THE CATEGORY THEORETIC SOLUTION OF RECURSIVE PROGRAM SCHEMES

STEFAN MILIUS AND LAWRENCE S. MOSS

Abstract. This paper provides a general account of the notion of recursive program schemes, studying both uninterpreted and interpreted solutions. It can be regarded as the category-theoretic version of the classical area of algebraic semantics. The overall assumptions needed are small indeed: working only in categories with "enough final coalgebras" we show how to formulate, solve, and study recursive program schemes. Our general theory is algebraic and so avoids using ordered, or metric structures. Our work generalizes the previous approaches which do use this extra structure by isolating the key concepts needed to study substitution in infinite trees, including second-order substitution. As special cases of our interpreted solutions we obtain the usual denotational semantics using complete partial orders, and the one using complete metric spaces. Our theory also encompasses implicitly defined objects which are not usually taken to be related to recursive program schemes. For example, the classical Cantor two-thirds set falls out as an interpreted solution (in our sense) of a recursive program scheme.

Contents

Date: 16 Dezember 2005.
Key words and phrases. recursive program scheme, Elgot algebra, coalgebra, completely iterative monad, algebraic trees, second-order substitution .

1. Introduction

The theory of *recursive program schemes* is a topic at the heart of semantics. One takes a system of equations such as

$$\begin{array}{rcl} \varphi(x) & \approx & F(x, \varphi(Gx)) \\ \psi(x) & \approx & F(\varphi(Gx), GGx) \end{array} \tag{1.1}$$

where F and G are *given* functions and where φ and ψ are defined in terms of them by (1.1). The problems are: to give some sort of semantics to schemes, and to say what it means to *solve* a scheme. Actually, we should distinguish between *interpreted* schemes, and *uninterpreted* schemes.

An interpreted scheme is one which comes with an algebra with operations for all the given operation symbols. Here is the standard example in the subject. Let Σ be the signature of given operation symbols with a constant one, one unary symbol pred, a binary symbol $*$ and a ternary one ifzero. The interpretation we have in mind is the natural numbers where $\mathsf{ifzero}_{\mathbb{N}}(k, n, m)$ returns m if k is 0 and n otherwise, and all other operations are obvious. The signature Φ of the recursively defined operations consists just of one unary symbol f. Consider the recursive program scheme

$$f(n) \approx \mathsf{ifzero}(n, \mathsf{one}, f(\mathsf{pred}(n)) * n)) \,. \tag{1.2}$$

Then (1.2) is a recursive program scheme defining the factorial function.

This paper presents a generalization of the classical theory based on *Elgot algebras* and *final coalgebras*. The point in a nutshell is that knowing that the infinite trees are the final coalgebra of a functor on sets leads to a purely algebraic account of first-order substitution and (co-)recursion, as shown in [AAMV, Mo_1]. One does not need to assume any metric or order to study infinite trees: the finality principle is sufficient. In this paper we show that corecursion allows us to give an uninterpreted semantics to a scheme; i. e., we show how to solve a scheme in final coalgebras.

For our interpreted semantics we work with Elgot algebras, a simple and fundamental notion introduced in [AMV_3]. We show how to give an interpreted solution to recursive program schemes in arbitrary Elgot algebras. We believe that our results in this area generalize and extend the previous work on this topic. Furthermore, we claim that our abstract categorical approach allows a unified view of several well-known approaches to the semantics of implicit recursive function definitions. Our method for obtaining interpreted solutions easily specializes to the usual denotational semantics using complete partial orders. As a second application we show how to solve recursive program schemes in complete metric spaces. For example, there is a unique contracting $f : [0, 1] \longrightarrow [0, 1]$ such that

$$f(x) = \frac{1}{4}\left(x + f\left(\frac{1}{2}\sin x\right)\right). \tag{1.3}$$

Concerning sets of real numbers, recall that the Cantor set c is the unique non-empty closed $c \subseteq [0, 1]$ such that

$$c = \frac{1}{3}c \;\cup\; \left(\frac{2}{3} + \frac{1}{3}c\right), \tag{1.4}$$

where $\frac{1}{3}c$ denotes the set $\{\, \frac{1}{3}x \mid x \in c \,\}$ as usual.

Finally, our theory also encompasses examples of recursive program schemes and their solutions which cannot be treated with the classical theory; in this paper we present recursive definitions of operations satisfying equations like commutativity. Other examples are recursive program scheme solutions in non-wellfounded sets (solving $x = \{\, \{\, y \mid y \subseteq x \text{ finite} \,\} \,\}$). We will present the applications of our results to non-wellfounded sets in a future paper.

Our purpose in this paper is to isolate general principles which imply the existence and uniqueness results for uninterpreted solutions of systems such as (1.1) and interpreted schemes such as (1.2–1.4).

1.1. Several Semantics.

There are several semantics for recursive program schemes, and it is worth mentioning a bit about them, both to situate our work and also because we shall be interested in showing that one can recover different flavors of semantics from our more general approach.

1.1.1. *Operational Semantics.* This gives semantics to interpreted schemes only. Solutions are defined by rewriting. In our factorial example, the semantics of (1.2) would be as follows: to compute the solution $f^\dagger(n)$ for a natural number n start with the term $f(n)$, substitute it by its right-hand side in the scheme and evaluate this term as much as possible. If the evaluated right-hand side still contains f replace it again by its right-hand side and then evaluate the whole term, and so on. If after finitely many steps this process returns a natural number we have computed $f^\dagger(n)$; otherwise we declare $f^\dagger(n)$ to be undefined. (Of course there are important and subtle points to be considered here pertaining to issues like call by name and call by

value interpretations of function symbols, and also about the overall strategy of rewriting.) In the factorial example the described process always stops. But in general this may not be the case.

1.1.2. *Denotational Semantics.* Again this provides semantics for interpreted schemes only. The algebra that comes as the interpretation of the given functions is a complete partial order A with continuous operations for all given operation symbols. A system like (1.1) gives then rise to a continuous function R on a function space. In our factorial example from (1.2) one would consider the natural numbers as a flat cpo and R assigns to a continuous function f on that cpo the function

$$Rf(n) = \begin{cases} \bot & \text{if } n = \bot \\ 1 & \text{if } n = 0 \\ f(n-1) \cdot n & \text{else} \end{cases} \tag{1.5}$$

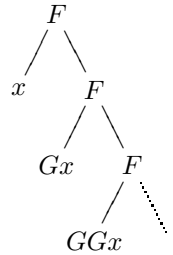
The semantics of the given scheme is then the least fixed point of R. More generally, denotational semantics provides a continuous operation φ_A on A for each newly defined operation symbol φ of a given recursive program scheme.

It is true but by no means obvious that the operational and denotational semantics agree in the appropriate sense. Thus there is a matter of semantic equivalence to be investigated. This was one of the primary starting points of the original theory of recursive program schemes, see [C, G, N]. In any case, there are two general themes that are raised by this matter of equivalence. First, one sees that the very definition of the denotational semantics requires order-theoretic methods. From our point of view, one must ask whether this is really necessary. One of the themes of work in coalgebra is that for some semantic problems, order-theoretic methods may be replaced by purely algebraic ones. This is a theme in some of the recent work in coalgebra that is relevant to our study. The reason is that coalgebra is often about semantics spaces for infinite behaviors of one type or another, and these are studied using universal properties (typically finality) instead of the extra structure coming from a cpo or complete metric space. The second general theme is what we might call *recovery of one semantics by another*. The first instance of this is the recovery of the operational semantics by the denotational semantics. One therefore feels that the denotational semantics is getting at something deeper. (At the same time, the very real subtle points of operational semantics are not easy to address with the methods of denotational semantics, so there is a loss as well.)

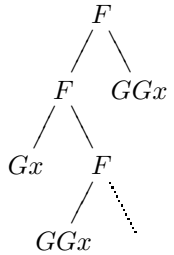
1.1.3. *Algebraic Semantics.* This considers *uninterpreted* recursive program schemes; i. e., where no interpretation is given. It is then an issue in this approach to make sure that one can recover the denotational and operational semantics inside the algebraic semantics. But first, one must say what a solution to (1.1) should be. Normally, one considers for a signature Σ and a set X of generators the set $T_\Sigma X$ of all finite and infinite Σ-trees over X, i. e., ordered and rooted trees labelled so that an inner node with n children, $n > 0$, is labelled by an n-ary operation symbol from Σ, and leaves are labelled by a constant symbol or a variable of X. Then one defines an appropriate notion of *second-order substitution* whereby Σ-trees may be substituted for operation symbols in Γ-trees for another signature Γ. In general, a recursive program scheme is given by two signatures Σ (of given operations) and Φ (of recursively defined operation), and a set of formal equations providing for each n-ary operation φ from Φ on the left-hand side a $(\Sigma + \Phi)$-tree over n syntactic variables on the right-hand side of the equation. A solution for such a recursive program scheme then assigns to each n-ary operation symbol φ from Φ a Σ-tree $\varphi^\dagger(x_1, \ldots, x_n)$ in n syntactic variables which is equal to the Σ-tree obtained by second-order substituting the solutions in the right-hand side of the formal equation defining φ.

As an example, consider the signature Σ with a binary operation symbol F and a unary symbol G. One might want to define new operations φ and ψ recursively as in (1.1) above. Notice that the system is *guarded*, or in *Greibach normal form*: the right-hand sides start with a symbol from Σ. One key opening result of algebraic semantics is that a unique solution exists for guarded systems among Σ-trees. For instance, for the above system (1.1) the solution consists of the Σ-trees

$\varphi^\dagger(x) =$



$\psi^\dagger(x) =$



(1.6)

This solution can in general be obtained as follows: the given scheme expands to a system of equations with recursion variables for each $(\Sigma + \Phi)$-tree t. This system gives to each recursion variable $\underline{t}$ where t is just a syntactic variable that variable itself, and otherwise the right-hand side of the system is given by replacing all appearances of symbols of Φ by their right-hand sides in the given scheme. For example, from (1.1) we obtain the equations

$$\begin{array}{rcl@{\qquad}rcl}
\underline{x} & \approx & x & \underline{\varphi(Gx)} & \approx & F(\underline{Gx}, \underline{\varphi(GGx)}) \\
\underline{\varphi(x)} & \approx & F(\underline{x}, \underline{\varphi(Gx)}) & \underline{F(x, \varphi(Gx))} & \approx & F(\underline{x}, \underline{F(Gx, \varphi(GGx))}) \\
\underline{\psi(x)} & \approx & F(\underline{\varphi(Gx)}, \underline{GGx}) & \underline{GGx} & \approx & G(\underline{Gx}) \\
\underline{\varphi(\psi(x))} & \approx & F(\underline{F(\varphi(Gx, GGx))}, \underline{\varphi(G(F(\varphi(Gx), GGx)))}) & & & \\
 & \vdots & & & &
\end{array} \tag{1.7}$$

and so on. Notice that each tree (here written as terms) on the right-hand side is a Σ-tree which is either just a syntactic variable or *flat*; i. e., one operation symbol from Σ with recursion variables. The solution of φ is now just the tree unfolding of the recursion variable $\underline{\varphi(x)}$ and similarly for ψ.

At this point, we have explained *how* we solve recursive program schemes in the algebraic semantics. But much has been omitted. For example, we gave no general account of *why* any solution method works, or even, why the above trees solve the given scheme.

The standard approach views infinite trees as either the completion of the finite trees, considered as a metric space, or else as the ideal completion of the set of finite trees. Second-order substitution is defined and studied using one of those methods, and using this one can say what it means to solve a recursive program scheme. We shall show in this paper that all of this can be done more generally and conceptually much nicer by considering Σ-trees as final coalgebras. More precisely, the algebra $T_\Sigma X$ of all Σ-trees over X is the final coalgebra for the functor $H_\Sigma(\_) + X$, where H_Σ is the polynomial endofunctors on sets associated to the signature Σ. It is the universal property of those final coalgebras, for any set X, which allows us to give a semantics to recursive program schemes.

1.1.4. *Category Theoretic Semantics.* As the title of our paper suggests, our goal is to propose a category-theoretic semantics of program schemes. The idea is to be even deeper than the algebraic semantics, and to therefore obtain results that are more general. Our theory is based on notions from category theory (monads, Eilenberg-Moore algebras) and coalgebra (finality, solution theorems, Elgot algebras). The overall assumptions are weak: there must be finite coproducts, and all functors we deal with must have "enough final coalgebras". More precisely, we work in a category $\mathcal{A}$ with finite coproducts and with functors $H : \mathcal{A} \longrightarrow \mathcal{A}$ such that for all objects X a final coalgebra TX of $H(\_) + X$ exists. We shall introduce and study recursive program schemes in this setting. In particular, we are able to prove the existence and uniqueness of interpreted and uninterpreted solutions to schemes. The price we pay for working in such a general setting is that our theory takes somewhat more effort to build. But this is not excessive, and perhaps our categorical proofs reveal more conceptual clarity than the classical ones.

Further, the issue of semantic recovery is quite interesting in our study. As we mentioned, we can recover the key aspects of the algebraic semantics in our approach. It follows that we can recover the other semantics as well. We shall interpret schemes in Elgot algebras. One of the key examples is the algebra $T_\Sigma X$ of all Σ-trees over X; and indeed, it appears that the fact that this is a free Elgot algebra on X implies all of the structural properties that are of interest when studying recursion. But in addition, there are many other interesting examples of Elgot algebras. As it happens, continuous algebras are Elgot algebras. We shall show that our general results on solving recursive program schemes in Elgot algebras directly specializes to *least fixed-point recursion* in continuous algebras. This yields the usual application of recursive program scheme solutions for the semantics of functional programs such as (1.2). Furthermore, algebras on complete metric spaces with contracting operations are Elgot algebras, and so our results specialize to the *unique fixed-point recursion* in completely metrized algebras provided by Banach's fixed-point theorem. This yields applications such as the ones in (1.3) and (1.4) above.

Related Work. The classical theory of recursive program schemes is compactly presented by Guessarian [G]. There one finds results on uninterpreted solutions of program schemes and interpreted ones in continuous algebras.

