Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 264 (2010) 3–23

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Recursive Program Schemes and Context-Free 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 of Prague, Czech Republic

Abstract

Solutions of recursive program schemes over a given signature Σ were characterized by Bruno Courcelle as precisely the *context-free* (or *algebraic*) Σ-trees. These are the finite and infinite Σ-trees yielding, via labelling of paths, context-free languages. Our aim is to generalize this to finitary endofunctors H of general categories: we construct a monad C^H "generated" by solutions of recursive program schemes of type H, and prove that this monad is ideal. In case of polynomial endofunctors of Set our construction precisely yields the monad of context-free Σ-trees of Courcelle. Our result builds on a result by N. Ghani et al on solutions of algebraic systems.

Keywords: algebraic trees, recursive program schemes, ideal theory, monads

1 Introduction

The aim of the current paper is to introduce, for a finitary endofunctor H of a "reasonable" category, the context-free monad C^H of H characterizing solutions of recursive program schemes of type H. 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 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 [17]. In contrast, C^H is given by the algebraic trees investigated in the pioneering paper of Bruno Courcelle [10]. We call these trees t *context-free* since in [10] they are characterized by the property that a certain natural language associated to the paths of t is context-free (whereas t is rational iff that language is regular).

Recall that a *recursive program scheme* (or rps for short) defines new operations $\varphi_1, \dots, \varphi_k$ of given arities $n_1, \dots, n_k$ recursively, using given operations represented

---

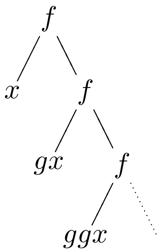
[1] Supported by the grant MSM 6840770014 of the Ministry of Education of the Czech Republic.

doi:10.1016/j.entcs.2010.07.011

by symbols from a signature Σ. Here is an example:

$$\varphi(x) = f(x, \varphi(gx)) \tag{1}$$

is a recursive program scheme defining a unary operation φ from the givens in $\Sigma = \{ f, g \}$ with f binary and g unary. The semantics of recursive program schemes is a topic at the heart of theoretical computer science, see [10,18]. 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



(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 signature of the newly defined operations 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 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,

$$\varphi_i(x_0, \ldots, x_{n-1}) = t_i \qquad (i = 1, \ldots, k) \tag{3}$$

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 [10].

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 (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 $t^\dagger(x)$

---

[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, Σ, say. See [10] or [22] for a category-theoretic description.

from (2) satisfies the corresponding equality of trees

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

This concept of solutions was formalized in [22] by means of the free completely iterative monad T^H on a functor H; in case $H = H_\Sigma$ this is the monad of all 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 (and renaming) of the concept of an algebraic tree of Courcelle [10]:

Definition 1.1 A Σ-tree is called *context-free* if there exists a recursive program scheme (3) such that $t = t_1^{\dagger}$.

Example 1.2 Every rational tree is context-free, and (2) shows a context-free tree that is not rational.

Courcelle proved that the monad C^{H_Σ} of all context-free Σ-trees as a submonad of T^{H_Σ} is iterative in the sense of Calvin Elgot [11]. Furthermore, context-free trees are closed under second-order substitution. The aim of the present paper is a construction of the context-free monad C^H for all finitary endofunctors H of locally finitely presentable categories. We prove that this monad is always ideal, i. e., it 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. [22,6,16]. 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.

Related work. Our work is based on the pioneering paper by Bruno Courcelle [10]. As we mentioned already, Irène Guessarian [18] 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 [23] and also appears independently and almost at the same time in the work of Neil Ghani et al [14] (see also [15]) and Peter Aczel et al [2] (see also [1]). Ghani et al [12] 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 that is a fixed point w. r. t. second-order substitution. The ideas of [12] were taken further in [22]; 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).

The present paper builds on ideas in [12,22]. Our construction of the context-free monad is new. It is inpired by the construction of the rational monad in [4], see also [13] for a more general construction.

2 Construction of the context-free monad

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

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

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

(i) 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.

(ii) If $\mathscr{A}$ is locally finitely presentable, then so is $\mathsf{Fun}_f(\mathscr{A})$, the category of all finitary endofunctors and natural transformations. In case $\mathscr{A} = \mathsf{Set}$, the polynomial endofunctor

$$H_\Sigma X = \coprod_{\sigma\in\Sigma} X^n \qquad n = \text{arity of } \sigma \tag{4}$$

is a finitely presentable object of $\mathsf{Fun}_f(\mathsf{Set})$ iff Σ is a finite set. This is easily seen using 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 objectwise and natural transformations are monic iff their components are monic.

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:

(i) The category

$$\mathsf{Mon}_f(\mathscr{A})$$

of all finitary monads on $\mathscr{A}$ and monad morphisms. This is a locally finitely presentable category. Indeed, as observed by Steve Lack [19], the forgetful functor

$$\mathsf{Mon}_f(\mathscr{A}) \to \mathsf{Fun}_f(\mathscr{A})$$

is finitary and monadic, thus, the local finite presentability of $\mathsf{Fun}_f(\mathscr{A})$ implies that of $\mathsf{Mon}_f(\mathscr{A})$, see [8], 2.78. It follows that filtered colimits of finitary monads are formed object-wise on the level of $\mathscr{A}$.

