

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

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Recursive Program Schemes and Context-Free Monads

Jiˇr´ı Ad´amek[a](#_bookmark0) Stefan Milius[a](#_bookmark0) Jiˇr´ı Velebil[b](#_bookmark1)*,*[1](#_bookmark2)

a *Institut fu¨r Theoretische Informatik, Technische Universita¨t 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 *CH* “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

# Introduction

The aim of the current paper is to introduce, for a finitary endofunctor *H* of a “reasonable” category, the context-free monad *CH* of *H* characterizing solutions of recursive program schemes of type *H*. This is analogous to our previous construction of the rational monad *RH* characterizing solutions of first-order recursive equations of type *H*, see [[4](#_bookmark48)]. In case of a polynomial functor *H* = *H*Σ on Set the monad *RH* is given by all rational Σ-trees, i. e., Σ-trees having (up to isomorphism) only a finite set of subtrees, see [[17](#_bookmark61)]. In contrast, *CH* is given by the algebraic trees investigated in the pioneering paper of Bruno Courcelle [[10](#_bookmark53)]. We call these trees *t context-free* since in [[10](#_bookmark53)] 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

*ϕ*1*,..., ϕk* of given arities *n*1*,..., nk* recursively, using given operations represented

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

1571-0661© 2010 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2010.07.011

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

*ϕ*(*x*) = *f* (*x, ϕ*(*gx*)) (1)

is a recursive program scheme defining a unary operation *ϕ* from the givens in Σ = *{ 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](#_bookmark53),[18](#_bookmark62)]. 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

*f,,*

*sss ,,*

*xs f,*

*sss*

*s*

*gx*

*,,,*

*f*

*ssss*

(2)

*ggx*

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

Observe that if Φ = *{ ϕ*1*,..., ϕk }* denotes the signature of the newly defined operations and

*H*Φ*X* = *Xn*1 + *···* + *Xnk*

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 *FH* the free monad on *H*, thus *FH*Φ is the monad of finite Φ-trees. A recursive program scheme can be formalized as a natural transformation

*e* : *H*Φ *→ FH*Σ+*H*Φ *.*

In fact, *FH*Σ+*H*Φ is the monad of all finite (Σ + Φ)-trees. Since *Xni* is a functor representable by *ni*, a natural transformation from *Xni* into *FH*Σ+*H*Φ is, by Yoneda Lemma, precisely an element of *FH*Σ+*H*Φ (*ni*), i. e., a finite (Σ + Φ)-tree on *ni* vari- ables. Thus, to give a natural transformation *e* as above means precisely to give *k* equations, one for each operation symbol *ϕi* from Φ,

*ϕi*(*x*0*,..., xn−*1) = *ti* (*i* = 1*,..., k*) (3)

where *ti* is a (Σ + Φ)-term on *{ x*0*,..., xn−*1 *}*. This is the definition of a recursive program scheme used in [[10](#_bookmark53)].

An uninterpreted solution of *e* : *H*Φ *→ FH*Σ+*Hϕ* is a *k*-tuple of Σ-trees *t† ,..., t†*

1 *k*

such that the above formal equations ([3](#_bookmark5)) become identities under the simultaneous second-order substitution [2](#_bookmark6) of *ti* for *fi*, for *i* = 1*,..., k*. For example, the tree *t†*(*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](#_bookmark53)] or [[22](#_bookmark66)] for a category-theoretic description.

from ([2](#_bookmark4)) satisfies the corresponding equality of trees

*t†*(*x*) = *g*(*x, t†*(*f x*))*.*

This concept of solutions was formalized in [[22](#_bookmark66)] by means of the free completely iterative monad *TH* on a functor *H*; in case *H* = *H*Σ this is the monad of all Σ-trees. We recall this in Section [2](#_bookmark8). The uninterpreted solution is a natural transformation *e†* : *H*Φ *→ TH*Σ and this leads us to the following reformulation (and renaming) of the concept of an algebraic tree of Courcelle [[10](#_bookmark53)]:

**Definition 1.1** A Σ-tree is called *context-free* if there exists a recursive program scheme ([3](#_bookmark5)) such that *t* = *t†* .

1

**Example 1.2** Every rational tree is context-free, and ([2](#_bookmark4)) shows a context-free tree that is not rational.

Courcelle proved that the monad *CH*Σ of all context-free Σ-trees as a submonad of *TH*Σ is iterative in the sense of Calvin Elgot [[11](#_bookmark54)]. Furthermore, context-free trees are closed under second-order substitution. The aim of the present paper is a construction of the context-free monad *CH* 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](#_bookmark66),[6](#_bookmark50),[16](#_bookmark60)]. However, at this moment we leave as open problems the proofs that *CH* 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](#_bookmark53)]. As we mentioned already, Ir`ene Guessarian [[18](#_bookmark62)] presents the classical algebraic se- mantics 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](#_bookmark67)] and also appears independently and almost at the same time in the work of Neil Ghani et al [[14](#_bookmark55)] (see also [[15](#_bookmark59)]) and Peter Aczel et al [[2](#_bookmark46)] (see also [[1](#_bookmark45)]). Ghani et al [[12](#_bookmark56)] were the first to present a semantics of uninter- preted 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](#_bookmark56)] were taken further in [[22](#_bookmark66)]; this fundamental study contains a comprehensive category-theoretic version of algebraic semantics in the coalgebraic setting: the pa- per 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](#_bookmark56),[22](#_bookmark66)]. Our construction of the context- free monad is new. It is inpired by the construction of the rational monad in [[4](#_bookmark48)], see also [[13](#_bookmark57)] for a more general construction.

# Construction of the context-free monad

Throughout the paper we assume that a finitary (i. e., filtered colimit preserving) endofunctor *H* of a category *A* is given, and that *H* preserves monomorphisms.