The first categorical accounts of infinite trees as monads of final coalgebras appear independently at almost at the same time in work of the second author [Mo_1], in Ghani et al. [GLMP_1, GLMP_2], and in Aczel et al. [AAV]. Furthermore, in [Mo_1] and in [AAV, AAMV] it is shown how solutions of formal recursive equations can be studied with coalgebraic methods. And in [AAMV] and [Mi_1] of the first author the

monad of final coalgebras is characterized as the free completely iterative monad, generalizing and extending results of Elgot et al. [E, EBT]. Hence, from [AAMV, Mi_1] it also follows how to generalize *second-order substitution* of infinite trees (see Courcelle [C]) to final coalgebras. The types of recursive equations studied in [Mo_1, AAMV] did not go as far as program schemes. It is thus an important test problem for coalgebra to see if work on solving systems of equations can extend to (un)interpreted recursive program schemes. We are pleased that this paper reports a success in this matter.

Ghani et al. [GLM] obtained a general solution theorem with the aim of providing a categorical treatment of uninterpreted program scheme solutions. Part of our proof for the solution theorem for uninterpreted schemes is inspired by their proof of the same fact. However, the notion of recursive program scheme in [GLM] is different from ours, and more importantly, our presentation of recursive program scheme solutions as fixed points with respect to to (generalized) second-order substitution as presented in [AAMV] is new here.

Complete metric spaces as a basis for the semantics of recursive program schemes have been studied for example by Arnold and Nivat [AN]. Bloom [Bl] studied interpreted solutions of recursive program schemes in so-called contraction theories. The semantics of recursively defined data types as fixed points of functors on the category of complete metric spaces has been investigated by Adámek and Reitermann [ARe] and by America and Rutten [ARu]. We build on this with our treatment of self-similar objects. These have also recently been studied in a categorical framework by Leinster, see [L_1, L_2, L_3]. The example in this paper uses standard results on complete metric spaces, see, e.g. [B]. We are not aware of any work on recursive program schemes that even mentions connections to examples like self-similar sets in mathematics, let alone develops the connection.

Finally, some of the results of this paper have appeared in our extended abstract [MM]. However, most of the technical details and all of the proofs were omitted. This full version explains our theory in much more detail and provides full proofs of all results.

Contents. The paper is structured as follows: Section 2 contains a brief summary of the definitions concerning monads. It also states the overall assumptions that we make in the rest of the paper. Section 3 presents the notions of *completely iterative algebra* and *Elgot algebra*, following [AMV_3]. Except for the Section 3.3, none of the results in this section is new here. But the paper as a whole would not make much sense to someone unfamiliar with completely iterative algebras and Elgot algebras. So we have included sketches of proofs in this section. The completely iterative monads of Section 4 are also not original here. But the properties of them developed in Section 5 are new. These are needed for the work on uninterpreted schemes (Section 6) and then interpreted schemes (Section 7). These are the heart of the paper.

2. Background: Iteratable Functors and Monads

This section contains most of the background which we need and also connects that background with recursive program schemes. Before reviewing monads in Section 2.2 we should mention the main base categories of interest in this paper, and our overall assumptions on those categories and endofunctors on them.

2.1. **Iteratable Functors.** Throughout this paper we assume that a category $\mathcal{A}$ with finite coproducts (having monomorphic injections) is given. In addition, all endofunctors H on $\mathcal{A}$ we consider are assumed to be *iteratable* [sic]: for each object X, the functor $H(\_) + X$ has a final coalgebra. We denote for an iteratable endofunctor H on $\mathcal{A}$ by

$$\mathcal{T}(H)X$$

the final coalgebra of $H(\_) + X$. Whenever confusion is unlikely we will drop the parenthetical (H) and simply write T for $\mathcal{T}(H)$. By the Lambek Lemma [L], the structure morphism of the final coalgebra is an isomorphism, and consequently, TX is a coproduct of HTX and X with injections

$$\begin{array}{lll} \eta^H_X & : X \longrightarrow TX & \text{“injection of variables”} \\ \tau^H_X & : HTX \longrightarrow TX & \text{“}TX\text{ is an }H\text{-algebra”} \end{array}$$

Again, the superscripts will be dropped if confusion is unlikely.

Having coproduct injections that are monomorphic is a mere technicality and could even be totally avoided, see [Mi_2], Sections 4 and 5. However, this assumption simplifies our presentation.

More serious is the assumption of iteratability. In one sense, this is a mild assumption: experience shows most endofunctors of interest are iteratable.

Example 2.1. We recall that ordinary signatures of function symbols Σ (as in universal algebra) give functors on Set. This is one of the central examples in this paper because it will allow us to recover the classical algebraic semantics from our category-theoretic one. A signature may be regarded as a functor

$\Sigma : \mathbb{N} \longrightarrow \mathsf{Set}$, where $\mathbb{N}$ is the discrete category with natural numbers as objects. For each n, write Σ_n for $\Sigma(n)$; this is the set of function symbols of arity n in Σ. There is no requirement that different numbers should give disjoint sets of function symbols of those arities. Given any signature Σ there is an associated polynomial endofunctor (henceforth called a *signature functor*)

$$H_\Sigma X = \Sigma_0 + \Sigma_1 \times X + \Sigma_2 \times X^2 + \cdots \tag{2.1}$$

on Set. When we need to refer to elements of $H_\Sigma X$, we shall use the notation $(f, \vec{x})$ for a generic element of $H_\Sigma X$; n is understood in this notation, $f \in \Sigma_n$, and $\vec{x}$ is an n-tuple of elements of X. Also observe that on morphisms $k : X \longrightarrow Y$ the action of H_Σ is given by $H_\Sigma k(f, \vec{x}) = (f, \overrightarrow{kx})$.

Signature functors H_Σ of Set are iteratable. In fact, consider the set

$$T_\Sigma X$$

of finite and infinite Σ-labelled trees with variables from the set X. That is, ordered and rooted trees labelled so that a node with n children, $n > 0$, is labelled by an operation symbol from Σ_n, and leaves are labelled by constant symbols or variables (elements of $X + \Sigma_0$).

This set $T_\Sigma X$ is the carrier of a final coalgebra for $H_\Sigma(\_) + X$. The coalgebra structure is the inverse of tree tupling paired with the obvious injection $\eta_X : X \longrightarrow T_\Sigma X$ which regards each variable as a one-point tree.

So at this point we have seen signature functors on Set as an examples of iteratable functors. Unfortunately, we must admit that iteratability is not a very nice notion with respect to closure properties of functors—for example, iteratable functors need not be closed under coproducts or composition. The main examples of base categories $\mathcal{A}$ in this paper are Set, CPO and CMS. In these categories there are stronger yet much nicer conditions that ensure iteratability.

Examples 2.2. (i) Accessible endofunctors of Set. Let λ be a regular cardinal. An endofunctor H of the category Set of sets and functions is called λ-*accessible* it it preserves λ-filtered colimits. It is shown in [AP], Proposition 5.2, that λ-accessible functors are precisely those endofunctors where for each set X any element $x \in HX$ lies in the image of $Hm : HY \longrightarrow HX$ for some subset $Y \hookrightarrow X$ of cardinality less than λ. As usual, we call an endofunctor H *finitary* if it is ω-accessible and we call H *accessible* if it is λ-accessible for some regular cardinal λ. Any accessible endofunctor is iteratable, see [Ba]. In particular, signature functors are finitary, whence iteratable (as we already know).

(ii) CPO is the category of ω-complete partial orders, i. e., partially ordered sets with joins of all increasing ω-chains (but not necessarily with a least element $\bot$). Morphisms of CPO are the continuous functions (preserving joins of all ω-chains). Hence the morphisms are monotone (preserve the order). Notice that coproducts in CPO are disjoint unions with elements of different summands incompatible. CPO is understood to be enriched in the obvious way. For given cpo's X and Y, the hom-set $\mathsf{CPO}(X, Y)$ is a cpo in the pointwise order. An endofunctor H on CPO is *locally continuous* if it preserves the extra structure just noted, i. e., each derived function $\mathsf{CPO}(X, Y) \longrightarrow \mathsf{CPO}(HX, HY)$ is continuous. Observe that not all locally continuous functors need be iteratable. For a counterexample consider the endofunctor assigning to a cpo X the powerset of the set of order components of X. This is a locally continuous endofunctor but it does not have a final coalgebra. However, any accessible endofunctor H on CPO has a final coalgebra, see [Ba], and moreover, H is iteratable.

(iii) Finally, CMS is the category of complete metric spaces with distances measured in the interval $[0, 1]$ together with *non-expanding* maps $f : X \longrightarrow Y$; i. e., $d_Y(fx, fy) \leq d_X(x, y)$ for all $x, y \in X$. A stronger condition is that f be ε-*contracting*: for some $\varepsilon < 1$ such that $d_Y(fx, fy) \leq \varepsilon \cdot d_X(x, y)$ for $x, y \in X$. Again, CMS is understood to be enriched. For complete metric spaces (X, d_X) and (Y, d_Y) the hom-set $\mathsf{CMS}(X, Y)$ is a complete metric space with the metric given by

$$d_{X,Y}(f, g) = \sup_{x \in X} d_Y(f(x), g(x)) \,.$$

Recall that a functor H on CMS is called ε-*contracting* if there exists a constant $\varepsilon < 1$ such that each derived function $\mathsf{CMS}(X, Y) \longrightarrow \mathsf{CMS}(HX, HY)$ is an ε-contracting map; i. e.,

$$d_{HX,HY}(Hf, Hg) \leq \varepsilon \cdot d_{X,Y}(f, g)$$

for all non-expanding maps $f, g : X \longrightarrow Y$. Contracting functors on CMS are iteratable, see [ARe].

Construction 2.3. Let H be an endofunctor of Set. We recall here that a final coalgebra T for H can (if it exists) be constructed as a limit of an (op-)chain. Let us define by transfinite induction the following chain indexed by all ordinal numbers:

$$\begin{array}{lllllll}
\text{Initial step:} & T_0 & = & 1, & t_{1,0} & \equiv & H1 \xrightarrow{!} 1, \\
\text{Isolated step:} & T_{i+1} & = & HT_i, & t_{i+1,j+1} & \equiv & HT_i \xrightarrow{Ht_{i,j}} HT_j \\
\text{Limit step:} & T_j & = & \lim\limits_{i<j} T_i & \text{with limit cone } t_{j,i} : T_j \longrightarrow T_i, & & i<j,
\end{array}$$

where the connecting map $t_{j+1,j}$ is uniquely determined by the commutativity of the squares

$$\begin{array}{ccc}
T_{i+1} = HT_i & \xleftarrow{Ht_{j,i}} & HT_j = T_{j+1} \\
{\scriptstyle t_{i+1,i}}\downarrow & & \vdots\,{\scriptstyle t_{j+1,j}} \\
T_i & \xleftarrow[t_{j,i}]{} & T_j
\end{array} \qquad i<j\,.$$

This chain it said to *converge* if $t_{i+1,i}$ is an isomorphism for some ordinal number i.

It has been proved by Adámek and Koubek [AK] that an endofunctor H has a final coalgebra iff the above chain converges; moreover, if i is an ordinal number such that $t_{i+1,i}$ is invertible, then $t_{i+1,i}^{-1} : T_i \longrightarrow HT_i$ is a final coalgebra for H. For many set endofunctors one can give a bound for the number of steps it will take until the above final coalgebra chain T_i converges. The following result has been established by Worrell [W].

Theorem 2.4. *For a λ-accessible endofunctor H of Set the final coalgebra chain T_i converges after $\lambda \cdot 2$ steps and $(T_{\lambda\cdot 2}, t_{\lambda\cdot 2+1,\lambda\cdot 2}^{-1})$ is a final coalgebra for H.*

In particular, for a finitary endofunctor a final coalgebra is obtained after $\omega + \omega$ steps. For some functors one can further improve on this bound. For endofunctors preserving limits of countable op-chains the final coalgebra chain converges after countably many steps so that $(T_\omega, t_{\omega+1,\omega}^{-1})$ is a final coalgebra. For example, each signature functor H_Σ of Set preserves limits of op-chains.

Examples 2.5. We mention additional examples of iteratable endofunctors H with their final coalgebras TX on the categories of interest.

(i) A functor $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ is finitary (i. e., it preserves filtered colimits) iff it is a quotient of some polynomial functor H_Σ, see [AT], III.4.3. The latter means that we have a natural transformation $\varepsilon : H_\Sigma \longrightarrow H$ with epimorphic components ε_X. These components are fully described by their kernel equivalence whose pairs can be presented in the form of so-called basic equations

$$\sigma(x_1, \ldots, x_n) = \rho(y_1, \ldots, y_m)$$

for $\sigma \in \Sigma_n$, $\rho \in \Sigma_m$ and $(\sigma, \vec{x}), (\rho, \vec{y}) \in H_\Sigma X$ for some set X including all x_i and y_j. Adámek and Milius [AM] have proved that the final coalgebra TX of $H(\_) + X$ is given by the quotient $T_\Sigma X/{\sim_X}$ where $\sim_X$ is the following congruence: for every Σ-tree t denote by $\partial_n t$ the finite tree obtained by cutting t at level n and labelling all leaves at that level by some symbol $\perp$ not from Σ. Then we have $s \sim_X t$ for two Σ-trees s and t iff for all $n < \omega$, $\partial_n s$ can be obtained from $\partial_n t$ by finitely many applications of the basic equations describing the kernel of ε_X. For example, the functor H which assigns to a set X the set $\{\, \{\, x, y \,\} \mid x, y \in X \,\}$ of unordered pairs of X is a quotient of $H_\Sigma X = X \times X$ expressing one binary operation b where ε_X is presented by commutativity of b; i. e., by the basic equation $b(x, y) = b(y, x)$. And TX is the coalgebra of all unordered binary trees with leaves labelled in the set X.

(ii) Consider the finite power set functor $\mathcal{P}_\mathrm{f} : \mathsf{Set} \longrightarrow \mathsf{Set}$. Under the Anti-Foundation Axiom (AFA), its final coalgebra is the set HF_1 of hereditarily finite sets; see [BM]. Analogously, the final coalgebra of $\mathcal{P}_\mathrm{f}(\_) + X$ is the set $HF_1(X)$ of hereditarily finite sets generated from the set X. Even without AFA, the final coalgebra of $\mathcal{P}_\mathrm{f}$ may be described as in Worrell [W]; it is the coalgebra formed by all strongly extensional trees; i. e., unordered trees so that for every node the subtrees defined by any two different children are not bisimilar. Analogously, the final coalgebra of $\mathcal{P}_\mathrm{f}(\_) + X$ is the coalgebra of all strongly extensional trees where some leaves have a label from the set X.