(ii) We will also make use of the fact that for every locally finitely presentable category $\mathscr{B}$ and object B the coslice category $B/\mathscr{B}$ of all morphisms with domain B is a locally finitely presentable category, see [8], 2.44.

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 $\mathscr{A}$. Denote by $\widehat{\eta}_X : X \to F^H X$ the universal arrow. As proved by M. Barr [9] the corresponding monad on $\mathscr{A}$

$$F^H$$

of free H-algebras 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

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

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

$$\begin{tikzcd} H \arrow[r, "\widehat{\kappa}"] \arrow[dr, "f"'] & F^H \arrow[d, "\overline{f}"] \\ & S \end{tikzcd} \tag{5}$$

Moreover, from [3] we have

$$F^H = HF^H + Id \quad \text{with injections } \varphi \text{ and } \widehat{\eta}. \tag{6}$$

Remark 2.3 The category $\mathsf{Mon}_f(\mathscr{A})$, 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

$$F^{H+K} = F^H \oplus F^K. \tag{7}$$

We shall use the same notation φ, $\widehat{\eta}$ and $\widehat{\kappa}$ for different 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

$$T^H X \to H\Big(T^H X\Big) + X. \tag{8}$$

By Lambek's lemma [20], this morphism is invertible, and we denote the components of the inverse by

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

respectively.

Notation 2.4 *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 [1,21]. The above natural transformation $\tau : HT \to T$ yields the universal arrow

$$\kappa = (H \xrightarrow{H\eta} HT \xrightarrow{\tau} T) \tag{9}$$

Moreover, in analogy to (6) above, we have

$$T = HT + Id \qquad \text{with injections } \tau \text{ and } \eta. \tag{10}$$

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

$$\begin{tikzcd} HTT \arrow[r, "\tau T"] \arrow[d, "H\mu"'] & TT \arrow[d, "\mu"] \\ HT \arrow[r, "\tau"'] & T \end{tikzcd} \tag{11}$$

Notation 2.5 (i) We denote by $\mathsf{Mon}(\mathscr{A})$ the category of all monads on $\mathscr{A}$ (which is usually not locally presentable). Given a finitary endofunctor H let

$$H/\mathsf{Mon}(\mathscr{A})$$

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

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

For example, F^H and T are H-pointed monads (via the universal arrows).

(ii) For every H-pointed monad (S, σ) we write

$$b = [\mu^S \cdot \sigma S, \eta^S] : HS + Id \to S.$$

Lemma 2.6 (Ghani et al [13]) *For every H-pointed monad (S, σ) the endofunctor $HS+Id$ carries a canonical monad structure whose unit is the coproduct injection*

$\mathsf{inr} : Id \to HS + Id$ *and whose multiplication is given by*

$$\begin{tikzcd}
(HS + Id)(HS + Id) \arrow[d, equal] \\
HS(HS + Id) + HS + Id \arrow[d, "HSb + HS + Id"] \\
HSS + HS + Id \arrow[d, "{[H\mu^S, HS] + Id}"] \\
HS + Id
\end{tikzcd} \tag{12}$$

Remark 2.7 For $HS + Id$ we also have an obvious H-pointing

$$\mathsf{inl} \cdot H\eta^S : H \to HS + Id. \tag{13}$$

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

$$\mathcal{H}(S, \sigma) = (HS + Id, \mathsf{inl} \cdot H\eta^S),$$

see [13] or [22], Lemma 5.2 for details.

Example 2.8 For every finitary endofunctor V we consider F^{H+V} as an H-pointed monad via

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

And $\mathcal{H}(F^{H+V}) = HF^{H+V} + Id$ is then an H-pointed monad via (13) which has the form

$$\psi = (H \xrightarrow{H\widehat{\eta}} HF^{H+V} \xrightarrow{\mathsf{inl}} HF^{H+V} + Id). \tag{14}$$

The proof of the following theorem is similar to the proof of Lemma 2.6 in [13]. The precise statement using the category $H/\mathsf{Mon}(\mathscr{A})$ can be found in [22], Theorem 5.4.

Theorem 2.9 *The terminal coalgebra for $\mathcal{H}$ is given by the H-pointed monad T, H-pointed as in (9), with the coalgebra structure $T \xrightarrow{\sim} \mathcal{H}T$ from (8).*

Definition 2.10 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 (6):

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

that is, we have a commutative triangle

$$\begin{tikzcd}
V \arrow[r, "e"] \arrow[dr, dashed, "e_0"'] & F^{H+V} \\
& HF^{H+V} + Id \arrow[u, "{[\varphi \cdot \mathsf{inl}\,, \widehat{\eta}]}"']
\end{tikzcd} \tag{15}$$

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 $\mathscr{A}$ has monic coproduct injections.

Example 2.11 In case of a polynomial endofunctor $H = H_\Sigma : \mathsf{Set} \to \mathsf{Set}$ every recursive program scheme (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 2.10: in lieu of a general finitely presentable endofunctor V, which is a quotient of H_Σ (cf. Example 2.1(iv)), we just take $V = H_\Sigma$.