We assume that *A* is locally finitely presentable, coproduct injections

inl : *X → X* + *Y* and inr : *Y → X* + *Y*

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

## Example 2.1

* 1. Sets, posets and graphs form locally finitely presentable categories, and our as- sumptions about monomorphisms hold in these categories. Finite presentabil- ity of objects means precisely that they are finite.
  2. If *A* is locally finitely presentable, then so is Fun*f* (*A* ), the category of all finitary endofunctors and natural transformations. In case *A* = Set, the poly- nomial endofunctor

*H*Σ*X* = *Xn n* = arity of *σ* (4)

*σ∈*Σ

is a finitely presentable object of Fun*f* (Set) iff Σ is a finite set. This is easily seen using Yoneda Lemma. In fact, the finitely presentable objects of Fun*f* (Set) are precisely quotients *H*Σ*/∼* of the polynomial functors with Σ finite, where

*~* is a congruence on *H*Σ, see [[5](#_bookmark49)].

Notice that our assumptions concerning monomorphisms carry over to Fun*f* (*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 pre- sentable but where the assumptions on monomorphisms above need not hold:

1. The category

Mon*f* (*A* )

of all finitary monads on *A* and monad morphisms. This is a locally finitely presentable category. Indeed, as observed by Steve Lack [[19](#_bookmark63)], the forgetful functor

Mon*f* (*A* ) *→* Fun*f* (*A* )

is finitary and monadic, thus, the local finite presentability of Fun*f* (*A* ) implies that of Mon*f* (*A* ), see [[8](#_bookmark52)], 2.78. It follows that filtered colimits of finitary monads are formed object-wise on the level of *A* .

1. We will also make use of the fact that for every locally finitely presentable category *B* and object *B* the coslice category *B/B* of all morphisms with domain *B* is a locally finitely presentable category, see [[8](#_bookmark52)], 2.44.

**Free monad.** Recall from [[3](#_bookmark47)] that since *H* is a finitary endofunctor, free *H*-algebras *ϕX* : *H*(*FHX*) *→ FHX* exist for all objects *X* of *A* . Denote by *ηX* : *X → FHX* the universal arrow. As proved by M. Barr [[9](#_bookmark58)] the corresponding monad on *A*

^

*FH*

of free *H*-algebras is a free monad on *H*. It follows that *FH* is a finitary monad, and its unit

*η* : *Id → FH*

^

together with the natural transformation

*ϕ* : *HFH → FH*

given by the above algebra structures *ϕX* yield the universal arrow

*κ*^ = ( *H*  *Hη*b z*H*,*F H*  *ϕ* z*F H*,)*.*

The universal property states that for every monad *S* and every natural transfor- mation *f* : *H → S* there exists a unique monad morphism *f* : *FH → S* such that the triangle below commutes:

*H ¸ κ*b

*¸¸¸¸*

z*F*,*H*

(5)

*f ¸¸¸ f*

r,¸

*S*

Moreover, from [[3](#_bookmark47)] we have

*FH* = *HFH* + *Id* with injections *ϕ* and *η*. (6)

^

**Remark 2.3** The category Mon*f* (*A* ), being locally finitely presentable, has co- products. We use the notation *⊕*.

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

*FH*+*K* = *FH ⊕ FK.* (7)

We shall use the same notation *ϕ*, *η*^ and *κ*^ for different endofunctors than *H*,

e. g. *κ*^ : *H* + *K → FH*+*K*.

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

*THX → H* *THX* + *X.* (8)

By Lambek’s lemma [[20](#_bookmark64)], this morphism is invertible, and we denote the components of the inverse by

*τX* : *H* *THX* *→ THX* and *ηX* : *X → THX.*

respectively.

**Notation 2.4** *Since THX 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](#_bookmark45)], *T* is the underlying functor of a monad (*T, η, μ*) with the unit *η* : *Id → T* above. This monad is, moreover, the free completely iterative monad on *H*, see [[1](#_bookmark45),[21](#_bookmark65)]. The above natural transformation *τ* : *HT → T* yields the universal arrow

*κ* = ( *H Hη* z*H*,*T τ* z*T* ,) (9)

Moreover, in analogy to ([6](#_bookmark12)) above, we have

*T* = *HT* + *Id* with injections *τ* and *η*. (10)

Also recall from loc. cit. that the monad multiplication *μ* : *TT → T* is a homomor- phism of *H*-algebras (here we drop objects in the square below as all arrows are natural transformations):

*HTT*  *τT* z*T T*,

*Hμ μ*

, ,

(11)

*HT τ* z*T* ,

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

*H/*Mon(*A* )

the category of *H*-*pointed monads*, i. e., pairs (*S, σ*) where *S* is a monad on *A* and *σ* : *H → S* is a natural transformation. This is isomorphic to the coslice category of *FH* :

*H/*Mon(*A* ) *∼*= *FH/*Mon(*A* )*.*

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

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

*b* = [*μS · σS, ηS*] : *HS* + *Id → S.*

**Lemma 2.6 (Ghani et al [**[**13**](#_bookmark57)**])** *For every H-pointed monad* (*S, σ*) *the endofunc- tor HS*+*Id carries a canonical monad structure whose unit is the coproduct injection*

inr : *Id → HS* + *Id and whose multiplication is given by*

(*HS* + *Id* )(*HS* + *Id* )

*HS*(*HS* + *Id* )+ *HS* + *Id*

*HSb*+*HS*+*Id*

*HSS* + *HS* + *Id*

,

[*HμS,HS*]+*Id*

,

*HS* + *Id*

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

(12)

inl *· HηS* : *H → HS* + *Id.* (13) This defines an endofunctor H : *H/*Mon(*A* ) *→ H/*Mon(*A* ) on objects by

H(*S, σ*) = (*HS* + *Id,* inl *· HηS*)*,* see [[13](#_bookmark57)] or [[22](#_bookmark66)], Lemma 5.2 for details.