(iii) The (unbounded) power set functor $\mathcal{P} : \mathsf{Set} \longrightarrow \mathsf{Set}$ does not have a final coalgebra, whence it is not iteratable. However, moving to the category of classes and functions between them, the power set functor turns out to be iteratable, see e. g. [AMV$_3$]. Indeed, some of the machinery that comes from iteratable functors turns out to have a surprisingly set-theoretic interpretation when specialized to this setting; see [Mo$_3$].

(iv) In our applications, the key point is that certain Set endofunctors lift to (iteratable) endofunctors on CPO. And we need to know that those liftings are locally continuous. In fact, let H be an iteratable Set functor with a locally continuous lifting H' on CPO; i. e., a functor H' so that for the forgetful functor $U : \mathsf{CPO} \longrightarrow \mathsf{Set}$ we have a commutative square

$$\begin{array}{ccc} \mathsf{CPO} & \xrightarrow{H'} & \mathsf{CPO} \\ {\scriptstyle U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathsf{Set} & \xrightarrow[H]{} & \mathsf{Set} \end{array}$$

Then H' is iteratable, and moreover, the final coalgebra functor $\mathcal{T}(H')$ is a lifting of $\mathcal{T}(H)$:

$$\begin{array}{ccc} \mathsf{CPO} & \xrightarrow{\mathcal{T}(H')} & \mathsf{CPO} \\ {\scriptstyle U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathsf{Set} & \xrightarrow[\mathcal{T}(H)]{} & \mathsf{Set} \end{array} \tag{2.2}$$

To see this, first recall that for any set X the final coalgebra $\mathcal{T}(H)X$ is obtained from the final coalgebra chain T_i of $H(\_) + X$, see Construction 2.3. In fact, $\mathcal{T}(H)X$ is the coalgebra $(T_j, t_{j+1,j}^{-1})$ for the smallest ordinal number j for which $t_{j+1,j}$ is invertible. As CPO is a complete category we can define for any endofunctor H' of CPO a final coalgebra chain T_i' in precisely the same way as in 2.3. Since the forgetful functor U preserves limits, it follows that for a cpo X the final coalgebra chain of $H'(\_) + X$ has the T_i as underlying sets. However, in CPO the continuous map $t_{j+1,j}$ might not be invertible. But since the chain of underlying sets converges at index j we know that for all ordinal numbers k the connecting maps $t_{j+k,j} : T_{j+k} \longrightarrow T_j$ are monomorphisms of CPO. Moreover, all cpos T_{j+k} have (up to isomorphism) the same underlying set T_j and therefore the partial orders on the T_{j+k}, $k \geq 0$, form a decreasing chain of subsets of $T_j \times T_j$. This implies that the final coalgebra chain has to converge at some index $j + k$, $k \leq \mathrm{card}(T_j \times T_j)$. By standard arguments it follows that the cpo T_{j+k} is the final coalgebra of $H'(\_) + X$. Thus, we may choose $\mathcal{T}(H')X = T_j$ equipped with the cpo structure given by its subcpo T_{j+k}, whence the square (2.2) commutes as desired.

For example, every signature functor H_Σ has a locally continuous and iteratable lifting H'. This lift is the functor

$$H'X = \Sigma_0 + \Sigma_1 \times X + \Sigma_2 \times X^2 + \cdots$$

on CPO. Here each Σ_n is a discrete ordered set, $+$ is the coproduct of CPO (a lift of the coproduct of Set) and $\times$ the usual product. It should be noted that even if X has a least element, $H'X$ almost never has one. Finally, $\mathcal{T}(H')X$ is the Σ-tree algebra $T_\Sigma X$ with the order induced by the order of the cpo X—we describe this order in more detail later in Example 7.13(i).

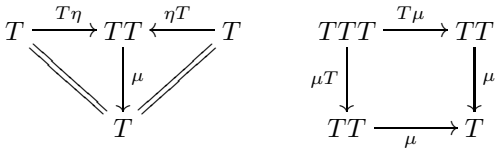
(v) Let $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ have a contracting lifting H' on CMS; i. e.,

$$\begin{array}{ccc} \mathsf{CMS} & \xrightarrow{H'} & \mathsf{CMS} \\ {\scriptstyle U}\downarrow & & \downarrow{\scriptstyle U} \\ \mathsf{Set} & \xrightarrow[H]{} & \mathsf{Set} \end{array}$$

for $U : \mathsf{CMS} \longrightarrow \mathsf{Set}$ the forgetful functor. Then H is iteratable and $U \cdot \mathcal{T}(H') = \mathcal{T}(H) \cdot U$. In fact, this follows from the results of [ARe] since U preserves limits. Any polynomial functor H on Set has a contracting lifting to CMS. For $HX = X^n$, define $H'(X, d) = (X, \frac{1}{2} d_{\max})$ (where $d_{\max}$ is the maximum metric) which is a contracting functor with $\varepsilon = \frac{1}{2}$. And coproducts of $\frac{1}{2}$-contracting liftings are $\frac{1}{2}$-contracting liftings of coproducts. The final coalgebra $\mathcal{T}(H')X$ is the Σ-tree algebra $T_\Sigma X$ equipped with a suitable complete metric. We will provide details of this metric later, see Remark 7.15.

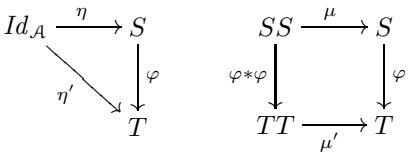
2.2. **Monads.** A *monad* on a category $\mathcal{A}$ is a triple (T, μ, η) consisting of a functor $T : \mathcal{A} \longrightarrow \mathcal{A}$, and natural transformations $\mu : TT \longrightarrow T$, and $\eta : Id \longrightarrow T$, satisfying the *unit laws* $\mu \cdot T\eta = \mu \cdot \eta T = id$, and the

associative law $\mu \cdot T\mu = \mu \cdot \mu T$:

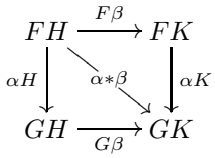


For a functor $H : \mathcal{A} \longrightarrow \mathcal{A}$ and a natural transformation $\alpha : F \longrightarrow G$ we use the usual notations $H\alpha$ and αH to denote the natural transformations with components $H(\alpha_X)$ and α_{HX}, respectively. Also it is customary to write just T for the monad in lieu of the triple, and we will follow this convention.

Let (S, η, μ) and (T, η', μ') be monads on the same category $\mathcal{A}$. A *morphism of monads* φ from S to T is a natural transformation $\varphi : S \longrightarrow T$ such that $\varphi \cdot \eta = \eta'$, and $\varphi \cdot \mu = \mu' \cdot (\varphi * \varphi)$:



This operation $*$ here is the *parallel composition* of natural transformations. In general, if $\alpha : F \longrightarrow G$ and $\beta : H \longrightarrow K$ are natural transformations, $\alpha * \beta : FH \longrightarrow GK$ is $\alpha K \cdot F\beta = G\beta \cdot \alpha H$.



We will denote by

$$\mathbf{Mon}(\mathcal{A})$$

the category of monads on $\mathcal{A}$ and their morphisms.

Example 2.6. Let H_Σ be a signature functor on Set. We already know how to define an object assignment T_Σ. In fact, T_Σ is a functor. We also have a natural transformation $\eta : Id \longrightarrow T_\Sigma$ which for each set X regards the elements of X as elements of $T_\Sigma X$. We additionally define a natural transformation $\mu_X : T_\Sigma(T_\Sigma X) \longrightarrow T_\Sigma X$, the operation which takes trees over trees over X into trees over X in the obvious way. In this way, we have a monad (T_Σ, μ, η) on Set.

Example 2.7. Let H be iteratable on a category $\mathcal{A}$. It has been shown in the previous work [AAMV, Mi$_2$] that the object assignment T (assigning to each object X the final coalgebra for $H(\_) + X$) gives rise to a monad on $\mathcal{A}$. This monad is characterized by a universal property—it is the free completely iterative monad on H. We will discuss this in detail later in Sections 3.4 and 4 below.

2.3. **Recursive Program Schemes.** We shall now explain how to capture recursive program schemes in a category-theoretic way. In order to do this we use a well-known adjunction between the category of signatures and the category of endofunctors of Set. For two categories $\mathcal{A}$ and $\mathcal{B}$ we denote by

$$[\mathcal{A}, \mathcal{B}]$$

the category of functors from $\mathcal{A}$ to $\mathcal{B}$.

Let Σ be a signature, i. e., $\Sigma : \mathbb{N} \longrightarrow \mathsf{Set}$ is a functor where $\mathbb{N}$ is regarded as a discrete category. Let $J : \mathbb{N} \longrightarrow \mathsf{Set}$ be the functor which maps a natural number n to the set $\{0, \ldots, n-1\}$. Recall that the functor $(\_) \cdot J : [\mathsf{Set}, \mathsf{Set}] \longrightarrow [\mathbb{N}, \mathsf{Set}]$ of restriction to $\mathbb{N}$ has a left-adjoint $\mathrm{Lan}_J(\_)$, i. e. the functor assigning to a signature its left Kan extension along J. Since $\mathbb{N}$ is a discrete category, the usual coend formula for computing left Kan extensions, see e. g. [ML], Theorem X.4.1, specializes to the coproduct in (2.1) above. That is, $\mathrm{Lan}_J(\Sigma)$ is the signature functor H_Σ. By virtue of the adjunction $\mathrm{Lan}_J(\_) \dashv (\_) \cdot J$ there is for any signature Σ and any endofunctor G of Set a bijection between natural transformations $\Sigma \longrightarrow G \cdot J$ and natural transformation $H_\Sigma \longrightarrow G$, and this bijection is natural in Σ and G. In fact, for any natural transformation $\alpha : \Sigma \longrightarrow G \cdot J$, i. e., a family of maps $\alpha_n : \Sigma_n \longrightarrow G\{0, \ldots, n-1\}$, we obtain a natural transformation $\beta : H_\Sigma \longrightarrow G$ as follows. The component β_X maps an element $(f, \vec{x})$ of $H_\Sigma X$ to $Gs(\alpha_n(f))$, where we consider the n-tuple $\vec{x}$ as a function $s : Jn \longrightarrow X$. Conversely, given $\beta : H_\Sigma \longrightarrow G$ define α by $\alpha_n(f) = \beta_{Jn}(f, i_n)$, where i_n is the n-tuple $(0, \ldots, n-1)$. It is easy to see that the two constructions yield natural transformations, are mutually inverse and natural in Σ and G.

We shall now use the above bijective correspondence to express recursive program schemes as natural transformations. Suppose we have a signature Σ of given operation symbols. Let Φ be a signature of new operation symbols. Classically a *recursive program scheme* (or shortly, *RPS*) gives for each operation symbol $f \in \Phi_n$ a term t^f over $\Sigma + \Phi$ in n variables. We thus have a system of formal equations

$$f(x_1, \ldots, x_n) \approx t^f(x_1, \ldots, x_n), \qquad f \in \Phi_n, \qquad n \in \mathbb{N}\,. \tag{2.3}$$

Now observe that the names of the variables in (2.3) do not matter. More precisely, regarding Φ an a functor from $\mathbb{N}$ to Set, any RPS as in (2.3) gives rise to a natural transformation

$$\Phi \longrightarrow T_{\Sigma+\Phi} \cdot J\,. \tag{2.4}$$

The formulation in (2.4) insures that in each equation of an RPS such as 2.3, if the symbol on the left side is n-ary, then the variables that can appear on the right are the n elements of $\{\, 0, \ldots, n-1 \,\}$. Notice as well that our formulation extends the classical notion of RPS in the sense that by taking $T_{\Sigma+\Phi}$ we allow infinite trees on the right-hand sides. Furthermore, we will generalize this notion of RPS.

The natural transformation in (2.4) corresponds to a unique natural transformation

$$H_\Phi \longrightarrow T_{\Sigma+\Phi}\,. \tag{2.5}$$

as explained above. The point is that the formulation in (2.5) is more useful to us than the one in (2.4) because (2.5) involves a natural transformations between endofunctors on one and the same category.

Now notice that $T_{\Sigma+\Phi} = \mathcal{T}(H_\Sigma + H_\Phi)$, where H_Σ and H_Φ denote the signature functors. With this in mind, we can rewrite (2.5), and we see that recursive program schemes correspond to natural transformations of the following form:

$$H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)\,.$$

This explains the work we have done so far.

To summarize: we abstract away from signatures and sets and study the uninterpreted and the interpreted semantics of recursive program schemes considered as natural transformations of the form

$$V \longrightarrow \mathcal{T}(H + V),$$

where H, V and $H + V$ are iteratable endofunctors on the category $\mathcal{A}$. Now to say what a solution of such a recursive program scheme is we first need to have a notion of (generalized) second-order substitution, see [C] for the classical notion. It turns out that the universal property of the free completely iterative monads $\mathcal{T}(H)$ readily yields this desired generalization. And this is the reason we are interested in monads in this paper.

Example 2.8. Let Σ contain a unary operation symbol G and a binary one F. The signature Φ of recursively defined operations contains two unary symbols φ and ψ. Consider the recursive program scheme (1.1). The signature functor expressing the givens is $H_\Sigma = X + (X \times X)$ and the recursively defined operations Φ are expressed by $H_\Phi X = X + X$. Thus, the scheme (1.1) gives a natural transformation $H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$. Similarly, the RPS (1.2) defining the factorial function with the signature Σ of givens and the signature Φ containing only the unary operation symbol f gives rise to a natural transformation $H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$, where $H_\Sigma X = 1 + X + X \times X$ and $H_\Phi X = X$.

2.4. **Eilenberg-Moore Algebras.** Recall that if (F, G, η, ϵ) is an adjunction, we get the associated monad on the domain of F by taking $T = GF$, $\mu = G\epsilon F$, and η from the adjunction. We also need a converse of this result. Given a monad T on $\mathcal{A}$, the *Eilenberg-Moore* category $\mathcal{A}^T$ of T has as objects the (monadic) T*-algebras*: these are morphisms $\alpha : TA \longrightarrow A$ such that the diagrams below commute:

$$\begin{array}{ccc} A & \xrightarrow{\eta_A} & TA \\ & \searrow & \downarrow{\scriptstyle \alpha} \\ & & A \end{array} \qquad \text{and} \qquad \begin{array}{ccc} TTA & \xrightarrow{\mu_A} & TA \\ {\scriptstyle T\alpha}\downarrow & & \downarrow{\scriptstyle \alpha} \\ TA & \xrightarrow[\alpha]{} & A\,. \end{array}$$

A morphism from $\alpha : TA \longrightarrow A$ to $\beta : TB \longrightarrow B$ is a morphism $h : A \longrightarrow B$ in $\mathcal{A}$ such that the square

$$\begin{array}{ccc} TA & \xrightarrow{\alpha} & A \\ {\scriptstyle Th}\downarrow & & \downarrow{\scriptstyle h} \\ TB & \xrightarrow[\beta]{} & B \end{array}$$

commutes. We usually write T-algebras using the notation of pairs, as in (A, α).