The system (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), are guarded. The unguarded ones such as $f(x) = f(x)$ are to be avoided if we want to work with unique solutions.

Definition 2.12 By a *solution* of a recursive program scheme $e : V \to F^{H+V}$ in an H-pointed monad (S, σ) is meant a natural transformation $e^\dagger : V \to S$ such that the unique monad morphism extending $[\sigma, e^\dagger] : H + V \to S$ (see (5)) makes the triangle below commutative:

$$\begin{tikzcd}
V \arrow[r, "e^\dagger"] \arrow[d, "e"'] & S \\
F^{H+V} \arrow[ur, "\overline{[\sigma, e^\dagger]}"']
\end{tikzcd} \tag{16}$$

Remark 2.13 (1) Every guarded recursive program scheme (15) 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 (14), 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

$$\overline{[\psi, e_0]} : F^{H+V} \to \mathcal{H}(F^{H+V}) \tag{17}$$

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.

(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 the left-hand component of $r \cdot \widehat{\kappa}$ being 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.

(3) For the terminal coalgebra T for $\mathcal{H}$, see Theorem 2.9, we thus obtain the unique coalgebra homomorphism

$$e^* : F^{H+V} \to T. \tag{18}$$

Remark 2.14 Our concept of a recursive program scheme is a special case of the algebraic systems studied by Neil Ghani et al [12]. Let us recall from that paper that

(i) an H-pointed monad is called *coalgebraic* if it is isomorphic to the monad $HS + Id$ of Lemma 2.6 via $b : HS + Id \to S$ in Notation 2.5(ii),

(ii) examples of coalgebraic monads include F^H, see (6), and T, see (10),

(iii) T is the final coalgebraic monad; we denote by $u_S : S \to T$ the unique morphism for a coalgebraic monad (S, σ),

(iv) an *algebraic system* is given by a finitary monad E, a finitary coalgebraic monad (S, σ) and a monad morphism

$$e : E \to H(S \oplus E) + Id,$$

(v) a *solution* of e is a monad morphism $s : E \to T$ such that the square below commutes:

$$\begin{tikzcd}
E \arrow[r, "s"] \arrow[d, "e"'] & T \arrow[d, "{[\tau,\eta]^{-1}}"] \\
H(S \oplus E) + Id \arrow[r, "{H([u_S,s])+Id}"'] & HT + Id
\end{tikzcd}$$

Theorem 2.15 (Ghani et al [12]) *Every algebraic system has a unique solution.*

This gives a solution theorem for recursive program schemes as follows: due to (7) we have the morhism $e_0 : V \to H(F^H \oplus F^V) + Id$ in (15) yielding an algebraic system via (5):

$$\overline{e_0} : F^V \to H(F^H \oplus F^V) + Id. \tag{19}$$

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

Theorem 2.16 *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 $e^* : F^{H+V} \to T$ by*

$$e^\dagger = (V \xrightarrow{\mathsf{inr}} H + V \xrightarrow{\widehat{\kappa}} F^{H+V} \xrightarrow{e^*} T). \tag{20}$$

Indeed, for the unique solution $s : F^V \to T$ of the algebraic system $\overline{e_0}$ in (19) above we obtain a solution $e^\dagger$ in the sense of Definitinon (2.10) by composing with $\widehat{\kappa} : V \to F^V$:

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

The proof that (16) commutes is performed using some diagram chasing. A somewhat subtle point is that for $u_S : S \to T$ (see Remark 2.14(iii)) we have the equality

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

Here the square brackets on the left refer to the coproduct of F^H and F^V in $H/\mathsf{Mon}(\mathscr{A})$ 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 (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 2.10 its extension $\overline{e^\dagger} : F^V \cdot T$ is a solution of the corresponding algebraic system $\overline{e_0}$.

Remark 2.17 It is our goal to define a submonad C of T formed by all solutions of recursive program schemes of type H. We do this in two steps.

(i) A finitary monad $\widetilde{C}$ together with a monad morphism $\widetilde{c} : \widetilde{C} \to T$ is constructed by forming a colimit of coalgebras for the endofunctor $\mathcal{H}$ obtained from all recursive program schemes.

(ii) The (strong epi, mono)-factorization (cf. Proposition 2.19 below) of $\widetilde{c}$ is formed to obtain the desired submonad:

$$\begin{tikzcd} \widetilde{C} \arrow[rr, "\widetilde{c}"] \arrow[dr, two heads, "k"'] & & T \\ & C \arrow[ur, tail, "c"'] & \end{tikzcd}$$

Unfortunately, $\mathsf{Mon}(\mathscr{A})$ need not have such factorizations in general. We therefore need to work in the category

$$\mathsf{Mon}_{\mathsf{acc}}(\mathscr{A})$$

of all monads on $\mathscr{A}$ that are *accessible*, that is, the underlying functors preserve, for some infinite cardinal λ, λ-filtered colimits. (Recall that a λ-filtered category is such that every subcategory with less than λ objects and morphisms has a cocone in it.)