**Example 2.8** For every finitary endofunctor *V* we consider *FH*+*V* as an *H*-pointed

monad via

*H*  inl z*H*,+ *V*  *κ*b z*F H*,+*V*

And H(*FH*+*V* ) = *HFH*+*V* + *Id* is then an *H*-pointed monad via ([13](#_bookmark22)) which has the form

*ψ* = ( *H*  *Hη*b z*H*,*F H*+*V*  inl z*H*,*F H*+*V* + *Id* )*.* (14)

The proof of the following theorem is similar to the proof of Lemma 2.6 in [[13](#_bookmark57)]. The precise statement using the category *H/*Mon(*A* ) can be found in [[22](#_bookmark66)], Theo- rem 5.4.

**Theorem 2.9** *The terminal coalgebra for* H *is given by the H-pointed monad T,*

*H-pointed as in* ([9](#_bookmark16))*, with the coalgebra structure T −→∼* H*T from* ([8](#_bookmark14))*.*

**Definition 2.10** A *recursive program scheme* (or *rps* for short) of type *H* is a natural transformation

*e* : *V → FH*+*V*

from an endofunctor *V* which is a finitely presentable object of Fun*f* (*A* ) to the free monad on *H* + *V* . It is called *guarded* provided that it factorizes through the summand *HFH*+*V* + *Id* of the coproduct ([6](#_bookmark12)):

*FH*+*V* = (*H* + *V* )*FH*+*V* + *Id* = *HFH*+*V* + *VFH*+*V* + *Id,*

that is, we have a commutative triangle

*V ¸ ¸*

*e* z*F*,*H*+*V*

*¸ ¸ ¸*

*¸*

*e*0 *¸ ¸*

¸

[*ϕ·*inl *,η*b]

(15)

t

*HFH*+*V* + *Id*

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 *A* has monic coproduct injections.

**Example 2.11** In case of a polynomial endofunctor *H* = *H*Σ : Set *→* Set every recursive program scheme ([3](#_bookmark5)) yields a natural transformation *e* : *H*Φ *→ FH*Φ+*H*Σ , as explained in the introduction. This is a special case of Definition [2.10](#_bookmark25): in lieu of a general finitely presentable endofunctor *V* , which is a quotient of *H*Σ (cf. Exam- ple [2.1](#_bookmark9)(iv)), we just take *V* = *H*Σ.

The system ([3](#_bookmark5)) 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](#_bookmark3)), 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 → FH*+*V* in an *H*-pointed monad (*S, σ*) is meant a natural transformation *e†* : *V → S* such that the unique monad morphism extending [*σ, e†*] : *H* + *V → S* (see ([5](#_bookmark11))) makes the triangle below commutative:

*V e†* z*S* ,

˛,

*e* (16)

*†*

*F*

*H*,+*V* [*σ,e* ]

**Remark 2.13** (1) Every guarded recursive program scheme ([15](#_bookmark26)) turns *FH*+*V* into a coalgebra for H. Indeed, *e*0 : *V →* H(*FH*+*V* ) together with the pointing *ψ*, see ([14](#_bookmark23)), yield a natural transformation [*ψ, e*0] : *H* + *V →* H(*FH*+*V* ) which, by the universal property of the free monad *FH*+*V* , provides a unique monad morphism

[*ψ, e*0] : *FH*+*V →* H(*FH*+*V* ) (17)

It preserves the pointing: we have

[*ψ, e*0] *·* (*κ*^ *·* inl ) = [*ψ, e*0] *·* inl = *ψ.*

Thus, *FH*+*V* is a coalgebra.

1. Conversely, every coalgebra for H carried by *FH*+*V* , where *V* is a finitely pre- sentable endofunctor, stems from a guarded recursive program scheme: the coalge- bra structure *r* : *FH*+*V →* H(*FH*+*V* ) is uniquely determined by *r · κ* : *H* + *V →* H(*FH*+*V* ), and the left-hand component of *r · κ*^ being the pointing *ψ*, we see that

^

*r* is determined by *e*0 = *r · κ ·* inr : *V →* H(*FH*+*V* ) defining a (unique) recursive program scheme.

^

1. For the terminal coalgebra *T* for H, see Theorem [2.9](#_bookmark24), we thus obtain the unique coalgebra homomorphism

*e∗* : *FH*+*V → T.* (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](#_bookmark56)]. Let us recall from that paper that

* 1. an *H*-pointed monad is called *coalgebraic* if it is isomorphic to the monad

*HS* + *Id* of Lemma [2.6](#_bookmark20) via *b* : *HS* + *Id → S* in Notation [2.5](#_bookmark19)(ii),

* 1. examples of coalgebraic monads include *FH* , see ([6](#_bookmark12)), and *T* , see ([10](#_bookmark17)),
  2. *T* is the final coalgebraic monad; we denote by *uS* : *S → T* the unique mor- phism for a coalgebraic monad (*S, σ*),
  3. an *algebraic system* is given by a finitary monad *E*, a finitary coalgebraic monad (*S, σ*) and a monad morphism

*e* : *E → H*(*S ⊕ E*)+ *Id,*

* 1. a *solution* of *e* is a monad morphism *s* : *E → T* such that the square below commutes:

*E*  *s* z*T* ,

*e* [*τ,η*]*−*1

*H*( ,) + *Id*  z*H*,*T* +, *Id*

*S ⊕ E*

*H*([*uS,s*])+*Id*

**Theorem 2.15 (Ghani et al [**[**12**](#_bookmark56)**])** *Every algebraic system has a unique solution.*

This gives a solution theorem for recursive program schemes as follows: due to ([7](#_bookmark13)) we have the morhism *e*0 : *V → H*(*FH ⊕ FV* ) + *Id* in ([15](#_bookmark26)) yielding an algebraic system via ([5](#_bookmark11)):

*e*0 : *FV → H*(*FH ⊕ FV* )+ *Id.* (19)

Indeed, take *E* = *FV* and *S* = *FH* . Thus, a unique solution *s* : *FV → T* exists.