The relation between this construction and monads is that for any monad T, there is an adjunction $(F^T, U^T, \eta, \epsilon)$ from $\mathcal{A}$ to $\mathcal{A}^T$ to which T is associated. Here F^T is the functor taking A to the free T-algebra $\mu_A : TTA \longrightarrow TA$; $U^T : \mathcal{A}^T \longrightarrow \mathcal{A}$ is the forgetful functor taking the T-algebra $\alpha : TA \longrightarrow A$ to its carrier A; and in the same notation, $\epsilon_{(A,\alpha)}$ is α itself, taken to be a morphism of T-algebras, see [ML], Section VI.2.

3. Completely Iterative Algebras and Complete Elgot algebras

For interpreted solutions of recursive program schemes we need a suitable notion of algebras which can serve as interpretation of the givens. By a "suitable algebra" we mean, of course, one in which recursive program schemes have a *solution*. For example, for the recursive program scheme in (1.1) we are interested in those Σ-algebras, Σ a signature with a binary symbol F and a unary one G, in which we can obtain two new operations φ_A, ψ_A on A so that the formal equations of (1.1) become valid identities in A. In the classical theory one works with continuous algebras; i. e., algebras carried by a cpo such that all operations are continuous maps. Alternatively, one can work with algebras carried by a complete metric space such that all operations are contracting maps. In both of these approaches one imposes extra structure on the algebra in a way that makes it possible to obtain the semantics of a recursive definition as a join (or limit, respectively) of finite approximations.

When analyzing the two above types of algebras it turns out that they share a crucial feature that allows for the solution of recursive program schemes: these algebras induce an *evaluation* of all Σ-trees. More precisely, we consider a Σ-algebra A with a canonical map $T_\Sigma A \longrightarrow A$ providing for each Σ-tree over A its evaluation in A. It seems to us that in order to be able to obtain solutions of recursive program schemes in a Σ-algebra the minimal requirement is the existence of such an evaluation map turning A into an Eilenberg-Moore algebra of the monad T_Σ, see Example 2.6. More generally, we work here with *complete Elgot algebras* for an iteratable endofunctor H, which turn out to be precisely the Eilenberg-Moore algebras for the monad $\mathcal{T}(H)$, see [AMV$_3$]. An important subclass of all complete Elgot algebras are *completely iterative algebras* [Mi$_2$]. One of our main results (Theorem 7.3(ii)) states that recursive program schemes have unique solutions in completely iterative algebras.

Let us begin by explaining the notion of completely iterative algebra with an example. Let Σ be a signature, and let Y be any set. We think of Y as a set of parameters. The Σ-algebra $T_\Sigma Y$ of all (finite and infinite) Σ-trees over Y allows for the unique solution of *flat systems* of equations, i. e., systems of formal equations

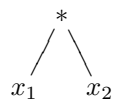
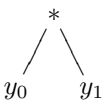
$$x_i \approx t_i \qquad i \in I, \tag{3.1}$$

with (recursion) variables from a set $X = \{ x_i \mid i \in I \}$, where either $t_i \in T_\Sigma Y$ is a Σ-tree with no recursion variables or else $t_i = \sigma(x_1, \ldots, x_n)$, $\sigma \in \Sigma_n$, $x_1, \ldots, x_n \in X$, is a flat Σ-tree.

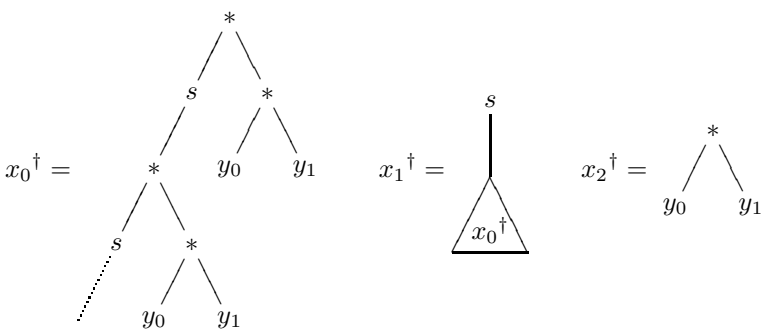
We have already begun in this paper to use the standard practice of using $\approx$ in a system to denote formal equations (recursive specifications of functions or other objects). We use $=$ to denote actual identity (see just below for an example). Flat systems have a unique *solution*: there exists a unique tuple ${x_i}^\dagger$, $i \in I$, of trees in $T_\Sigma Y$ such that the identities

$${x_i}^\dagger = t_i[\, x_i := {x_i}^\dagger \,]_{i \in I}$$

hold. For example, let Σ consist of a binary operation symbol $*$ and a unary one s. The following flat system of equations

$x_0 \approx$  $x_1 \approx$  $x_2 \approx$ 

with variables $X = \{ x_1, x_2, x_3 \}$ and parameters $Y = \{ y_0, y_1 \}$ has as its unique solution the following trees in $T_\Sigma Y$:



Observe that to give a flat system of equations is the same as to give a mapping $e : X \longrightarrow H_\Sigma X + T_\Sigma Y$ and a solution is the same as a mapping $e^\dagger : X \longrightarrow T_\Sigma Y$ such that the following square

$$\begin{array}{ccc} X & \xrightarrow{\quad e^\dagger \quad} & T_\Sigma Y \\ {\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [\tau, T_\Sigma Y]} \\ H_\Sigma X + T_\Sigma Y & \xrightarrow[H_\Sigma e^\dagger + T_\Sigma Y]{} & H_\Sigma T_\Sigma Y + T_\Sigma Y \end{array}$$

commutes, where τ denotes the tree-tupling map. (We adopt the convention of denoting in a commutative diagram the identity on an object by that object itself.)

3.1. **Completely Iterative Algebras.** The example above suggests the following definition, originally studied in [Mi$_2$].

Definition 3.1. Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be an endofunctor. By a *flat equation morphism* in an object A (*of parameters*) we mean a morphism

$$e : X \longrightarrow HX + A\,.$$

Let $a : HA \longrightarrow A$ be an H-algebra. We say that $s : X \longrightarrow A$ is a *solution* of e in A if the square below commutes:

$$\begin{array}{ccc} X & \xrightarrow{\quad s \quad} & A \\ {\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [\alpha, A]} \\ HX + A & \xrightarrow[Hs + A]{} & HA + A \end{array} \tag{3.2}$$

We call A a *completely iterative algebra* (or *cia*, for short) if every flat equation morphism in A has a unique solution.

Observe that we have no restriction on our objects of variables. (That is, in the case of Set, we do not require that the set of variables be finite.) Imposing this restriction weakens the notion to what [AMV$_1$] calls an *iterative algebra*. It will be essential in this paper to consider equation morphisms whose domain is not finite.

Examples 3.2.

(i) Let $\mathcal{P}_\mathrm{f}$ be the finite power set functor on Set, and assume the Anti-Foundation Axiom. Let HF_1 be the set of hereditarily finite sets. Let τ be the inclusion of $\mathcal{P}_\mathrm{f}(HF_1)$ into HF_1. This map τ turns HF_1 into a cia with respect to $\mathcal{P}_\mathrm{f}$.

(ii) Consider the subalgebra $HF_{1/2}$ of sets whose transitive closure is finite, then the *complete* iterativity is lost. Only finite systems can be solved in this setting. For more on this example and the last, see Section 18.1 of [BM].

(iii) Final coalgebras. In [Mi$_2$] it is proved that for a final H-coalgebra $\alpha : T \longrightarrow HT$ the inverse $\tau : HT \longrightarrow T$ of the structure map yields a cia. Analogously, for every object Y of $\mathcal{A}$ a final coalgebra TY of $H(\_) + Y$ yields a cia, see Theorem 3.12 below. This generalizes the first example and the examples $T_\Sigma Y$ of all Σ-trees over a set Y.

(iv) Let H be a contracting endofunctor on the category CMS of complete metric spaces, see Example 2.2(iii). Then any non-empty H-algebra (A, α) is completely iterative. In fact, given any flat equation morphism $e : X \longrightarrow HX + A$ in CMS, it is not difficult to prove that the assignment $s \longmapsto \alpha \cdot (Hs + A) \cdot e$ is a contracting function of $\mathsf{CMS}(X, A)$, see [AMV$_3$]. Then, by Banach's Fixed Point Theorem, there exists a unique fixed point of that contracting function, viz. a unique solution $e^\dagger$ of e. Notice that $e^\dagger$ is obtained as a limit of a Cauchy sequence. In fact, choose some element $a \in A$ and define the Cauchy sequence $(e^\dagger_n)_{n \in \mathbb{N}}$ in $\mathsf{CMS}(X, A)$ by recursion as follows: let $e^\dagger_0 = \mathsf{const}_a$, and given $e^\dagger_n$ define $e^\dagger_{n+1}$ by the commutativity of the square

$$\begin{array}{ccc} X & \xrightarrow{\quad e^\dagger_{n+1} \quad} & A \\ {\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [\alpha, A]} \\ HX + A & \xrightarrow[He^\dagger_n + A]{} & HA + A \end{array} \tag{3.3}$$

(v) Non-empty compact subsets form cias. Let (X,d) be a complete metric space. Consider the set $C(X)$ of all non-empty compact subspaces of X together with the so-called Hausdorff metric h; for two compact subsets A and B of X the distance $h(A,B)$ is the smallest number r such that B can be covered by open balls of radius r around each point of A, and vice versa, A can be covered by such open balls around each point of B. In symbols, $h(A,B) = \max\{\, d(A \to B), d(B \to A)\,\}$, where $d(A \to B) = \max_{a\in A} \min_{b\in B} d(a,b)$. It is well-known that $(C(X),h)$ forms a complete metric space; see, e.g. [B]. Furthermore, if $f_i : X \longrightarrow X$, $i = 1,\ldots,n$, are contractions of the space X with contraction factors c_i, $i = 1,\ldots,n$, then it is easy to show that the map

$$\alpha_X : C(X)^n \longrightarrow C(X) \qquad (A_i)_{i=1,\ldots,n} \longmapsto \bigcup_{i=1}^{n} f_i[A_i]$$

is a contraction with contraction factor $c = \max_i c_i$ (the product $C(X)^n$ is, of course, equipped with the maximum metric). In other words, given the f_i, we obtain on $C(X)$ the structure α_X of an H-algebra of the contracting endofunctor $H(X,d) = (X^n, c\cdot d_{\max})$. Thus, if there exists a non-empty compact subset of X, then $(C(X),\alpha_X)$ is a cia.

As an illustration we show that the Cantor "middle third" set s may be obtained via the cia structure on a certain space. Recall that s is the unique non-empty closed subset of the interval $I = [0,1]$ which satisfies $s = \frac{1}{3}s \cup (\frac{2}{3} + \frac{1}{3}s)$. (We write $\frac{1}{3}s$ to mean $\{\, \frac{x}{3} \mid x \in s\,\}$, of course.) So let (X,d) be the Euclidean interval $I = [0,1]$ and consider the $\frac{1}{3}$-contracting functions $f(x) = \frac{1}{3}x$ and $g(x) = \frac{1}{3}x + \frac{2}{3}$ on I. Then $\alpha_I : C(I)\times C(I) \longrightarrow C(I)$ with $\alpha_I(A,B) = f[A]\cup g[B]$ gives the structure of a cia on $C(I)$ of the functor $H(X,d) = (X\times X, \frac{1}{3}\cdot d_{\max})$, which is a lifting of the signature functor $H_\Sigma X = X\times X$ of Set expressing one binary operation symbol α. Now consider the formal equation

$$x \approx \alpha(x,x)\,.$$

It gives rise to a flat equation morphism $e : 1 \longrightarrow H1 + C(I)$ which maps the element of the trivial one point space 1 to the element of $H1 = 1$. The unique solution $e^\dagger : 1 \longrightarrow C(I)$ picks a non-empty closed set s satisfying $s = \alpha(s,s) = f[s]\cup g[s]$, i. e., $e^\dagger$ picks the Cantor set.

(vi) Continuing with our last point, for each non-empty closed $t \in C(I)$, there is a unique $s = s(t)$ with $s = \alpha(s,t)$. The argument is just as above. But the work we have done does *not* show that the map $t \longmapsto s(t)$ is continuous. For this, we would have to study a recursive program scheme $\varphi(x) \approx \alpha(\varphi(x),x)$ and solve this in $(C(I),\alpha_I)$ in the appropriate sense. Our work later in the paper does exactly this, and it follows that the solution to $\varphi(x) \approx \alpha(\varphi(x),x)$ in the given algebra is the *continuous* function $t \longmapsto s(t)$.

(vii) Suppose that $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ has a lifting to a contracting endofunctor H' on CMS, see Example 2.5(vi). Let $\alpha : HA \longrightarrow A$ be an H-algebra. If there exists a complete metric, say d, on A such that α is a nonexpanding map $H'(A,d) \longrightarrow (A,d)$, then A is a completely iterative algebra: to every equation morphism $e : X \longrightarrow HX+A$ assign the unique solution of $e : (X,d_0) \longrightarrow H'(X,d_0)+(A,d)$, where d_0 is the discrete metric on X; i. e., $d(x,x') = 1$ iff $x \neq x'$.

3.2. **Complete Elgot Algebras.** In many settings, one studies a fixed point operation on a structure like a complete partial order. And in such settings, one typically does not have *unique* fixed points. So completely iterative algebras are *not* the unifying concept capturing precisely what is needed to solve recursive program schemes. Instead, we shall need a weakening of the notion of a cia.