Here is our basic example of an accessible but not finitary monad:

Lemma 2.18 *For every finitary endofunctor H the monad T (see Notation 2.4) is accessible.*

Proof. It is proved in Proposition 5.16 of [4] that TZ can be constructed as the colimit of the diagram of all coalgebras for $H(-) + Z$ carried by all countably presentable objects. Thus, T coincides with the $\aleph_1$-accessible monad $R^{\aleph_1}$ of loc. cit. □

Proposition 2.19 *The category $\mathsf{Mon}(\mathscr{A})$ has as monomorphisms precisely the monad morphisms with monic components. The subcategory $\mathsf{Mon}_{\mathsf{acc}}(\mathscr{A})$ has (strong epi, mono)-factorizations and is closed in $\mathsf{Mon}(\mathscr{A})$ under strong epimorphisms and monomorphisms.*

Proof. (1) The category $\mathsf{Fun}(\mathscr{A})$ of all endofunctors on $\mathscr{A}$ has a generator formed by all accessible functors. In fact, let $u, v : K \to L$ be distinct natural transformations. Then $u_A \neq v_A$ for some object A. Since $\mathscr{A}$ is locally finitely presentable, A is λ-presentable for some λ, see [8]. Thus, A lies in the small full subcategory $E : \mathscr{A}_\lambda \hookrightarrow \mathscr{A}$ representing all λ-presentable objects. The functor K has a λ-accessible coreflection $c : K' \to K$ obtained as the left Kan extension of $K \cdot E$ along E. Since $A \in \mathscr{A}_\lambda$ implies that c_A is an isomorphism, we conclude that $u \cdot c \neq v \cdot c$, as desired.

(2) The first statement of our proposition follows from the fact that every monomorphism $m : P \to Q$ in $\mathsf{Mon}(\mathscr{A})$ is monomorphic in $\mathsf{Fun}(\mathscr{A})$. By item (1), we only need to consider $u, v : K \to P$ with $m \cdot u = m \cdot v$ where K is λ-accessible. Then free K-algebras exist, see [3]. Therefore a free monad F^K exists, cf. [9]. The corresponding monad morphisms $\overline{u}, \overline{v} : F^K \to P$ (cf. (5)) fulfil $m \cdot \overline{u} = m \cdot \overline{v}$. 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.

(3) The category $\mathsf{Mon}_\lambda(\mathscr{A})$ of all λ-accessible monads is closed under monomorphisms in $\mathsf{Mon}(\mathscr{A})$ since (by the same argument as in item (2)) monomorphisms in $\mathsf{Mon}_\lambda(\mathscr{A})$ are precisely the morphisms that are collectively monic. And it is closed under strong epimorphisms in $\mathsf{Mon}(\mathscr{A})$ since this subcategory is coreflective; indeed, all left adjoints preserve strong epimorphisms. For $\lambda = \aleph_0$ this was proved in [7], and for general λ the proof is (easy and) completely analogous.

(4) The category $\mathsf{Mon}_\lambda(\mathscr{A})$ is locally λ-presentable and therefore (strong epi,mono)-factorizations exist, see [8]. From item (3) it now follows that also $\mathsf{Mon}_{\mathsf{acc}}(\mathscr{A})$ has (strong epi, mono) factorizations and is closed under monos and strong epis in $\mathsf{Mon}(\mathscr{A})$. □

Corollary 2.20 *The functor $\mathcal{H}$ preserves monomorphisms.*

Indeed, given a monomorphism $m : (S, \sigma) \to (S', \sigma')$ in $H/\mathsf{Mon}(\mathscr{A})$, then m is componentwise 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 $\mathscr{A}$).

Construction 2.21 The H-pointed monad $\widetilde{C}^H$. For every guarded recursive program scheme (15) consider F^{H+V} as a coalgebra for the functor $\mathcal{H}$, see (17).

We denote by

$$\mathsf{EQ}_0 \subseteq \mathsf{Coalg}\,\mathcal{H}$$

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

$$\widetilde{C}^H = \operatorname{colim} \mathsf{EQ}_0 \qquad (\text{in } \mathsf{Coalg}\,\mathcal{H}).$$

Thus, we have a finitary monad $\widetilde{C}$ with an H-pointing and a coalgebra structure denoted by

$$\widetilde{\rho} : H \to \widetilde{C}^H \qquad \text{and} \qquad \widetilde{r} : \widetilde{C}^H \to \mathcal{H}(\widetilde{C}^H)$$

respectively, together with a colimit cocone

$$e^\sharp : F^{H+V} \to \widetilde{C}^H \qquad \text{for all rps } e : V \to F^{H+V},$$

formed by coalgebra homomorphisms for $\mathcal{H}$ preserving the pointing (14), i. e. with

$$\widetilde{\rho} = e^\sharp \cdot (\widehat{\kappa} \cdot \mathsf{inl}\,) \qquad \text{for every } e.$$

We see in the next lemma that EQ_0 is a connected category. Since the forgetful functors

$$\mathsf{Coalg}\,\mathcal{H} \to H/\mathsf{Mon}(\mathscr{A}) \to \mathsf{Mon}(\mathscr{A})$$

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

Lemma 2.22 EQ_0 *is closed under finite coproducts in* $\mathsf{Coalg}\,\mathcal{H}$.