**Theorem 2.16** *Every guarded recursive program scheme of type H has a unique solution e† in T. It can be computed from the unique coalgebra homomorphism e∗* : *FH*+*V → T by*

*e†* = ( *V*  inr z*H*,+ *V*  *κ*b z*F H*,+*V e∗* z*T* ),*.* (20)

Indeed, for the unique solution *s* : *FV → T* of the algebraic system *e*0 in ([19](#_bookmark31)) above we obtain a solution *e†* in the sense of Definitinon ([2.10](#_bookmark25)) by composing with *κ*^ : *V → FV* :

*e†* = ( *V*  *κ*b z*F*,*V*  *s* z*T* ,)*.*

The proof that ([16](#_bookmark27)) commutes is performed using some diagram chasing. A some- what subtle point is that for *uS* : *S → T* (see Remark [2.14](#_bookmark30)(iii)) we have the equality

[*uS, s*] = [*κ*^*, e†*] : *FH*+*V → T.*

Here the square brackets on the left refer to the coproduct of *FH* and *FV* in *H/*Mon(*A* ) and those on the right to *H* + *V* in Fun*f* (*A* ). The verification uses the universal property of the free monad on *H* + *V* and is not difficult. The fact that ([20](#_bookmark32)) holds follows from the same diagram.

To prove that *e†* is unique use the fact that for any solution *e†* in the sense of Definition [2.10](#_bookmark25) its extension *e†* : *FV · T* is a solution of the corresponding algebraic system *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.

1. A finitary monad *C* together with a monad morphism *c* : *C → T* is constructed by forming a colimit of coalgebras for the endofunctor H obtained from all recursive program schemes.

˜ ˜ ˜

1. The (strong epi, mono)-factorization (cf. Proposition [2.19](#_bookmark35) below) of *c* is formed to obtain the desired submonad:

˜

e*c*

˜

*C ¸¸¸¸¸*

*¸¸¸¸¸*

*k ¸*

*...*z˛*T* ,,

*......*

*. c*

*¸* *C* ˛*.* ,

Unfortunately, Mon(*A* ) need not have such factorizations in general. We there- fore need to work in the category

Monacc(*A* )

of all monads on *A* that are *accessible*, that is, the underlying functors pre- serve, 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*](#_bookmark15)*) is accessible.*

**Proof.** It is proved in Proposition 5.16 of [[4](#_bookmark48)] that *TZ* can be constructed as the colimit of the diagram of all coalgebras for *H*(*−*)+ *Z* carried by all countably pre- sentable objects. Thus, *T* coincides with the *ℵ*1-accessible monad *Rℵ*1 of loc. cit.

**Proposition 2.19** *The category* Mon(*A* ) *has as monomorphisms precisely the monad morphisms with monic components. The subcategory* Monacc(*A* ) *has (strong epi, mono)-factorizations and is closed in* Mon(*A* ) *under strong epimorphisms and mono- morphisms.*

**Proof.** (1) The category Fun(*A* ) of all endofunctors on *A* has a generator formed by all accessible functors. In fact, let *u, v* : *K → L* be distinct natural transforma- tions. Then *uA /*= *vA* for some object *A*. Since *A* is locally finitely presentable, *A* is *λ*-presentable for some *λ*, see [[8](#_bookmark52)]. Thus, *A* lies in the small full subcategory *E* : *Aλ ‹→ A* representing all *λ*-presentable objects. The functor *K* has a *λ*- accessible coreflection *c* : *K' → K* obtained as the left Kan extension of *K · E* along

*E*. Since *A ∈ Aλ* implies that *cA* is an isomorphism, we conclude that *u · c /*= *v · c*, as desired.

* 1. The first statement of our proposition follows from the fact that every monomorphism *m* : *P → Q* in Mon(*A* ) is monomorphic in Fun(*A* ). By item (1), we only need to consider *u, v* : *K → P* with *m · u* = *m · v* where *K* is *λ*-accessible. Then free *K*-algebras exist, see [[3](#_bookmark47)]. Therefore a free monad *FK* exists, cf. [[9](#_bookmark58)]. The corresponding monad morphisms *u, v* : *FK → P* (cf. ([5](#_bookmark11))) fulfil *m · u* = *m · v*. This implies *u* = *v* since *m* is monic as a monad morphism. Thus, *u* = *u · κ* = *v · κ* = *v* as desired.

^  ^

* 1. The category Mon*λ*(*A* ) of all *λ*-accessible monads is closed under monomor- phisms in Mon(*A* ) since (by the same argument as in item (2)) monomorphisms in Mon*λ*(*A* ) are precisely the morphisms that are collectively monic. And it is closed under strong epimorphisms in Mon(*A* ) since this subcategory is coreflective; indeed, all left adjoints preserve strong epimorphisms. For *λ* = *ℵ*0 this was proved in [[7](#_bookmark51)], and for general *λ* the proof is (easy and) completely analogous.
  2. The category Mon*λ*(*A* ) is locally *λ*-presentable and therefore (strong epi,mono)- factorizations exist, see [[8](#_bookmark52)]. From item (3) it now follows that also Monacc(*A* ) has (strong epi, mono) factorizations and is closed under monos and strong epis in Mon(*A* ).