Remark 3.3. We will need two operations in the statement of Definition 3.4 below. The first operation formalizes the renaming of parameters in a flat equation morphism. More precisely, for a flat equation morphism $e : X \longrightarrow HX + A$ and a morphism $h : A \longrightarrow B$ we define

$$h \bullet e \equiv X \xrightarrow{\ e\ } HX + A \xrightarrow{\ HX+h\ } HX + B\,.$$

The second operation allows us to combine two flat equation morphisms where the parameters of the first are the variables of the second into one "simultaneous" flat equation morphism. More precisely, given two flat equation morphisms $e : X \longrightarrow HX + Y$ and $f : Y \longrightarrow HY + A$ we define

$$f \blacksquare e \equiv X + Y \xrightarrow{\ [e,\mathsf{inr}]\ } HX + Y \xrightarrow{\ HX+f\ } HX + HY + A \xrightarrow{\ \mathsf{can}+A\ } H(X+Y) + A\,.$$

Definition 3.4. A complete *Elgot algebra* is a triple $(A,a,(\_)^\dagger)$, where (A,a) is an H-algebra, and $(\_)^\dagger$ assigns to every flat equation morphism e with parameters in A a solution $e^\dagger : X \longrightarrow A$ such that the following two laws hold:

Functoriality. Solutions respect renaming of variables. Given two flat equation morphisms e and f with parameters in A and a morphism h of equations between them; i. e., the square

$$\begin{array}{ccc} X & \xrightarrow{e} & HX + A \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle Hh+A} \\ Y & \xrightarrow[f]{} & HY + A \end{array}$$

commutes. We then have

$$e^\dagger = f^\dagger \cdot h\,.$$

Compositionality. Simultaneous recursion may be performed sequentially. For all flat equation morphisms $e : X \longrightarrow HX + Y$ and $f : Y \longrightarrow HY + A$, the solution of the combined equation morphism $f \blacksquare e$ is obtained by first solving f and then solving e "plugging in" $f^\dagger$ for the parameters:

$$(f^\dagger \bullet e)^\dagger = (f \blacksquare e)^\dagger \cdot \mathsf{inl}\,.$$

Definition 3.5. A *homomorphism* h from an Elgot algebra $(A, a, (\_)^\dagger)$ to an Elgot algebra $(B, b, (\_)^\ddagger)$ (for the same functor H) is a morphism $h : A \longrightarrow B$ that preserves solutions, i. e., for every flat equation morphism $e : X \longrightarrow HX + A$ we have a commutative triangle

$$\begin{array}{ccc} & X & \\ {\scriptstyle e^\dagger}\swarrow & & \searrow{\scriptstyle (h\bullet e)^\ddagger} \\ A & \xrightarrow[h]{} & B\,. \end{array}$$

Proposition 3.6. [AMV$_3$] *Every homomorphism $h : (A, a, (\_)^\dagger) \longrightarrow (B, b, (\_)^\ddagger)$ of Elgot algebras is a homomorphism of H-algebras; i. e., the square*

$$\begin{array}{ccc} HA & \xrightarrow{a} & A \\ {\scriptstyle Hh}\downarrow & & \downarrow{\scriptstyle h} \\ HB & \xrightarrow[b]{} & B \end{array}$$

commutes. The converse is false in general. If, however, A and B are cias, then any H-algebra morphism is a homomorphism of Elgot algebras.

Proof. We sketch the argument of the first statement, omitting some of the details. First, consider the flat equation morphism

$$e_A \equiv HA + A \xrightarrow{H\mathsf{inr}+A} H(HA + A) + A\,.$$

Its solution is $e_A^\dagger = [a, A] : HA + A \longrightarrow A$ as one easily establishes using the commutativity of Diagram 3.2 for $e_A^\dagger$. Similarly, we have $e_B : HB + B \longrightarrow H(HB + B) + B$ with $e_B^\ddagger = [b, B]$. Now consider h as in our proposition. Then the equation $(h \bullet e_A)^\ddagger = h \cdot e_A^\dagger$ holds. Furthermore, $Hh + h : HA + A \longrightarrow HB + B$ is a morphism of equations from $h \bullet e_A$ to e_B, thus Functoriality yields $(h \bullet e_A)^\ddagger = e_B^\ddagger \cdot (Hh + h)$. Now one readily performs the computation

$$[h \cdot a, h] = h \cdot e_A^\dagger = (h \bullet e_A)^\ddagger = e_B^\ddagger \cdot (Hh + h) = [b \cdot Hh, h]\,.$$

The desired equation $h \cdot a = b \cdot Hh$ follows from the left-hand component. □

Remark 3.7.

(i) In [AMV$_3$] there is also a notion of a non-complete Elgot algebra. Since we will only be interested in using complete Elgot algebras we will henceforth understand by an Elgot algebra a complete one.

(ii) The axioms of Elgot algebras are inspired by the axioms of iteration theories of Bloom and Ésik [BE]. In fact, the two laws above are essentially "flat" versions of the commutative identities and the left pairing identity (also known as Bekić-Scott law) from [BE].

One justification for the above axioms is that Elgot algebras turn out to be the Eilenberg-Moore category of the monad T, see Subsection 3.5. We shall mention this result at the end of this section. Applied to a signature functor H_Σ on Set, that means a Σ-algebra A is an Elgot algebra precisely if there exists a canonical map $T_\Sigma A \longrightarrow A$ evaluating all Σ-trees in A.

(iii) Notice, that flat equation morphisms are precisely the coalgebras of the functor $H(\_) + A$. Thus, the Functoriality above states that $(\_)^\dagger$ is a functor from the category of flat equation morphisms to the comma category $\mathcal{A}/A$.

Examples 3.8. We present some examples of Elgot algebras. None but the first are in general cias.

(i) Completely iterative algebras are Elgot algebras. It is proved in [AMV$_3$] that for a cia the assignment of the unique solution to any flat equation morphism satisfies the Functoriality and the Compositionality.

(ii) Continuous algebras. Let H be a locally continuous endofunctor on CPO, see Example 2.2(ii). It is shown in [AMV$_3$] that any H-algebra (A, a) with a least element $\bot$ is an Elgot algebra when to a flat equation morphism $e : X \longrightarrow HX + A$ the least solution $e^\dagger$ is assigned. More precisely, define $e^\dagger$ as the join of the following increasing ω-chain in $\mathsf{CPO}(X, A)$: $e^\dagger_0$ is the constant function $\bot$; and given $e^\dagger_n$ let $e^\dagger_{n+1} = [a, A] \cdot (He^\dagger_n + A) \cdot e$, so that Diagram (3.3) commutes.

(iii) Suppose that $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ is a functor with a locally continuous lifting $H' : \mathsf{CPO} \longrightarrow \mathsf{CPO}$. An H-algebra $\alpha : HA \longrightarrow A$ is called CPO-*enrichable* if there exists a complete partial order $\sqsubseteq$ on A such that A becomes a continuous algebra $\alpha : H'(A, \sqsubseteq) \longrightarrow (A, \sqsubseteq)$ with a least element. Any CPO-enrichable algebra A is an Elgot algebra: to every equation morphism $e : X \longrightarrow HX + A$, let $\leq$ be the discrete order on X, consider $\widehat{e} : (X, \leq) \longrightarrow H'(X, \leq) + (A, \sqsubseteq)$ defined in the obvious way, and assign $U\widehat{e}^\dagger : X \longrightarrow A$, where $\widehat{e}^\dagger$ is from part (ii), and $U : \mathsf{CPO} \longrightarrow \mathsf{Set}$ is the forgetful functor.

(iv) Every complete lattice A is an Elgot algebra of the endofunctor $HX = X \times X$ of Set. In fact, taking binary joins yields an H-algebra structure $\vee : A \times A \longrightarrow A$. Furthermore, observe that the algebra TA of all binary trees over A has an evaluation $\alpha : TA \longrightarrow A$ mapping every binary tree in TA to the join of its leaf labels. For any flat equation morphism $e : X \longrightarrow X \times X + A$ form the flat equation morphism $\eta_A \bullet e : X \longrightarrow X \times X + TA$ take its unique solution $(\eta_A \bullet e)^\dagger : X \longrightarrow TA$ and let $e^* = \alpha \cdot (\eta_A \bullet e)^\dagger$. Then $(A, a, (\_)^*)$ is an Elgot algebra for H, see [AMV$_3$], Example 3.9. Notice that this is usually not a cia since the formal equation $x \approx x \vee x$ has in general many different solutions in a complete lattice.

3.3. **Computation Tree Elgot Algebras.** In this section we present Elgot algebras for a signature that uses undefined elements and also conditionals. Let Σ be a signature, and let $H = H_\Sigma$ be the associated endofunctor on Set. Let (A, a) be any H_Σ-algebra, and let $\uparrow$ be any element of A. We shall define an Elgot algebra structure on A related to the natural computation tree semantics of recursive definitions, where solutions are obtained by rewriting. The idea is that $\uparrow$ is our "scapegoat" for ungrounded definitions.

We shall assume that the algebra A interprets the function symbols in Σ in a strict fashion: if any argument a_i is $\uparrow$, then the overall value $g_A(a_1, \ldots, a_n)$ is $\uparrow$ as well. Conversely, if $g_A(a_1, \ldots, a_n) = \uparrow$, we require that ($n \geq 1$ and) some a_i is $\uparrow$. We make this assumption for all function symbols g *except for* the *conditional symbol* $\mathsf{ifzero} \in \Sigma_3$. We make a different assumption on ifzero. For this, fix an element $0 \in A$ other than $\uparrow$. We want

$$\mathsf{ifzero}_A(x, y, z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{if } x \neq 0 \text{ and } x \neq \uparrow \\ \uparrow & \text{otherwise} \end{cases} \tag{3.4}$$

To summarize, in this section we work with algebras of signature functors on Set which come with designated objects $\uparrow$ and 0 satisfying the assumptions above.

We shall work with partial functions and we use some notation which is standard. For partial functions $p, q : X \rightharpoonup A$, $p(x)\uparrow$ means that p is not defined on x, and we write $p(x)\downarrow$ if p is defined on x. Finally, $p(x) \simeq p(y)$ means that if either $p(x)$ or $q(y)$ is defined, then so is the other; and in this case, the values are the same.

Definition 3.9. Let $e : X \longrightarrow HX + A$ be a flat equation morphism in A. We define a partial function $\hat{e} : X \rightharpoonup A$ as follows:

(i) If $e(x) = a$ and $a \neq \uparrow$, then $\hat{e}(x) \simeq a$.
(ii) If $e(x) = g(x_1, \ldots, x_k)$, $g \neq \mathsf{ifzero}$, and for each i, $\hat{e}(x) \simeq a_i$, then $\hat{e}(x) \simeq g_A(a_1, \ldots, a_k)$.
(iii) If $e(x) = \mathsf{ifzero}(y, z, w)$ and $\hat{e}(y) \simeq 0$, then $\hat{e}(x) \simeq \hat{e}(z)$.
(iv) If $e(x) = \mathsf{ifzero}(y, z, w)$ and $\hat{e}(y)\downarrow$ but $\hat{e}(y) \not\simeq 0$, then $\hat{e}(x) \simeq \hat{e}(w)$.

We call $\hat{e}$ the *computation function corresponding to* e.

We intend this to be a definition of a partial function by recursion, so that we may prove facts about $\hat{e}$ by induction. Here is a first example, one which will be important in Proposition 3.10 below: if $\hat{e}(x)\downarrow$, then $\hat{e}(x) \neq \uparrow$.

Now that we have $\hat{e}$, we define $e^\dagger(x)$ to be $\hat{e}(x)$ if $\hat{e}$ is defined; if it is not, we set $e^\dagger(x) = \uparrow$. (Note that $e^\dagger(x) = \uparrow$ iff $\hat{e}(x)\uparrow$.)

In the statement of the result below, we also mention the main way that one obtains structures which satisfy the standing hypotheses of this section.

Proposition 3.10. *Let $A_0 = (A_0, a_0)$ be any H_Σ-algebra, let $0 \in A_0$, let $\uparrow \notin A_0$, and let $A = A_0 \cup \{\uparrow\}$. Let $A = (A, a)$ be defined in terms of this data by extending a_0 to the function $a : H_\Sigma A \longrightarrow A$ strictly on all function symbols except* ifzero*, and with* ifzero_A *given by (3.4). Let $(\_)^\dagger$ be as above. Then $(A, a, (\_)^\dagger)$ is an Elgot algebra.*

Proof. Clearly the assumption of this section hold for the algebra A. These assumptions ensure that A is CPO-enrichable. In fact, equip A with the flat cpo structure with the least element $\uparrow$. Then all operations on A are easily checked to be monotone, whence continuous; thus, $a : H_\Sigma A \longrightarrow A$ is a continuous algebra. By Example 3.8(iii), we obtain an Elgot algebra $(A, a, (\_)^*)$. We will prove that for any flat equation morphism $e : X \longrightarrow HX + A$ the least solution e^* agrees with the map $e^\dagger$ given by the computation function $\hat{e}$. To this end recall first that the set $\mathsf{Par}(X, A)$ of partial functions from X to A is a cpo with the order given by $f \sqsubseteq g$ if for all $x \in X$, $f(x)\downarrow$ implies $g(x)\downarrow$ and $f(x) = g(x)$. Now observe that the definition of $\hat{e}$ by recursion means that $\hat{e}$ is the join of an increasing chain in $\mathsf{Par}(X, A)$. In fact, let $\hat{e}_0$ be the everywhere undefined function and given $\hat{e}_n$ define $\hat{e}_{n+1}$ as follows: in the clauses (i)–(iv) in Definition 3.9 replace the term $\hat{e}(x)$ by $\hat{e}_{n+1}(x)$ and replace all occurrences of $\hat{e}$ on right-hand sides of defining equations by $\hat{e}_n$. Clearly, $(\hat{e}_n)_{n<\omega}$ is an increasing chain in $\mathsf{Par}(X, A)$ whose join is $\hat{e}$.