Proof. Consider two objects of EQ_0 determined by

$$e : V \to HF^{H+V} + Id \qquad \text{and} \qquad e' : V' \to HF^{H+V'} + Id$$

The coproduct injections $i : H+V \to H+V+V'$ and $i' : H+V' \to H+V+V'$ yield corresponding monad morphisms $\widetilde{i} : F^{H+V} \to F^{H+V+V'}$ and $\widetilde{i'} : F^{H+V'} \to F^{H+V}$. Denote by

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

the canonical morphism. We prove that the object $f : V + V' \to F^{H+V+V'}$ of EQ_0 determined by

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

is the coproduct of the two given objects.

We know from Remark 2.13 that morphisms from the above object into an $\mathcal{H}$-coalgebra $X = ((S,s),p)$ are given by natural transformations

$$t : V + V' \to S$$

such that the extension $\overline{[s,t]} : F^{H+V+V'} \to S$ of the transformation $[s,t] : H + V + V' \to S$ to a monad morphism fulfils

$$p \cdot r = (H\overline{[s,t]} + Id) \cdot f.$$

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

(i) the left-hand component $q : V \to S$ of r gives rise to a morphism of $\mathsf{Coalg}\,\mathcal{H}$ from the object determined by e_0 into X

(ii) and the right-hand component $q' : V' \to S$ yields a morphism from the object determined by e_0' into X.

For that observe first that the diagram

$$\begin{tikzcd}
F^{H+V} \arrow[r, "\widetilde{i}"] \arrow[dr, "\overline{[s,q]}"'] & F^{H+V+V'} \arrow[d, "\overline{[s,t]}"'] & F^{H+V'} \arrow[l, "\widetilde{i'}"'] \arrow[dl, "\overline{[s,q']}"] \\
 & S &
\end{tikzcd}$$

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

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

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

$$\begin{tikzcd}
V + V' \arrow[rrr, "f"] \arrow[dd, "t"'] & & & HF^{H+V+V'} + Id \arrow[dd, "H\overline{[s,t]}+Id"] \\
 & V \arrow[ul, dashed, "\mathsf{inl}"'] \arrow[dl, dashed, "q"] \arrow[r, dashed, "e_0"'] & HF^{H+V} + Id \arrow[ur, dashed, "H\widetilde{i}+Id"] \arrow[dr, dashed, "H\overline{[s,q]}+Id"'] & \\
S \arrow[rrr, "p"'] & & & HS + Id
\end{tikzcd}$$

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

Corollary 2.23 *$\widetilde{C}^H$ is a filtered colimit of the closure* EQ *of* EQ_0 *under coequalizers in* $\mathsf{Coalg}\,\mathcal{H}$*.*

Indeed, since EQ_0 is closed under finite coproducts, EQ is closed under finite colimits, thus, it is filtered. And $\operatorname{colim} \mathsf{EQ} \cong \operatorname{colim} \mathsf{EQ}_0$.

Definition 2.24 The context-free monad C^H. Denote by

$$\widetilde{c} : \widetilde{C}^H \to T$$

the unique coalgebra homomorphism (see Theorem 2.9) and define the *context-free monad* of H as the submonad C^H of T obtained by the following (strong epi, mono)-factorization of $\widetilde{c}$ in $\mathsf{Mon}(\mathscr{A})$:

$$\begin{tikzcd}
 & C^H \arrow[d, tail, "c"] \\
\widetilde{C}^H \arrow[ur, two heads, "k"] \arrow[r, "\widetilde{c}"'] & T
\end{tikzcd}$$

Remark 2.25 (i) Since $\widetilde{C}^H$ is finitary and T accessible, see Lemma 2.18, we have the desired factorization by Proposition 2.19.

(ii) The context-free monad is pointed: The pointing $\widetilde{\rho} : H \to \widetilde{C}^H$ of $\widetilde{C}^H$ yields the pointing

$$\rho = k \cdot \widetilde{\rho} : H \to C^H$$

of C^H which c preserves (because $\widetilde{c}$ is a morphism of $H/\mathsf{Mon}(\mathscr{A})$).

(iii) Analogously to T we shall write C and $\widetilde{C}$ without the upper index H from now on.

Observation 2.26 The functor $\mathcal{H}$ preserves monomorphisms by Corollary 2.20, thus, C carries a canonical structure r of an $\mathcal{H}$-coalgebra derived from the structure $\widetilde{r}$ for $\widetilde{C}$:

$$\begin{tikzcd}
\widetilde{C} \arrow[r, two heads, "k"] \arrow[d, "\widetilde{r}"'] & C \arrow[d, "c"] \arrow[ddl, "r"'] \\
\mathcal{H}\widetilde{C} \arrow[d, "\mathcal{H}k"'] & T \arrow[d, "\simeq"] \\
\mathcal{H}C \arrow[r, tail, "\mathcal{H}c"'] & \mathcal{H}T
\end{tikzcd} \tag{21}$$

Indeed, recall that $c \cdot k = \widetilde{c}$ 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 r.