**Corollary 2.20** *The functor* H *preserves monomorphisms.*

Indeed, given a monomorphism *m* : (*S, σ*) *→* (*S', σ'*) in *H/*Mon(*A* ), then *m* is componentwise monic, thus, so is *Hm* (since *H* preserves monomorphisms), and so is also H*m* = *Hm* + *id* (since coproducts of monomorphisms are monic in *A* ).

**Construction 2.21** The *H*-pointed monad *CH* . For every guarded recursive pro- gram scheme ([15](#_bookmark26)) consider *FH*+*V* as a coalgebra for the functor H, see ([17](#_bookmark29)).

˜

We denote by

EQ0 *⊆* Coalg H

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

*C*˜*H* = colim EQ0 (in Coalg H)*.*

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

˜

*ρ*˜ : *H → C*˜*H* and *r*˜ : *C*˜*H →* H(*C*˜*H* )

respectively, together with a colimit cocone

*e* : *FH*+*V → C*˜*H* for all rps *e* : *V → FH*+*V* ,

formed by coalgebra homomorphisms for H preserving the pointing ([14](#_bookmark23)), i. e. with

*ρ*˜ = *e ·* (*κ*^ *·* inl ) for every *e*.

We see in the next lemma that EQ0 is a connected category. Since the forgetful functors

Coalg H *→ H/*Mon(*A* ) *→* Mon(*A* )

clearly preserve connected colimits, the above cocone *e* : *FH*+*V → T* is also a colimit cocone in Mon(*A* ).

**Lemma 2.22** EQ0 *is closed under ﬁnite coproducts in* Coalg H*.*

**Proof.** Consider two objects of EQ0 determined by

*e* : *V → HFH*+*V* + *Id* and *e'* : *V ' → HFH*+*V '* + *Id*

The coproduct injections *i* : *H* +*V → H* +*V* +*V '* and *i'* : *H* +*V ' → H* +*V* +*V '* yield corresponding monad morphisms *i* : *FH*+*V → FH*+*V* +*V '* and *i'* : *FH*+*V ' → FH*+*V* . Denote by

˜ ˜

*k* = (*HF*

*H*+*V* + *Id* )+ (*HF*

*H*+*V '* + *Id* ) [*H*e*i*+*Id,Hi*e*'*+*Id* ] *H*+*V* +*V '* + *Id*

the canonical morphism. We prove that the object *f* : *V* + *V ' → FH*+*V* +*V '* of EQ

z*H*,*F*

0

determined by

*f*0 = *k ·* (*e*0

+ *e'* ) : *V* + *V ' → HFH*+*V* +*V '* + *Id*

is the coproduct of the two given objects.

0

We know from Remark [2.13](#_bookmark28) that morphisms from the above object into an

H-coalgebra *X* = ((*S, s*)*, p*) are given by natural transformations

*t* : *V* + *V ' → S*

such that the extension [*s, t*] : *FH*+*V* +*V ' → S* of the transformation [*s, t*] : *H* + *V* +

*V ' → S* to a monad morphism fulfils

*p · r* = (*H*[*s, t*]+ *Id* ) *· f.*

We claim that this holds for *t* : *V* + *V ' → S* iff

1. the left-hand component *q* : *V → S* of *r* gives rise to a morphism of Coalg H from the object determined by *e*0 into *X*
2. and the right-hand component *q'* : *V ' → S* yields a morphism from the object

determined by *e'* into *X*.

0

For that observe first that the diagram

*F*  e*i* z*F*,

*'*  *i*e*' '*

*F*

*H*+*V*

*¸¸*

*H*+*V* +*V* ,r

*H*+*V*

*,*

*,,,*

*¸¸¸¸*

*¸¸¸*

*¸¸¸* [*s,t*]

*,,,,*

*,,,,* [*s,q'*]

*S*

[*s,q*]

*¸¸¸*t, *,*¸ *,,*r

commutes: indeed, all these morphisms are monad morphisms. The left-hand tri-

angle commutes since ˜*i* *· κ*^*H*+*V*

= *κ*^

*H*+*V* +*V '*

*· i*, therefore,

([*s, t*] *·* ˜*i*) *· κ*^ = [*s, t*] *· i* = [*s, q*] = [*s, q*] *· κ*^

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

*V* + *V ' f* z*H*,*F H*+*V* +*V '* + *Id*

*¸*,*¸*,

*¸ ¸*

*¸*

inl *¸ ¸*

*t , V*

*,*

*\_ e\_ \_*

*, ,*

z*H*,*F H*+*V* + *Id*

*\_*

*H*e*i*+*Id, , , ,* ,

*,*

*\_*

*, ,*

*H*[*s,t*]+*Id*

,*,* *,*

0

*, ,q ,*

*¸ ¸ ¸* *¸ ¸*

*H*[*s,q*]+*Id*

*¸ ¸ ¸*

*¸*s ,

*S* ¸r

*p* z*H*,*S* + *Id*

commutes iff [*s, q*] and [*s, q'*] are morphisms of Coalg H into *X*: in the diagram we indicated the left-hand component (commuting iff *p · q* = (*H*[*s, q*]+ *Id* ) *· e*0, that is, *q* is a homomorphism), analogously for the right-hand one.

## Corollary 2.23

*in* Coalg H*.*

*C*˜*H is a ﬁltered colimit of the closure* EQ *of* EQ0 *under coequalizers*

Indeed, since EQ0 is closed under finite coproducts, EQ is closed under finite colimits, thus, it is filtered. And colim EQ *∼*= colim EQ0.