Now recall from Example 3.8(ii) that e^* is the join of the chain e^*_n in $\mathsf{CPO}(X, A)$, where X is discretely ordered. We shall show by induction that for every $x \in X$ the equation

$$e^*_n(x) = \begin{cases} \hat{e}_n(x) & \text{if } \hat{e}_n(x)\downarrow \\ \uparrow & \text{else} \end{cases} \tag{3.5}$$

holds. The base case is obvious. For the induction step we proceed by case analysis. If $e(x) = a$, then $e^*_{n+1}(x) = a$ and so (3.5) holds. The second case is $e(x) = g(x_1, \ldots, x_k)$, $g \neq \mathsf{ifzero}$. We have $e^*_{n+1}(x) = g_A(e^*_n(x_1), \ldots, e^*_n(x_k))$. By our assumptions, this equals $\uparrow$ precisely if at least one of the $e^*_n(x_j)$ is $\uparrow$, which in turn holds if and only if $\hat{e}_n(x_k)\uparrow$ for some j; and equivalently, $\hat{e}_{n+1}(x)\uparrow$. Otherwise all $\hat{e}_n(x_j)$ are defined and by induction hypothesis we get

$$e^*_{n+1}(x) = g_A(e^*_n(x_1), \ldots, e^*_n(x_k)) = g_A(\hat{e}_n(x_1), \ldots, \hat{e}_n(x_k)) = \hat{e}_{n+1}(x)\,.$$

Thirdly, assume that $e(x) = \mathsf{ifzero}(y, z, w)$. Then similarly as before we have

$$e^*_{n+1}(x) = \mathsf{ifzero}(e^*_n(y), e^*_n(z), e^*_n(w))\,.$$

We obtain $e^*_{n+1}(x) = \uparrow$ whenever $e^*_n(y) = \uparrow$. But this happens precisely if $\hat{e}_n(y)\uparrow$, which implies that $\hat{e}_{n+1}(x)\uparrow$. Now if $e^*_n(y) \neq \uparrow$, then we have equivalently that $\hat{e}_n(y)\downarrow$. We treat here only the case that $\hat{e}_n(y) = 0$; the remaining case is similar. In our present case it follows that $e^*_{n+1}(x) = e^*_n(z)$ and $\hat{e}_{n+1}(x) \simeq \hat{e}_n(z)$. Therefore, by the induction hypothesis, the desired equation (3.5) holds.

Finally, from (3.5) we conclude that for the least fixed points e^* and $\hat{e}$ we have

$$e^*(x) = \begin{cases} \hat{e}(x) & \text{if } \hat{e}(x)\downarrow \\ \uparrow & \text{else.} \end{cases}$$

Thus, we get $e^* = e^\dagger$ which completes the proof. □

Definition 3.11. Let $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$ be a signature functor, let $A_0 = (A_0, a_0)$ be any H_Σ-algebra as in the hypothesis of Proposition 3.10. We call the Elgot algebra $(A, a, (\_)^\dagger)$ the *computation tree Elgot algebra* induced by A_0.

We shall study the interpreted solutions of recursive program schemes in computation tree Elgot algebras in Section 7.1.

3.4. **Dramatis Personae.** As we mentioned already the classical theory of recursive program schemes rests on the fact that in any continuous algebra A all Σ-trees over A can be evaluated; i. e., there is a canonical map $T_\Sigma A \longrightarrow A$. In a suitable category of cpos the structures $T_\Sigma X$ play the rôle of *free algebras*. The freeness is used to define maps *out* of those algebras. In our setting, the Σ-trees are the final coalgebra. So in order to generalize the classical theory, we need a setting in which the final coalgebras TY are *free algebras*. The following result gives such a setting. It is fundamental for the rest of the paper and collects the results of Theorems 2.8 and 2.10 of [Mi_2] and Theorem 5.6 of [AMV_3]. We sketch a proof for the convenience of the reader.

Theorem 3.12. *Let H be any endofunctor of $\mathcal{A}$. The following are equivalent:*

(i) *TY is a final coalgebra of $H(\_) + Y$,*
(ii) *TY is a free completely iterative H-algebra on Y, and*
(iii) *TY is a free (complete) Elgot H-algebra on Y.*

In more detail: if (TY, α_Y) is a final coalgebra for $H(\_) + Y$, the inverse $[\tau_Y, \eta_Y] : HTY + Y \longrightarrow Y$ of α_Y gives a cia for H, which as an Elgot algebra is free on Y. Conversely, given a free Elgot H-algebra $(TY, \tau_Y, (\_)^\dagger)$ with a universal arrow $\eta_Y : Y \longrightarrow TY$, then this is a cia, whence a free cia on Y, and $[\tau_Y, \eta_Y]$ is an isomorphism whose inverse is the structure map of a final coalgebra for $H(\_) + Y$.

Sketch of Proof. Suppose first that (TY, α_Y) is a final coalgebra for $H(\_) + Y$. Let $[\tau_Y, \eta_Y]$ be the inverse of α_Y. Then $\tau_Y : HTY \longrightarrow TY$ is a completely iterative algebra for H, and therefore an Elgot algebra. In fact, for any flat equation morphism $e : X \longrightarrow HX + TY$, form the following coalgebra

$$c \equiv X + TY \xrightarrow{[e,\mathsf{inr}]} HX + TY \xrightarrow{HX+\alpha_Y} HX + HTY + Y \xrightarrow{\mathsf{can}+Y} H(X + TY) + Y$$

and define

$$e^\dagger \equiv X \xrightarrow{\mathsf{inl}} X + TY \xrightarrow{h} TY\,,$$

where h is the unique homomorphism from the coalgebra $(X + TY, c)$ to the final one. It is not difficult to prove that $e^\dagger$ is the unique solution of e.

Furthermore, $(TY, \tau_Y, (\_)^\dagger)$ is a free Elgot algebra on Y. For any Elgot algebra $(A, a, (\_)^\ddagger)$ and any morphism $m : Y \longrightarrow A$ form the equation morphism

$$m \bullet \alpha_Y \equiv TY \xrightarrow{\alpha_Y} HTY + Y \xrightarrow{HTY+m} HTY + A$$

It is shown in Theorem 5.8 of [AMV$_3$] that the solution $h = (m \bullet \alpha_Y)^\ddagger$ yields the unique homomorphism $h : TY \longrightarrow A$ of Elgot algebras such that $h \cdot \eta_Y = m$.

Now conversely, assume that $(TY, \tau_Y, (\_)^\dagger)$ is a free Elgot algebra on Y with a universal arrow $\eta_Y : Y \longrightarrow TY$. It can be shown that $[\tau_Y, \eta_Y]$ is an isomorphism, see Lemma 5.7 of [AMV$_3$]. Denote by α_Y its inverse. Then (TY, τ_Y) is a cia; i. e., for any flat equation morphism $e : X \longrightarrow HX + TY$ the solution $e^\dagger$ is unique. In fact, suppose that s is any solution of e. It follows that s is a morphism of equations from e to the flat equation morphism

$$f \equiv TY \xrightarrow{\alpha_Y} HTY + Y \xrightarrow{HTY+\eta_Y} HTY + TY\,.$$

Thus, $f^\dagger \cdot s = e^\dagger$ by Functoriality of $(\_)^\dagger$. Next one can show, using the Compositionality, that $f^\dagger : TY \longrightarrow TY$ is a homomorphism of Elgot algebras satisfying $f^\dagger \cdot \eta_Y = \eta_Y$. Thus, by the freeness of TY, $f^\dagger = id$. This proves that (TY, τ_Y) is a cia, which implies that it is the free one on Y. It is not difficult to show that this yields a final coalgebra of $H(\_) + Y$. In fact, for any coalgebra $c : C \longrightarrow HC + Y$ the unique solution of the flat equation morphism $\eta_Y \bullet c$ yields a unique homomorphism $(C, c) \longrightarrow (TY, \alpha_Y)$ of coalgebras. □

Theorem 3.12 has an important consequence for our work. Recall that we assume H is iteratable, so $H(\_) + Y$ *does* have a final coalgebra for all Y. The next result gives the *dramatis personae* for the rest of the paper.

Theorem 3.13. *There is a left adjoint to the forgetful functor from $\mathsf{Alg}^\dagger\, H$, the category of Elgot algebras and their homomorphisms, to the base category $\mathcal{A}$*

$$\mathsf{Alg}^\dagger\, H \underset{U}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathcal{A}\,.$$

The left-adjoint L assigns to each object Y of $\mathcal{A}$ a free Elgot algebra $(TY, \tau_Y, (\_)^\dagger)$ on Y (equivalently, (TY, α_Y) where $\alpha_Y = [\tau_Y, \eta_Y]^{-1}$ is a final coalgebra of $H(\_) + Y$). The unit of the adjunction is η whose components are given by the universal arrows $\eta_Y : Y \longrightarrow TY$ of the free Elgot algebras. The counit ε gives for each Elgot algebra $(A, a, (\_)^\ddagger)$ the unique homomorphism $\widetilde{a} : TA \longrightarrow A$ of Elgot algebras such that $\widetilde{a} \cdot \eta_A = id$. We have

$$\widetilde{a} = (\alpha_A)^\ddagger : TA \longrightarrow A\,, \tag{3.6}$$

where $\alpha_A : TA \longrightarrow HTA + A$ is considered as a flat equation morphism with parameters in A.

Moreover, we obtain additional structure:

(i) *A monad $(\mathcal{T}(H), \eta^H, \mu^H)$ on $\mathcal{A}$ such that for all objects Y of $\mathcal{A}$,*
  (a) *$\mathcal{T}(H)Y = TY$ is the carrier of a final coalgebra of $H(\_) + Y$;*

(b) μ^H_Y *is the (unique) solution of* α_{TY}*, considered as a flat equation morphism with parameters in* TY*.*

(ii) *A natural transformation* $\alpha^H : T \longrightarrow HT + Id$*.*

(iii) *A natural transformation* $\tau^H : HT \longrightarrow T$ *such that* $[\tau^H, \eta^H]$ *is the inverse of* α^H*.*

(iv) *A "canonical embedding"* κ^H *of* H *into* T*:*

$$\kappa^H \equiv H \xrightarrow{H\eta^H} HT \xrightarrow{\tau^H} T\,. \tag{3.7}$$

Proof. It is obvious that the assignment of Y to a free Elgot algebra on Y yields a left-adjoint to U. This adjunction gives rise to a monad $(\mathcal{T}(H), \eta^H, \mu^H)$ on $\mathcal{A}$ which assigns to every object of $\mathcal{A}$ the underlying object TY of a free Elgot algebra on Y, see Section 2.4. Thus item (a) follows from Theorem 3.12. The monad multiplication μ^H is given by $U\varepsilon L$, i. e., $\mu^H_Y : TTY \longrightarrow TY$ is the unique homomorphism of Elgot algebras such that $\mu^H_Y \cdot \eta^H_{TY} = id$. It follows from the proof of Theorem 3.12 that $\widetilde{a} = (m \bullet \alpha_A)^\ddagger$, where m is the identity on A. The special instance of this where A is the free Elgot algebra TY yields (b). The functoriality of T implies that the algebra structures τ_Y, and the coalgebra structures α_Y, Y in $\mathcal{A}$, form natural transformations. It follows from Theorem 3.12 that α^H and $[\tau^H, \eta^H]$ are mutually inverse. □

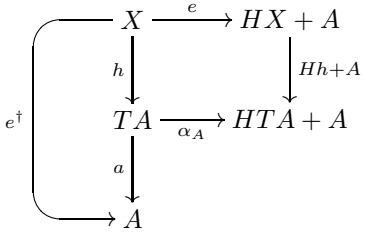
We call the monad $(\mathcal{T}(H), \eta^H, \mu^H)$ the *completely iterative monad* generated by H. (The name comes from an important property which we discuss in Section 4.) As always, we just write $\mathcal{T}(H)$, or even shorter T to denote this monad, and we shall frequently drop the superscripts when dealing with the structure coming from a single endofunctor H. (But as the reader will see later, we frequently do need to consider the structures coming from two endofunctors. This is particularly pertinent in our study since in recursive program schemes we usually have two signatures, hence two functors, see Section 2.3).

For any Elgot algebras $(A, a, (\_)^\dagger)$ for H we call the homomorphism $\widetilde{a} : TA \longrightarrow A$ in (3.6) above the *evaluation morphism* of that Elgot algebra. Theorem 3.14 below shows that Elgot algebras are equivalently presented by their evaluation morphisms.

3.5. **The Eilenberg-Moore Category of $\mathcal{T}(H)$.**

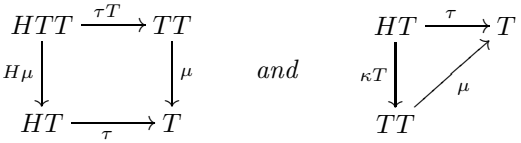
Theorem 3.14. [AMV$_3$] *The category* $\mathsf{Alg}^\dagger H$ *of Elgot algebras is isomorphic to the Eilenberg-Moore category* $\mathcal{A}^T$ *of monadic* T*-algebras. More precisely, for any Elgot algebra* $A = (A, a, (\_)^\dagger)$*, the evaluation morphism* $U\epsilon_A = \widetilde{a} : TA \longrightarrow A$ *is an Eilenberg-Moore algebra of* T*.*

Conversely, for any Eilenberg-Moore algebra $a : TA \longrightarrow A$ *we obtain an Elgot algebra by using as structure map the composite* $a \cdot \kappa_A : HA \longrightarrow A$*, and by defining for a flat equation morphism* $e : X \longrightarrow HX + A$ *the solution* $e^\dagger = a \cdot h$*, where* h *is the unique coalgebra homomorphism from* (X, e) *to* (TA, α_A)*:*

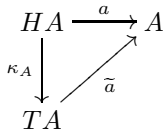


These two constructions extend to the level of morphisms, and they yield the desired isomorphism between the two categories $\mathsf{Alg}^\dagger H$ *and* $\mathcal{A}^T$*.*

Corollary 3.15. *The diagrams*



commute, and for every Elgot algebra $(A, a, (\_)^\dagger)$ *the triangle*



commutes.

Proof. The lower triangle commutes because the two constructions of Theorem 3.14 are mutually inverse. The special cases of this triangle for $A = TY$ and $a = \tau_Y$ yield the commutativity of the upper right-hand triangle since $\mu_Y = U\varepsilon_{TY} = \widetilde{\tau_Y}$. Finally, the upper left-hand square commutes since for each Y in $\mathcal{A}$, μ_Y is a homomorphism of Elgot algebras, whence an H-algebra homomorphism by Proposition 3.6. □

4. Completely Iterative Monads

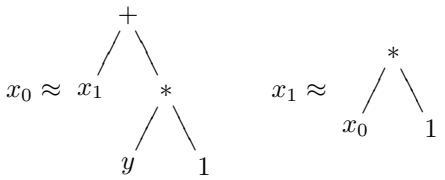
Before we can state a theorem providing solutions of (generalized) recursive program schemes, we need to explain what a solution is. In the classical setting one introduces *second-order substitution* of all Σ-trees, i. e., substitution of trees for operation symbols, see [C]. We present a generalization of second-order substitution to the final coalgebras given by $\mathcal{T}(H)$.