Theorem 2.27 *Every guarded recursive program scheme $e : V \to F^{H+V}$ has a unique solution in the context-free monad of H.*

Proof. 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 $e^\ddagger = \overline{[\rho, e^\ddagger]} \cdot e$. Recall that the colimit injection $e^\sharp : F^{H+V} \to \widetilde{C}$ in Construction 2.21 is a coalgebra homomorphism for $\mathcal{H}$, hence, so is $\widetilde{c} \cdot e^\sharp$, which proves

$$e^* = \widetilde{c} \cdot e^\sharp,$$

see Theorem 2.16 (because T is a terminal coalgebra by Theorem 2.9). Therefore, by (20) we have

$$e^\dagger = \widetilde{c} \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr} = c \cdot k \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr}\,.$$

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

$$\begin{tikzcd}
V \arrow[r, "e^\ddagger"] \arrow[rr, bend left, "e^\dagger"] \arrow[d, "e"'] & C \arrow[r, "c"] & T \\
F^{H+V} \arrow[ur, "\overline{[\rho,e^\ddagger]}"'] \arrow[urr, bend right, "\overline{[\kappa,e^\dagger]}"'] & &
\end{tikzcd}$$

the outside commutes, see (16) with $\sigma = \kappa$, and the right-hand part does since $\kappa = c \cdot \rho$ (see Definition 2.24). Consequently, the left-hand triangle commutes: recall from Definition 2.24 that c is a monomorphism.

The uniqueness follows from the same diagram: if the left-hand triangle commutes, so does the outside, and since $e^\dagger$ is uniquely determined (see Theorem 2.16), we conclude $e^\dagger = c \cdot e^\ddagger$. Finally, use again that c is monic. □

3 The context-free monad is ideal

Under the assumptions of Section 2 we prove that C is an ideal monad in the sense of C. Elgot [11] 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 sixtuple $(S, \eta, \mu, S', \sigma, \mu')$ where (S, η, μ) is a monad,

$$\sigma : S' \to S \qquad (\text{“the ideal”})$$

is a subfunctor such that $S = S' + Id$ with injection σ and η, and

$$\mu' : S'S \to S'$$

is a natural transformation restricting μ in the sense that

$$\mu \cdot \sigma S = \sigma \cdot \mu'$$

Example 3.2

(i) The free monad F^H is ideal: its ideal is HF^H, see (6).

(ii) The free completely iterative monad T is ideal: its ideal is HT, see (10).

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 $r : C \to HC + Id$, see Observation 2.26, is (analogously to the two examples F^H and T above) invertible, as we prove below: its inverse is the morphism

$$b \equiv HC + Id \xrightarrow{\rho C + Id} CC + Id \xrightarrow{[\mu^C, \eta^C]} C, \tag{22}$$

cf. Notation 2.5(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 H.*

Proof. We first prove $r = b^{-1}$.

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

$$\begin{tikzcd}
C \arrow[r, "r"] \arrow[d, "c"'] & HC + Id \arrow[r, "\rho C + Id"] \arrow[d, "Hc+Id"] \arrow[rr, bend left, "b"] & CC + Id \arrow[r, "{[\mu^C, \eta^C]}"] \arrow[d, "c*c+Id"] & C \arrow[d, "c"] \\
T \arrow[r, "{[\tau,\eta]^{-1}}"] & HT + Id \arrow[r, "\kappa T + Id"'] \arrow[rr, bend right, "{[\tau,\eta]}"'] & TT + Id \arrow[r, "{[\mu,\eta]}"'] & T
\end{tikzcd}$$

Indeed, the right-hand square commutes since $c : C \to T$ is a monad morphism, the left-hand one does because c is a coalgebra homomorphism for $\mathcal{H}$ (see (21)), and the middle square follows from fact that by Remark 2.25 c preserves the pointing, i.e., $c \cdot \rho = \tau \cdot H\eta$. Finally, the lower part follows from (11):

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

So the outside of the diagram commutes:

$$c \cdot b \cdot r = c,$$

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

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

$$\begin{tikzcd}
HC \arrow[r, equal] \arrow[d, "\mathsf{inl}"'] & HC \arrow[d, "\mathsf{inl}"] \\
HC + Id \arrow[r, "r \cdot b"'] & HC + Id \\
Id \arrow[r, equal] \arrow[u, "\mathsf{inr}"] & Id \arrow[u, "\mathsf{inr}"']
\end{tikzcd}$$

For the commutativity of the lower square we have since r is a monad morphism and the unit of the monad in the codomain is, by Lemma 2.6, inr that