**Definition 2.24** The context-free monad *CH* . Denote by

˜*c* : *C*˜*H → T*

the unique coalgebra homomorphism (see Theorem [2.9](#_bookmark24)) and define the *context-free monad* of *H* as the submonad *CH* of *T* obtained by the following (strong epi, mono)-factorization of ˜*c* in Mon(*A* ):

*H*

*C*

*..* ,

*k* ,

*c*

*.....* ,

*C*˜*H*

e*c* z*T* ,

**Remark 2.25** (i) Since *CH* is finitary and *T* accessible, see Lemma [2.18](#_bookmark34), we have the desired factorization by Proposition [2.19](#_bookmark35).

˜

1. The context-free monad is pointed: The pointing *ρ* : *H → CH* of *CH* yields the pointing

˜ ˜ ˜

*ρ* = *k · ρ* : *H → CH*

˜

of *CH* which *c* preserves (because *c* is a morphism of *H/*Mon(*A* )).

˜

1. Analogously to *T* we shall write *C* and *C* without the upper index *H* from now on.

˜

**Observation 2.26** The functor H preserves monomorphisms by Corollary [2.20](#_bookmark36), thus, *C* carries a canonical structure *r* of an H-coalgebra derived from the structure *r*˜ for *C*˜:

*k* zz*C*,,

*C*˜

*cccc*

*c*

*r*e *ccc*

, *r**cc* ,

*c*

(21)

H*C*˜

*ccc T*

H*k ccc*

*c*

,˛ r*c* ,

H*C*

H*T*

z, z,

H*c*

Indeed, recall that *c · k* = *c* is an 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 → FH*+*V has a unique solution in the context-free monad of H.*

**Proof.** We use *e‡* for solutions in *C* and *e†* for solutions in *T* throughout this proof. We are to prove that there exists a unique natural transformation *e‡* : *V → C* with *e‡* = [*ρ, e‡*] *· e*. Recall that the colimit injection *e* : *FH*+*V → C*˜ in Construction

[2.21](#_bookmark37) is a coalgebra homomorphism for H, hence, so is ˜*c · e*, which proves

*e∗* = ˜*c · e,*

see Theorem [2.16](#_bookmark33) (because *T* is a terminal coalgebra by Theorem [2.9](#_bookmark24)). Therefore, by ([20](#_bookmark32)) we have

*e†* = ˜*c · e · κ*^ *·* inr = *c · k · e · κ*^ *·* inr *.*

Thus for *e‡* = *k · e · κ*^ *·* inr we obtain

*e†* = *c · e‡.*

We conclude that *e‡* is the desired solution in *C*: in the following diagram

*V*J˜

*e*

*e†*

*‡ c*

*e* z,

*,,,,* *C*,

*,,,,,,*

*,*

, *,*

[*ρ,e‡*]

`\,

¸

z*T* ,

,

*FH*+*V*

[*κ,e†*]

the outside commutes, see ([16](#_bookmark27)) with *σ* = *κ*, and the right-hand part does since *κ* = *c · ρ* (see Definition [2.24](#_bookmark39)). Consequently, the left-hand triangle commutes: recall from Definition [2.24](#_bookmark39) that *c* is a monomorphism.

The uniqueness follows from the same diagram: if the left-hand triangle com- mutes, so does the outside, and since *e†* is uniquely determined (see Theorem [2.16](#_bookmark33)), we conclude *e†* = *c · e‡*. Finally, use again that *c* is monic.

# 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](#_bookmark54)] 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 *η* : *Id → S* is a subfunctor *σ* : *S' ‹→ S* of *S* (thus, *S* = *S'* + *Id* ) and *μ* restricts to a natural transformation *μ'* : *S'S → S'*. For general categories “ideal” is not a property but a structure:

**Definition 3.1** ([[1](#_bookmark45)]) An *ideal monad* is a sixtuple (*S, η, μ, S', σ, μ'*) where (*S, η, μ*) is a monad,

*σ* : *S' → S* (“the ideal”)

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

*μ'* : *S'S → S'*

is a natural transformation restricting *μ* in the sense that

*μ · σS* = *σ · μ'*

## Example 3.2

* 1. The free monad *FH* is ideal: its ideal is *HFH* , see ([6](#_bookmark12)).
  2. The free completely iterative monad *T* is ideal: its ideal is *HT* , see ([10](#_bookmark17)).

**Remark 3.3** It is our goal to prove that the context-free monad (*C, ηC, μC*) is ideal. The H-coalgebra structure *r* : *C → HC* + *Id* , see Observation [2.26](#_bookmark41), is (analogously to the two examples *FH* and *T* above) invertible, as we prove below: its inverse is the morphism

*ρC*+*Id* [*μC ,ηC* ]

+ *Id* + *Id*

*b ≡ HC* z*CC*, z*C,*,

(22)

cf. Notation [2.5](#_bookmark19)(ii). From that we will derive that *C* is an ideal monad with the ideal

*b ·* inl : *HC → 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 · r* = *id* follows, since *c* is a monomorphism, from the com- mutativity of the following diagram (here *c ∗ c* denotes the parallel composition of natural transformations):

*C*  *r* z*H*,*C*

*b*

+J˜*Id*  *ρC*+*Id* z*C*,*C* + *Id*

[*μC ,ηC* ]

,

z*C*,

`\

*c*

, [*τ,η*]*−*1

*Hc*+*Id*

,

*c∗c*+*Id c*

, ,

*T* z*H*,*T* +,¸*Id κT* +*Id* z*T T*,+ *Id* [*μ,η*]

¸

[*τ,η*]

z*T*, ,

Indeed, the right-hand square commutes since *c* : *C → T* is a monad morphism, the left-hand one does because *c* is a coalgebra homomorphism for H (see ([21](#_bookmark42))), and the middle square follows from fact that by Remark [2.25](#_bookmark40) *c* preserves the pointing, i.e., *c · ρ* = *τ · Hη*. Finally, the lower part follows from ([11](#_bookmark18)):

*μ · τT · HηT* = *τ · Hμ · HηT* = *τ.*

So the outside of the diagram commutes:

*c · b · r* = *c,*

and since *c* is a monomorphism, we see that *b · r* = *id* .