In fact, in [AAMV, Mi$_1$] it is proved that the monad $\mathcal{T}(H)$ of Theorem 3.13 is characterized by an important universal property—it is the free *completely iterative monad* on H. This freeness of $\mathcal{T}(H)$ specializes to second-order substitution of Σ-trees, a fact we illustrate at the end of the current section.

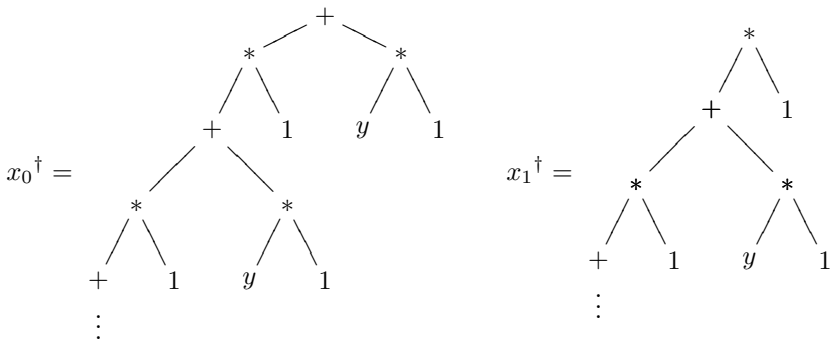
Here we shall quickly recall those results of [AAMV] which we will need in the current paper. For a well-motivated and more detailed exposition of the material presented here we refer the reader to one of [AAMV, Mi$_2$].

Example 4.1. We have seen in Section 3 that for a signature Σ, flat systems of formal equations have unique solutions whose components are Σ-trees over a set of parameters. But it is also possible to uniquely solve certain non-flat systems equations. More precisely, for a given signature Σ, consider a system of equations (3.1) where each right-hand side t_i, $i \in I$ is any Σ-tree from $T_\Sigma(X + Y)$ which is not just a single variable from X. Such systems are called *guarded*. Guardedness suffices to ensure the existence of a unique solution of the given system.

For example, let Σ consist of binary operations $+$ and $*$ and a constant 1. The following system of equations



with $X = \{ x_0, x_1 \}$ and $Y = \{ y \}$ is guarded. The solution is given by the following trees in $T_\Sigma Y$:



Unique solutions of guarded equations can be obtained more generally for every monad $\mathcal{T}(H)$ of Theorem 3.13 above. Before we state this result precisely we recall the notion of an ideal monad. It adds to an arbitrary monad S enough structure to be able to speak of guarded equations for S.

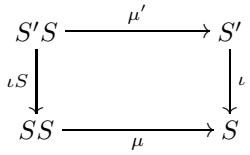
Definition 4.2. By an *ideal monad* we mean a six-tuple

$$(S, \eta, \mu, S', \iota, \mu')$$

consisting of a monad (S, η, μ), and a (right) *ideal* (S', μ'), which consists of a subfunctor $\iota : S' \hookrightarrow S$, i. e., a monomorphism ι in the functor category $[\mathcal{A}, \mathcal{A}]$, and a natural transformation $\mu' : S'S \longrightarrow S'$ such that the following two conditions hold:

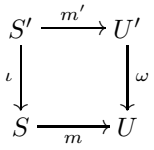
(i) $S = S' + Id$ with coproduct injections ι and η

(ii) μ restricts to μ' along ι, i.e., the square

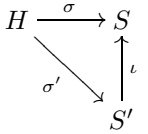


commutes.

An *ideal monad morphism* from an ideal monad $(S, \eta^S, \mu^S, S', \iota, \mu'^S)$ to an ideal monad $(U, \eta^U, \mu^U, U', \omega, \mu'^U)$ is a monad morphism $m : (S, \eta^S, \mu^S) \longrightarrow (U, \eta^U, \mu^U)$ which has a domain-codomain restriction to the ideals. That is, there exists a natural transformation $m' : S' \longrightarrow U'$ such that the square below commutes:



For any endofunctor H and ideal monad S, a natural transformation $\sigma : H \longrightarrow S$ is *ideal* if it factors through the ideal $\iota : S' \hookrightarrow S$ as follows:

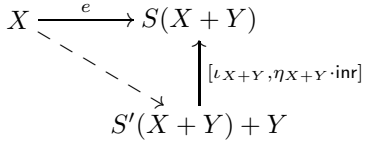


Example 4.3. Recall that the underlying functor of the monad T of Theorem 3.13 is a coproduct of HT and Id. Taking for ι the left-hand coproduct injection $\tau : HT \hookrightarrow T$ and for μ' the natural transformation $H\mu : HTT \longrightarrow HT$ we see that T is an ideal monad. Furthermore, since κ in (3.7) is $\tau \cdot H\eta$, $\kappa : H \longrightarrow T$ is an ideal natural transformation.

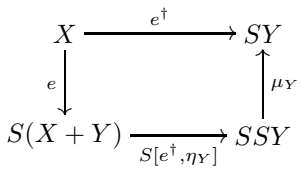
Definition 4.4. (i) For an ideal monad S on $\mathcal{A}$ an *equation morphism* is a morphism

$$e : X \longrightarrow S(X + Y).$$

It is called *guarded* if it factors as follows:



(ii) A *solution* of an equation morphism e is a morphism $e^\dagger : X \longrightarrow SY$ such that the following square



commutes.

(iii) An ideal monad is called *completely iterative* provided that any guarded equation morphism has a unique solution.

The first item of the following result is called the *Parametric Corecursion Theorem* in [Mo$_1$] and the *Solution Theorem* in [AAMV]. See also [Mi$_2$] for an extension of this result to all cias. The second item is the main result of [AAMV].

Theorem 4.5. *For any iteratable endofunctor H,*

(i) *the ideal monad $T = \mathfrak{T}(H)$ is completely iterative, and*

(ii) *T is free on H. More precisely, for all completely iterative monads S and ideal natural transformations $\sigma : H \longrightarrow S$ there exists a unique monad morphism $\overline{\sigma} : T \longrightarrow S$ such that $\overline{\sigma} \cdot \kappa^H = \sigma$:*

$$\begin{array}{ccc} H & \xrightarrow{\kappa^H} & T \\ & \searrow^{\forall \sigma} & \downarrow \exists! \overline{\sigma} \\ & & S\,. \end{array} \tag{4.1}$$

And the induced morphism $\overline{\sigma}$ is an ideal monad morphism.

In our work in the subsequent sections we shall often use the special case of Theorem 4.5 where the completely iterative monad S is $\mathcal{T}(K)$ for some iteratable endofunctor K. For that special case we need the following explicit description of the restriction of the monad morphism $\overline{\sigma}$ to the subfunctors $H\mathcal{T}(H)$ and $S' = K\mathcal{T}(K)$.

Lemma 4.6. *If H and K are iteratable endofunctors and $\sigma : H \longrightarrow \mathcal{T}(K)$ is an ideal transformation, i. e., $\sigma = \tau^K \cdot \sigma'$, then for the unique induced ideal monad morphism $\overline{\sigma}$ the following is a commutative diagram:*

$$\begin{array}{ccccc} H\mathcal{T}(H) & \xrightarrow{\sigma' * \overline{\sigma}} & K\mathcal{T}(K)\mathcal{T}(K) & \xrightarrow{K\mu^K} & K\mathcal{T}(K) \\ \downarrow{\scriptstyle \tau^H} & & & & \downarrow{\scriptstyle \tau^K} \\ \mathcal{T}(H) & & \xrightarrow[\overline{\sigma}]{} & & \mathcal{T}(K) \end{array}$$

Proof. Consider the diagram

$$\begin{array}{ccccc} H\mathcal{T}(H) & \xrightarrow{\sigma' * \overline{\sigma}} & K\mathcal{T}(K)\mathcal{T}(K) & \xrightarrow{K\mu^K} & K\mathcal{T}(K) \\ \downarrow{\scriptstyle \kappa^H * \mathcal{T}(H)} & & \downarrow{\scriptstyle \tau^K * \mathcal{T}(K)} & & \\ \mathcal{T}(H)\mathcal{T}(H) & \xrightarrow{\overline{\sigma} * \overline{\sigma}} & \mathcal{T}(K)\mathcal{T}(K) & \searrow^{\mu^K} & \downarrow{\scriptstyle \tau^K} \\ \downarrow{\scriptstyle \mu^H} & & & & \\ \mathcal{T}(H) & & \xrightarrow[\overline{\sigma}]{} & & \mathcal{T}(K) \end{array}$$

(with τ^H from the upper left to the lower left)

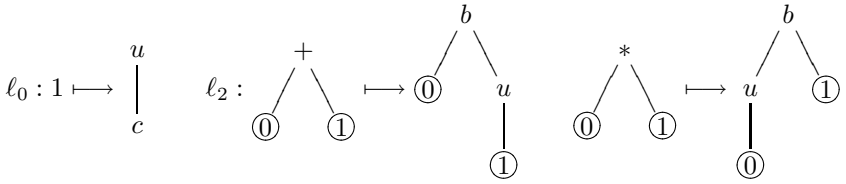
It clearly commutes. The left-hand and right-hand parts commute by Corollary 3.15. The upper part commutes by naturality and since $\overline{\sigma} \cdot \kappa^H = \sigma = \tau^K \cdot \sigma'$, and the lower one since $\overline{\sigma}$ is a monad morphism. □

For signature functors on Set, the freeness of $T = \mathcal{T}(H_\Sigma)$ specializes to *second-order substitution*, i. e., substitution of (finite or infinite) trees for operation symbols. Second-order substitution is a key point in the classical theory of recursive program schemes because the notion of an *uninterpreted solution* rests on it. We believe that the connection of second-order substitution and any notion of freeness is new.

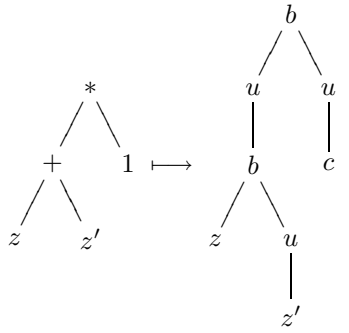
Example 4.7. Let Σ and Γ be signatures (considered as functors $\mathbb{N} \longrightarrow \mathsf{Set}$). Each symbol $\sigma \in \Sigma_n$ is considered as a flat tree in n variables. A second-order substitution gives an "implementation" to each such σ as a Γ-tree in the same n variables. We model this by a natural transformation $\ell : \Sigma \longrightarrow T_\Gamma \cdot J$, i. e., a family or maps $\ell_n : \Sigma_n \longrightarrow T_\Gamma\{0, \ldots n-1\}$, $n \in \mathbb{N}$. As we have seen in Section 2.3, this gives rise to a natural transformation $\lambda : H_\Sigma \longrightarrow T_\Gamma$. When infinite trees are involved there is usually the restriction to so-called *non-erasing* substitutions; i. e., all Σ-symbols are assigned to trees which are not just single node trees labelled by a variable. That means that λ is an ideal natural transformation. Thus, from Theorem 4.5 we get a monad morphism $\overline{\lambda} : T_\Sigma \longrightarrow T_\Gamma$. For any set X of variables its action is that of second-order substitution, i. e., $\overline{\lambda}_X$ replaces every Σ-symbol in a tree t from $T_\Sigma X$ by its implementation according to λ. More precisely, let $t = \sigma(t_1, \ldots, t_n)$ with $\sigma \in \Sigma_n$ and let $t'(x_1, \ldots x_n) \in T_\Gamma X$ be the implementation of σ, i. e., $\lambda_X(\sigma(\vec{x})) = t'(\vec{x})$. Then we have

$$\overline{\lambda}_X(t) = t'(\overline{\lambda}_X(t_1), \ldots, \overline{\lambda}_X(t_n)).$$

For example, suppose that Σ consists of two binary symbols $+$ and $*$ and a constant 1, and Γ consists of a binary symbol b, a unary one u and a constant c. Furthermore, let λ be given by $\ell : \Sigma \longrightarrow T_\Gamma \cdot J$ as follows:



and else ℓ_n is the unique map from the empty set. For the set $Z = \{ z, z' \}$, the second-order substitution morphism $\overline{\lambda}_Z$ acts for example as follows:



5. $\mathcal{T}(H)$ as Final Coalgebra among Monads

In this section we will state and prove some technical results which are essential for the proofs of our results on uninterpreted and interpreted program schemes. The culmination of the work comes in Corollary 5.5.

We would like to mention that the results and proofs in Section 5.1 are inspired by the work of Ghani et al. [GLM]. However, that paper does not work in the same category as we do. Our setting is slightly more general, and perhaps, conceptually slightly clearer. We do not believe that any of our subsequent new results in Sections 6 and 7 can be obtained by simply applying the work of [GLM].

We still assume that every functor H we consider is an iteratable endofunctor. Recall that for each object Y, TY is a final coalgebra for the functor $H(\_) + Y$ on $\mathcal{A}$.

We are going to prove a number of results that strengthen this. First, consider the functor category $[\mathcal{A}, \mathcal{A}]$. H may be regarded as a functor on this, by $F \longmapsto H \cdot F$. We also get a related functor $H \cdot \_ + Id$. For the functor T, the value of this functor at T is $H \cdot T + Id$. So the natural transformation $\alpha^H : T \longrightarrow HT + Id$ of Theorem 3.13(ii) may be regarded as a coalgebra structure for T. The proof of the following theorem is straightforward and therefore left to the reader.

Theorem 5.1. *(T, α) is a final coalgebra for $H \cdot \_ + Id$.*

5.1. T Gives a Final Coalgebra as a Monad.

We next consider the comma-category

$$H/\mathbf{Mon}(\mathcal{A}),$$

whose objects are given by pairs $(S, \sigma : H \longrightarrow S)$, where S is a monad on $\mathcal{A}$, and morphisms $h : (S_1, \sigma_1) \longrightarrow (S_2, \sigma_2)$ are monad morphisms $h : S_1 \longrightarrow S_2$ such that $h \cdot \sigma_1 = \sigma_2$. For example, one object of $H/\mathbf{Mon}(\mathcal{A})$ is (T, κ), where $\kappa = \tau \cdot H\eta$ is the canonical natural transformation of Theorem 3.13(iv). We show that H determines an endofunctor $\mathcal{H}$ on this category, and that (T, κ) is the underlying object of a final $\mathcal{H}$-coalgebra. We then extend this finality result by considering a subcategory of $H/\mathbf{Mon}(\mathcal{A})$. We slightly abuse notation and denote by $H/\mathbf{CIM}(\mathcal{A})$ the category whose objects are the pairs (S, σ), where S is *completely iterative* and σ is an *ideal* transformation; the morphisms in $H/\mathbf{CIM}(\mathcal{A})$ are given by the ideal monad morphisms, see Definitions 4.2 and 4.4. Again, (T, κ) is an object in this category, and we show that $\mathcal{H}$ restricts to an endofunctor on $H/\mathbf{CIM}(\mathcal{A})$, and that (T, κ) is once again a final coalgebra for $\mathcal{H}$.