$$r \cdot b \cdot \mathsf{inr} = r \cdot \eta^C = \mathsf{inr}\,.$$

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

$$\begin{tikzcd}
HC \arrow[r, "\tilde{\rho}C"] \arrow[rr, bend left, "\rho C"] \arrow[d, "H\eta^{\widetilde{C}}C"] \arrow[dd, bend right, "H\eta^{C}C"'] \arrow[dr, phantom, "\text{(i)}"] & \widetilde{C}C \arrow[r, "kC"] \arrow[d, "\tilde{r}C"] \arrow[dr, phantom, "\text{(ii)}"] & CC \arrow[r, "\mu^{C}"] \arrow[d, "rC"] \arrow[ddr, phantom, "\text{(iii)}"] & C \arrow[ddd, "r"] \\
H\widetilde{C}C \arrow[r, "\mathsf{inl}"] \arrow[d, "HkC"] \arrow[dr, phantom, "\text{(iv)}"] & (H\widetilde{C}+Id)C \arrow[r, "(Hk+Id)C"] & (HC+Id)C \arrow[dd, "(HC+Id)r"] & \\
HCC \arrow[urr, "\mathsf{inl}"] \arrow[d, "HCr"] \arrow[dd, bend right, equal] \arrow[dr, phantom, "\text{(v)}"] & & & \\
HC(HC+Id) \arrow[rr, "\mathsf{inl}"'] \arrow[d, "HCb"] \arrow[drr, phantom, "\text{(vi)}"] & & (HC+Id)(HC+Id) \arrow[r, "\widetilde{\mu}"'] & HC+Id \\
HCC \arrow[rrr, "H\mu^{C}"'] & & & HC \arrow[u, "\mathsf{inl}"']
\end{tikzcd}$$

Here $\widetilde{\mu}$ denotes the monad multiplication (12) of Lemma 2.6, where $S = C$ and $\sigma = \rho$. Indeed, all inner parts commute: the two left-hand parts commute since $k \cdot \eta^{\widetilde{C}} = \eta^{C}$ and $b \cdot r = id$, for part (i) recall that the coalgebra structure $\tilde{\rho}$ is a morphism in $H/\mathsf{Mon}(\mathscr{A})$, part (ii) commutes since k is a coalgebra homomorphism for $\mathcal{H}$, for (iii) use that r is a monad morphism, (iv) and (v) are trivial, and part (vi) commutes by (12). 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 $r \cdot \mu^{C} \cdot \rho C = \mathsf{inl} : HC \to HC + Id$, and this completes the proof. □

4 Context-free trees

We now return to the original concept of a context-free (or algebraic) Σ-tree on a given signature Σ, as studied by Bruno Courcelle, see the 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 context-free Σ-trees of Definition 1.1.

Observation 4.1 Polynomial endofunctors are projective in $\mathsf{Fun}_f(\mathsf{Set})$. That is, for every epimorphism (which means a componentwise 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 $g = p \cdot f$:

$$\begin{tikzcd}
F \arrow[r, two heads, "p"] & G \\
H_\Sigma \arrow[u, "\exists f"] \arrow[ur, "\forall g"'] &
\end{tikzcd}$$

In case Σ consists of a single n-ary symbol, this follows from 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 Yoneda Lemma to each of them separately.

Theorem 4.2 *For every signature Σ we have:*

$$C^{H_\Sigma} = \textit{the monad of context-free } \Sigma\textit{-trees}$$

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 2.27.

Indeed, since $\widetilde{C}$ is the filtered colimit of EQ, see Corollary 2.23, and filtered colimits of finitary functors in $\mathsf{Mon}(\mathscr{A})$ (and thus also in $H/\mathsf{Mon}(\mathscr{A})$) 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

$$r^\sharp_X : SX \to \widetilde{C}X$$

where $s : (S, \sigma) \to \mathcal{H}(S, \sigma)$ ranges over all coalgebras in EQ and $s^\sharp : S \to \widetilde{C}$ is the colimit cocone.

Since EQ is a closure of EQ_0 under coequalizers, every object of EQ is a quotient of one in EQ_0. Thus, we have a guarded recursive program scheme

$$e : V \to F^{H+V} \tag{23}$$

and an epimorphic coalgebra homomorphism for $\mathcal{H}$:

$$\begin{tikzcd}
(F^{H+V}, \widehat{\kappa} \cdot \mathsf{inl}) \arrow[r] \arrow[d, "q"'] & \mathcal{H}(F^{H+V}, \widehat{\kappa} \cdot \mathsf{inl}) \arrow[d, "\mathcal{H}q"] \\
(S, \sigma) \arrow[r, "s"] & \mathcal{H}(S, \sigma)
\end{tikzcd}$$

Since V is a finitely presentable functor, there exists by Example 2.1(ii) 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). Due to the projectivity of H_Φ we obtain a natural transformation f_0 making the diagram

$$\begin{tikzcd}
H_\Phi \arrow[r, "f_0"] \arrow[d, two heads, "p"'] & HF^{H+H_\Phi} + Id \arrow[dd, two heads, "H\widetilde{p}+Id"] \\
V \arrow[d, "\widehat{\kappa}\cdot\mathsf{inr}"'] \arrow[dr, "e_0"] & \\
F^{H+V} \arrow[r, "\overline{[\psi, e_0]}"'] & HF^{H+V} + Id
\end{tikzcd}$$

commutative (see Observation 4.1.) 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 2.13, the above monad morphism $\widetilde{p}$ is a coalgebra homomorphism.