* + 1. To prove that *r · b* = *id* we show that the diagram below commutes:

*HC*

inl

*HC*

inl

*HC* +, *Id*  z*H*,*C* +, *Id*

¸

inr

*r·b*

¸

inr

*Id Id*

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](#_bookmark20), inr that

*r · b ·* inr = *r · ηC* = inr *.*

Since *b ·* inl = *μC · ρC* = *μC ·* (*kC · ρ*˜*C*), the commutativity of the upper square boils

down to showing that the outside of the following diagram commutes:

*ρC*

J˜

*ρ*˜*C*

*kC*

z,

`z\,,

*μC* z,

*HC*

*C*e (i)

*C*e*C*

*r*˜*C*

*CC C*

* + - 1. *rC*

*HηC C*

*Hη C*

,

*HC*e*C*

inl ,

(*HC*+*Id* )*C*

z,e

(*Hk*+*Id* )*C* ,

(*H*¸*C*+*Id* )*C*

z,

*HkC*

(iv)

* + - 1. *r*

r,¸

*HCC*

*HCr*

inl

(v)

(*HC*+*Id* )*r*

, z,

, ,

*HC*(*HC*+*Id* )

*HCb*

inl

(vi)

(*HC*+*Id* )(*HC*+*Id* )

*μ*˜ z*HC*,+ *Id*¸

inl

, z,

*HCC HμC HC*

Here *μ* denotes the monad multiplication ([12](#_bookmark21)) of Lemma [2.6](#_bookmark20), where *S* = *C* and

˜

*σ* = *ρ*. Indeed, all inner parts commute: the two left-hand parts commute since

*k · ηC*e = *ηC* and *b · r* = *id* , for part (i) recall that the coalgebra structure *ρ*˜ is a morphism in *H/*Mon(*A* ), part (ii) commutes since *k* is a coalgebra homomorphism

for H, for (iii) use that *r* is a monad morphism, (iv) and (v) are trivial, and part

(vi) commutes by ([12](#_bookmark21)). The remaining upper part commutes since *k* preserves the *H*-pointing. Finally, using the monad law *μC · ηCC* = *id* , we get *r · μC · ρC* = inl : *HC → HC* + *Id* , and this completes the proof.

# 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 *CH*Σ of the polynomial endofunctor *H*Σ of Set is indeed precisely the submonad *CH*Σ *‹→ TH*Σ of the Σ-tree monad consisting of all context- free Σ-trees of Definition [1.1](#_bookmark7).

**Observation 4.1** Polynomial endofunctors are projective in Fun*f* (Set). That is, for every epimorphism (which means a componentwise surjective natural transfor- mation) *p* : *F → G* and every natural transformation *g* : *H*Σ *→ G* there exists a natural transformation *f* : *H*Σ *→ F* with *g* = *p · f* :

*F p* zz*G*,,

¸ ¸,

*∃f*

*∀g*

*H*Σ

In case Σ consists of a single *n*-ary symbol, this follows from Yoneda Lemma, since *H*Σ *∼*= Set(*n, −*): the natural transformation *g* corresponds to an element of *Gn*, and we find its inverse image (under *pn*) in *F n*, giving us *f* : *H*Σ *→ F* . If Σ has

more symbols, apply Yoneda Lemma to each of them separately.

**Theorem 4.2** *For every signature* Σ *we have:*

*CH*Σ = *the monad of context-free* Σ*-trees*

**Proof.** Throughout the proof we write *H* in lieu of *H*Σ and *C* in lieu of *CH*Σ .

1. We prove that every element of *CX* lies in the image of *e‡* for some guarded recursive program scheme

*e* : *H*Φ *→ FH*+*H*Φ

where *e‡* is the unique solution in *C*, see Theorem [2.27](#_bookmark43).

Indeed, since *C*˜ is the filtered colimit of EQ, see Corollary [2.23](#_bookmark38), and filtered

colimits of finitary functors in Mon(*A* ) (and thus also in *H/*Mon(*A* )) are computed

on the level of the underlying functors (in other words: filtered colimits are formed object-wise in *A* ), we have for every set *X* a colimit cocone

*rX* : *SX → C*˜*X*

where *s* : (*S, σ*) *→* H(*S, σ*) ranges over all coalgebras in EQ and *s* : *S → C* is the colimit cocone.

˜

Since EQ is a closure of EQ0 under coequalizers, every object of EQ is a quotient of one in EQ0. Thus, we have a guarded recursive program scheme

*e* : *V → FH*+*V* (23)

and an epimorphic coalgebra homomorphism for H:

(*FH*+*V , κ*^ *·* inl ) zH,(*F H*+*V , κ*^ *·* inl )

*q*

( ,

*S, σ*)

H*q*

*s* z,,

H(*S, σ*)

Since *V* is a finitely presentable functor, there exists by Example [2.1](#_bookmark9)([ii](#_bookmark10)) a finite signature Φ and an epimorphic natural transformation

*p* : *H*Φ *→ V.*

The free-monad functor takes *H* + *p* : *H* + *H*Φ *→ H* + *V* to a monad morphism *p* : *FH*+*H*Φ *→ FH*+*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

˜

*H*  *f*0 z*H*,*F H*+*H*Φ + *Id*

Φ

*p*

,

*V ¸¸¸¸¸¸*

*Hp*e+*Id*

*κ*b*·*inr

,

*¸¸¸e*0*¸*

*¸¸¸¸*s ,

*¸¸¸*

*FH*+*V*  z*H*,*F H*+*V* + *Id*

[*ψ,e*0]

commutative (see Observation [4.1](#_bookmark44).) Here *f*0 is the guard of a “classical” guarded recursive program scheme *f* : *H*Φ *→ FH*+*H*Φ and for the corresponding H-coalgebra on *FH*+*H*Φ , see Remark [2.13](#_bookmark28), the above monad morphism *p* is a coalgebra homo- morphism.

