci. 25 (1983) 95–169.
[12] C.C. Elgot, Monadic computation and iterative algebraic theories, in: H.E. Rose, J.C. Sheperdson (Eds.), Logic Colloquium '73, North-Holland Publishers, Amsterdam, 1975.
[13] J.H. Gallier, DPBS'a in atomic normal form and applications to equivalence problems, Theoret. Comput. Sci. 14 (1981) 155–186.
[14] N. Ghani, C. Lüth, F. De Marchi, Solving algebraic equations using coalgebra, Theor. Inform. Appl. 37 (2003) 301–314.
[15] N. Ghani, C. Lüth, F. De Marchi, Monads of coalgebras: rational terms and term graphs, Math. Structures Comput. Sci. 15 (3) (2005) 433–451.
[16] N. Ghani, C. Lüth, F. De Marchi, A.J. Power, Algebras, coalgebras, monads and comonads, in: Proc. Coalgebraic Methods in Computer Science, CMCS'01, Electron. Notes Theor. Comput. Sci., vol. 44, 2001, pp. 128–145.
[17] N. Ghani, C. Lüth, F. De Marchi, A.J. Power, Dualizing initial algebras, Math. Structures Comput. Sci. 13 (2) (2003) 349–370.
[18] N. Ghani, T. Uustalu, Coproducts of ideal monads, Theor. Inform. Appl. 38 (4) (2004) 321–342.
[19] S. Ginali, Regular trees and the free iterative theory, J. Comput. System Sci. 18 (1979) 228–242.
[20] I. Guessarian, Algebraic Semantics, in: Lecture Notes in Comput. Sci., vol. 99, Springer, 1981.
[21] S. Lack, On the monadicity of finitary monads, J. Pure Appl. Algebra 140 (1999) 65–73.
[22] J. Lambek, A fixpoint theorem for complete categories, Math. Z. 103 (1968) 151–161.
[23] S. Milius, Completely iterative algebras and completely iterative monads, Inform. and Comput. 196 (2005) 1–41.
[24] S. Milius, L.S. Moss, The category theoretic solution of recursive program schemes, Theoret. Comput. Sci. 366 (2006) 3–59.
[25] L.S. Moss, Parametric corecursion, Theoret. Comput. Sci. 260 (1–2) (2001) 139–163.