We conclude that the triangles for $f^\dagger$ (see Theorem 2.16) and $f^\ddagger$ (see Theorem 2.27)

$$\begin{tikzcd}
H_\Phi \arrow[r, "\mathsf{inr}"] \arrow[rrddd, dashed, "f^\ddagger"] \arrow[rrdddd, "f^\dagger"'] & H + H_\Phi \arrow[r, "\widehat{\kappa}"] & F^{H+H_\Phi} \arrow[d, "\widetilde{p}"] \\
 & & F^{H+V} \arrow[d, "q"] \\
 & & S \arrow[d, "s^\sharp"] \\
 & & C \arrow[d, "c"] \\
 & & T^H
\end{tikzcd}$$

commute: recall from (20) that the coalgebra homomorphism f^* fulfils

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

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

(2) We will verify that $c_X : CX \hookrightarrow TX$ consists precisely of the context-free Σ-trees on 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 $e^\dagger_X = c_X \cdot e^\ddagger_X$, 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 rps $e : H_\Phi \to F^{H+H_\Phi}$. □

5 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 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 the present paper we also exhibited a general construction: for every finitary endofunctor H we provided a context-free monad C^H based on solutions of recursive program schemes of type H. The existence and uniqueness of these solutions were derived from the corresponding more general solution theorem of Ghani et al [12]. In case H is actually a polynomial endofunctor of Set associated to a signature Σ, our monad coincides with the monad of context-free (= algebraic) trees of Bruno Courcelle [10]. However, whereas Courcelle proved that the context-free-tree monad is iterative, we were only able to prove that the general context-free monad is 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 [22]. Again, this was, for context-free Σ-trees, proved by Bruno Courcelle.

Finally, the rational monad R^H and the monad T^H are both characterized by universal properties; R^H is the free iterative monad and T^H the free completely iterative one. It remains to be seen whether C^H can be characterized by some universal property, too. Unfortunately, context-free trees cannot serve as a guiding example in this respect as no universal property of them is known.

Acknowledgement

We are grateful to the anonymous referees for their comments which helped improving the presentation of our results.

References

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

[2] P. Aczel, J. Adámek, and J. Velebil. A coalgebraic view of infinite trees and iteration. In *Proc. Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44 of *Electron. Notes Theor. Comput. Sci.*, pages 1–26, 2001.

[3] J. Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolin.*, 15:589–602, 1974.

[4] J. Adámek, S. Milius, and J. Velebil. Iterative algebras at work. *Math. Structures Comput. Sci.*, 16(6):1085–1131, 2006.

[5] J. Adámek, S. Milius, and J. Velebil. Semantics of higher-order recursion schemes. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *Proc. Coalgebraic and Algebraic Methods in Computer Science (CALCO'09)*, volume 5728 of *Lecture Notes Comput. Sci.*, pages 49–63. Springer, 2009.

[6] J. Adámek, S. Milius, and J. Velebil. Iterative reflections of monads. to appear in *Math. Structures in Comput. Sci.*, published online by Cambridge University Press, doi:10.1017/S0960129509990326, February 2010.

[7] J. Adámek, S. Milius, and J. Velebil. Some Remarks on Finitary and Iterative Monads. *Appl. Categ. Structures*, 11(6):521–541, 2003.

[8] J. Adámek and J. Rosický. *Locally presentable and accessible categories*. Cambridge University Press, 1994.

[9] M. Barr. Coequalizers and free triples. *Math. Z.*, 116:307–322, 1970.

[10] B. Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.

[11] C. C. Elgot. Monadic computation and iterative algebraic theories. In H. E. Rose and J. C. Sheperdson, editors, *Logic Colloquium '73*, Amsterdam, 1975. North-Holland Publishers.

[12] N. Ghani, C. Lüth, and F. D. Marchi. Solving algebraic equations using coalgebra. *Theor. Inform. Appl.*, 37:301–314, 2003.

[13] N. Ghani, C. Lüth, and F. D. Marchi. Monads of coalgebras: rational terms and term graphs. *Math. Structures Comput. Sci.*, 15(3):433–451, 2005.

[14] N. Ghani, C. Lüth, F. D. Marchi, and A. J. Power. Algebras, coalgebras, monads and comonads. In *Proc. Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44 of *Electron. Notes Theor. Comput. Sci.*, pages 128–145, 2001.

[15] N. Ghani, C. Lüth, F. D. Marchi, and A. J. Power. Dualizing initial algebras. *Math. Structures Comput. Sci.*, 13(2):349–370, 2003.

[16] N. Ghani and T. Uustalu. Coproducts of ideal monads. *Theor. Inform. Appl.*, 38(4):321–342, 2004.

[17] S. Ginali. Regular trees and the free iterative theory. *J. Comput. System Sci.*, 18:228–242, 1979.

[18] I. Guessarian. *Algebraic Semantics*, volume 99 of *Lecture Notes in Comput. Sci.* Springer, 1981.

[19] S. Lack. On the monadicity of finitary monads. *J. Pure Appl. Algebra*, 140:65–73, 1999.

[20] J. Lambek. A fixpoint theorem for complete categories. *Math. Z.*, 103:151–161, 1968.

[21] S. Milius. Completely iterative algebras and completely iterative monads. *Inform. and Comput.*, 196:1–41, 2005.

[22] S. Milius and L. S. Moss. The category theoretic solution of recursive program schemes. *Theoret. Comput. Sci.*, 366:3–59, 2006.

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