˜

We conclude that the triangles for *f†* (see Theorem [2.16](#_bookmark33)) and *f‡* (see Theo- rem [2.27](#_bookmark43))

*H*Φ*¸¸*

inr z*H*,+ *H*Φ *κ*b z*F*,*H*+*H*Φ

*¸¸¸¸¸¸*

*¸¸¸ ¸*

*¸¸¸¸ ¸ ¸*

*¸¸¸ ¸*

*¸*

*¸*

*¸ ¸f ‡*

*p*e

,

*FH*+*V*

*q*

*¸¸¸ ¸ ¸*

*¸*

*¸¸ ¸ ¸* ,

*f† ¸¸¸¸ ¸ ¸ S*

*¸¸¸¸ ¸ ¸ s*

*¸¸¸¸*

*¸¸¸*

*¸* ,

*C*

*¸¸¸¸ c*

r,¸

*TH*

commute: recall from ([20](#_bookmark32)) that the coalgebra homomorphism *f∗* fulfils

*f†* = *f∗ · κ*^ *·* inr *,*

and so we only need to notice that the vertical arrow, being a coalgebra homo- morphism, is equal to *f∗*. Since *c* is a monomorphism, the upper triangle also

*X*

commutes. Thus, every element in the image of *s*

*X*

lies in the image of *f‡*

for the

above recursive program scheme *f* .

1. We will verify that *cX* : *CX ‹→ TX* consists precisely of the context-free Σ-trees on *X*. Indeed, every context-free Σ-tree has the form

*t* = *e†* (*x*)

*X*

for some guarded recursive program scheme *e* : *H*Φ *→ FH*+*H*Φ and since *e†* =

*X*

*cX · e‡* , the tree *t* lies in *CX*.

*X*

Conversely every element of *CX* has, by item (1) above, the form *e‡* (*x*) for some guarded rps *e* : *H*Φ *→ FH*+*H*Φ .

*X*

# Conclusions and Open Problems

The aim of our paper was to construct for a finitary endofunctor *H* a monad ex- pressing 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](#_bookmark48)]: there we defined the rational monad *RH* based on solutions of recursive equations, we proved that *RH* is iterative (and, in particular, ideal) in the sense of Calvin Elgot, and we characterized *RH* as the free iterative monad on

*H*. From this we derived, in case of endofunctors of Set, that *RH* 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 *CH* 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](#_bookmark56)]. 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](#_bookmark53)]. 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 *CH* would be proved to be iterative, the intuition says that this is not enough: the next open problem is, then, whether *CH* is closed under second-order substitution in the sense of [[22](#_bookmark66)]. Again, this was, for context-free Σ-trees, proved by Bruno Courcelle.

Finally, the rational monad *RH* and the monad *TH* are both characterized by universal properties; *RH* is the free iterative monad and *TH* the free completely iterative one. It remains to be seen whether *CH* 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 improv- ing the presentation of our results.

# References

1. P. Aczel, J. Ad´amek, 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. Ad´amek, 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. Ad´amek. Free algebras and automata realizations in the language of categories.

*Comment. Math. Univ. Carolin.*, 15:589–602, 1974.

1. J. Ad´amek, S. Milius, and J. Velebil. Iterative algebras at work. *Math. Structures Comput. Sci.*, 16(6):1085–1131, 2006.
2. J. Ad´amek, 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.

1. J. Ad´amek, 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.
2. J. Ad´amek, S. Milius, and J. Velebil. Some Remarks on Finitary and Iterative Monads.

*Appl. Categ. Structures*, 11(6):521–541, 2003.

1. J. Ad´amek and J. Rosicky´. *Locally presentable and accessible categories*. Cambridge University Press, 1994.
2. M. Barr. Coequalizers and free triples. *Math. Z.*, 116:307–322, 1970.
3. B. Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.
4. 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.
5. N. Ghani, C. Lu¨th, and F. D. Marchi. Solving algebraic equations using coalgebra.

*Theor. Inform. Appl.*, 37:301–314, 2003.

1. N. Ghani, C. Lu¨th, and F. D. Marchi. Monads of coalgebras: rational terms and term graphs.

*Math. Structures Comput. Sci.*, 15(3):433–451, 2005.

1. N. Ghani, C. Lu¨th, 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.
2. N. Ghani, C. Lu¨th, F. D. Marchi, and A. J. Power. Dualizing initial algebras. *Math. Structures* *Comput. Sci.*, 13(2):349–370, 2003.
3. N. Ghani and T. Uustalu. Coproducts of ideal monads. *Theor. Inform. Appl.*, 38(4):321–342, 2004.
4. S. Ginali. Regular trees and the free iterative theory. *J. Comput. System Sci.*, 18:228–242, 1979.
5. I. Guessarian. *Algebraic Semantics*, volume 99 of *Lecture Notes in Comput. Sci.* Springer, 1981.
6. S. Lack. On the monadicity of finitary monads. *J. Pure Appl. Algebra*, 140:65–73, 1999.
7. J. Lambek. A fixpoint theorem for complete categories. *Math. Z.*, 103:151–161, 1968.
8. S. Milius. Completely iterative algebras and completely iterative monads. *Inform. and Comput.*, 196:1–41, 2005.
9. S. Milius and L. S. Moss. The category theoretic solution of recursive program schemes.

*Theoret. Comput. Sci.*, 366:3–59, 2006.

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