Let us begin by defining $\mathcal{H}$ on objects of $H/\mathbf{Mon}(\mathcal{A})$ as the assignment

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

and for a morphism $h : (S_1, \sigma_1) \longrightarrow (S_2, \sigma_2)$ we let $\mathcal{H}(h) = Hh + Id$.

Furthermore, notice that for any object (S, σ) of $H/\mathbf{Mon}(\mathcal{A})$ there is a natural transformation

$$\xi_{(S,\sigma)} \equiv HS + Id \xrightarrow{[\mu \cdot \sigma S, \eta]} S$$

As it turns out, the $\xi_{(S,\sigma)}$ are the components of a natural transformation $\xi : \mathcal{H} \longrightarrow Id$ turning $\mathcal{H}$ into a well-copointed endofunctor on $H/\mathbf{Mon}(\mathcal{A})$.

Lemma 5.2.

(i) *$\mathcal{H}$ is an endofunctor of $H/\mathbf{Mon}(\mathcal{A})$;*
(ii) *$\xi : \mathcal{H} \longrightarrow Id$ is a natural transformation;*
(iii) *The functor $\mathcal{H}$ is well-copointed. That is, $\xi : \mathcal{H} \longrightarrow Id$ is a natural transformation with $\mathcal{H}\xi_{(S,\sigma)} = \xi_{\mathcal{H}(S,\sigma)}$ for all objects (S,σ) of $H/\mathbf{Mon}(\mathcal{A})$.*

Proof. (i) Given the object (S,σ) we define a natural transformation ν as

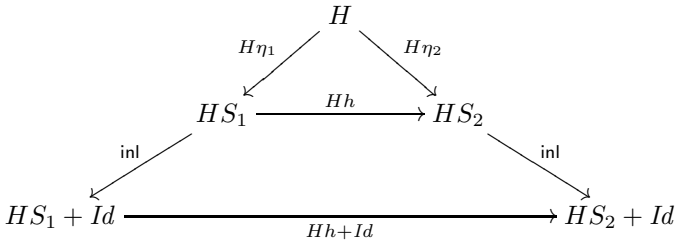
$$\begin{array}{c} (HS+Id)^2 = HS(HS+Id)+HS+Id \\ \Big\downarrow{\scriptstyle HS[\mu\cdot\sigma S,\eta]+HS+Id} \\ HSS+HS+Id \\ \Big\downarrow{\scriptstyle [H\mu,\mathsf{inl}]+Id} \\ HS+Id \end{array}$$

It is easy to check that $(HS+Id, \mathsf{inr}, \nu)$ is a monad, see [Mi$_1$], Lemma 3.4. Together with the natural transformation

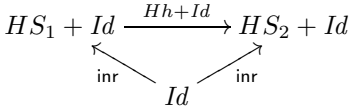
$$H \xrightarrow{H\eta} HS \xrightarrow{\mathsf{inl}} HS+Id$$

we obtain an object of $H/\mathbf{Mon}(\mathcal{A})$.

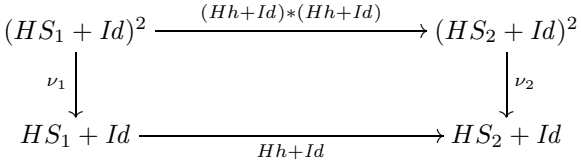
Now suppose that $h : (S_1,\sigma_1) \longrightarrow (S_2,\sigma_2)$ is a morphism in $H/\mathbf{Mon}(\mathcal{A})$. We establish below that $\mathcal{H}(h) = Hh + Id$ is a monad morphism. Together with the commutativity of the diagram



this establishes the action of $\mathcal{H}$ on morphisms. That $\mathcal{H}$ preserves identities and composition is obvious. Let us check then that $\mathcal{H}(h)$ is a monad morphism. The unit law is the commutativity of the triangle

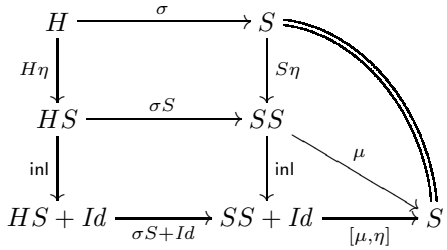


Thus, to complete the proof of (i) we must check that the following square commutes:

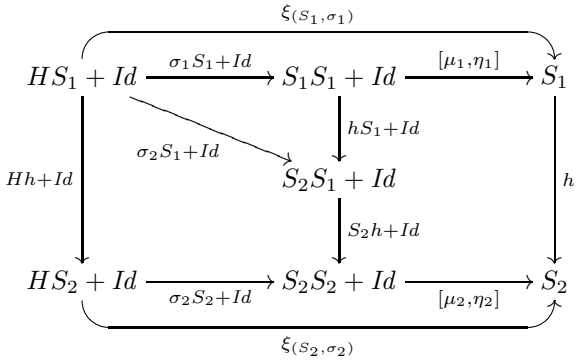


Expanding by using the definition of ν_i, $i = 1,2$, this is an essentially easy chase through some large diagrams using only the monad laws for the S_i as well as the fact that h is a morphism in $H/\mathbf{Mon}(\mathcal{A})$. For the sake of brevity we leave this straightforward task to the reader.

(ii) It is easy to prove that each component $\xi_{(S,\sigma)}$ is a monad morphism, see the proof of Theorem 3.1 in [Mi$_1$]. Moreover, by the commutativity of the following diagram, we obtain a morphism in $H/\mathbf{Mon}(\mathcal{A})$:



Finally, we check naturality of ξ. Suppose that $h : (S_1, \sigma_1) \longrightarrow (S_2, \sigma_2)$ is a morphism in $H/\mathbf{Mon}(\mathcal{A})$. Then, the following diagram commutes:



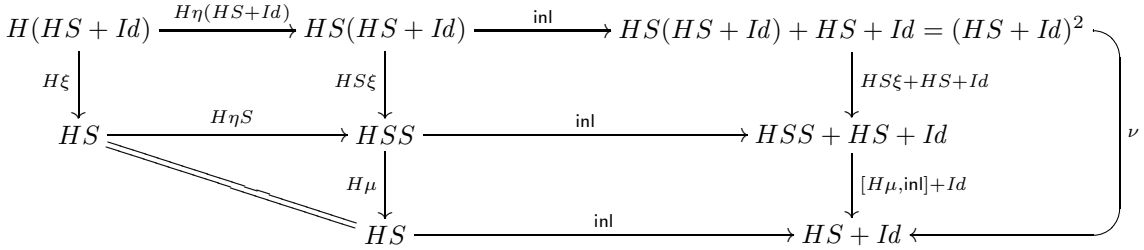
On the left we are using the naturality of σ_2, on the right and in the triangle that h is a morphism of $H/\mathbf{Mon}(\mathcal{A})$.

(iii) For any object (S, σ) we have by definition that

$$\xi_{\mathcal{H}(S,\sigma)} \equiv [\nu \cdot \mathsf{inl} \cdot H\eta(HS + Id), \mathsf{inr}] : H(HS + Id) + Id \longrightarrow HS + Id.$$

We show that this is the same as $H\xi_{(S,\sigma)} + Id$.

We consider the components of the left-hand coproduct separately. Equality on the right-hand component is obvious. For the left-hand one consider the following diagram:



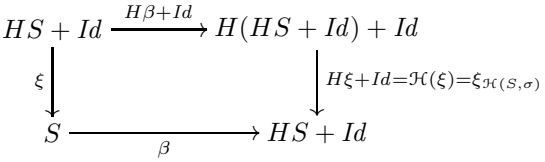
The upper and right outer edges compose to yield the left-hand component of $\xi_{\mathcal{H}(S,\sigma)}$ and the left and lower outer edges yield the left-hand component of $H\xi_{(S,\sigma)} + Id$. The desired commutativity of the outer shape follow since all inner parts of the diagram trivially commute. □

Lemma 5.3. *Let $((S, \sigma), \beta)$ be an $\mathcal{H}$-coalgebra with the structure $\beta : (S, \sigma) \longrightarrow \mathcal{H}(S, \sigma)$ in $H/\mathbf{Mon}(\mathcal{A})$. Then*

$$\xi_{(S,\sigma)} : \mathcal{H}(S, \sigma) \longrightarrow (S, \sigma)$$

is an $\mathcal{H}$-coalgebra homomorphism.

Proof. It is clear that $\mathcal{H}(S, \sigma) = (HS + Id, \mathsf{inl} \cdot H\eta)$ is an $\mathcal{H}$-coalgebra with the structure $H\beta + Id$, and that $\xi_{(S,\sigma)}$ is a morphism in $H/\mathbf{Mon}(\mathcal{A})$. From the naturality and well-copointedness of ξ, the following square



commutes; (see Lemma 5.2). □

Theorem 5.4. *$((T,\kappa),\alpha)$ is a final coalgebra for the functor $\mathcal{H}$ on $H/\mathbf{Mon}(\mathcal{A})$.*

Proof. Recall that the coalgebra structure $\alpha : T \longrightarrow HT + Id$ is given by the inverse of $[\tau,\eta] : HT + Id \longrightarrow T$. It is clearly a morphism in $H/\mathbf{Mon}(\mathcal{A})$. Indeed, recall from Corollary 3.15 that $\tau = \mu \cdot \kappa T$. Hence, $[\tau,\eta] = \xi_{(T,\kappa)}$ and so α being an inverse of a monad morphism is itself a monad morphism, and clearly we have $\alpha \cdot \kappa = \mathsf{inl} \cdot H\eta$.

Now suppose that $\beta : (S,\sigma) \longrightarrow \mathcal{H}(S,\sigma)$ is any $\mathcal{H}$-coalgebra. So the natural transformation $\beta : S \longrightarrow HS + Id$ is a monad morphism such that $\beta \cdot \sigma = \mathsf{inl} \cdot H\eta^S$. By Theorem 5.1, T is a final coalgebra on the level of endofunctors. Thus there exists a unique natural transformation $h : S \longrightarrow T$ such that the following square commutes:

$$\begin{array}{ccc} S & \xrightarrow{\beta} & HS + Id \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle Hh+Id} \\ T & \xrightarrow[\alpha]{} & HT + Id \end{array} \tag{5.1}$$

Hence our only task is to show that h is a morphism in $H/\mathbf{Mon}(\mathcal{A})$. With an easy computation we establish the unit law:

$$\begin{aligned} h \cdot \eta^S &= \alpha^{-1} \cdot (Hh + Id) \cdot \beta \cdot \eta^S && \text{(by (5.1))} \\ &= [\tau,\eta] \cdot (Hh + Id) \cdot \mathsf{inr} && \text{(since } \alpha^{-1} = [\tau,\eta]\text{)} \\ &= [\tau,\eta] \cdot \mathsf{inr} \\ &= \eta && \text{(computation with inr)} \end{aligned}$$

and from this it follows that

$$\begin{aligned} h \cdot \sigma &= \alpha^{-1} \cdot (Hh + Id) \cdot \beta \cdot \sigma && \text{(by (5.1))} \\ &= [\tau,\eta] \cdot (Hh + Id) \cdot \mathsf{inl} \cdot H\eta^S && \text{(since } \alpha^{-1} = [\tau,\eta]\text{)} \\ &= \tau \cdot Hh \cdot H\eta^S && \text{(composition with inl)} \\ &= \tau \cdot H\eta && \text{(since } h \cdot \eta^S = \eta\text{)} \\ &= \kappa && \text{(by (3.7))}\,. \end{aligned}$$

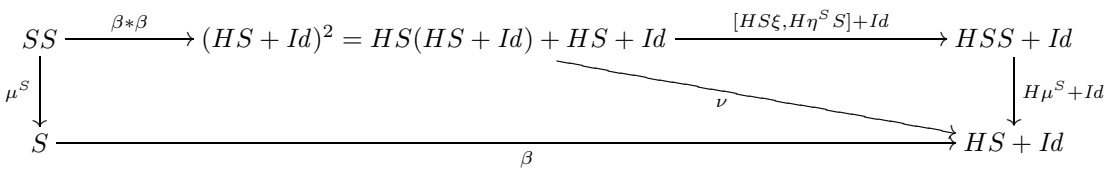
Finally, we check that the following square commutes:

$$\begin{array}{ccc} SS & \xrightarrow{\mu^S} & S \\ {\scriptstyle h*h}\downarrow & & \downarrow{\scriptstyle h} \\ TT & \xrightarrow[\mu]{} & T \end{array} \tag{5.2}$$

In order to do this we use that T is a final coalgebra for the functor $H \cdot \_ + Id : [\mathcal{A},\mathcal{A}] \longrightarrow [\mathcal{A},\mathcal{A}]$, and establish below that the arrows in square (5.2) are coalgebra homomorphisms. To do this, we need to first specify the coalgebra structures on the objects in (5.2). For S and T, we of course use β and α, respectively. For SS, we use the following coalgebra structure

$$SS \xrightarrow{\beta*\beta} (HS + Id)^2 = HS(HS + Id) + HS + Id \xrightarrow{[HS\xi, H\eta^S S] + Id} HSS + Id$$

It is easily established that $\mu^S : SS \longrightarrow S$ is a coalgebra homomorphism. Indeed, the following diagram commutes:



The left-hand part commutes since β is a monad morphism and the right-hand one is the definition of ν (use that $H\mu^S \cdot H\eta S = 1_{HS}$). Similarly, there is a coalgebra structure on TT such that $\mu : TT \longrightarrow T$ is a

coalgebra homomorphism. Finally, we show that $h * h : SS \longrightarrow TT$ is a coalgebra homomorphism. In fact, the following diagram commutes:

$$\begin{array}{ccccc}
SS & \xrightarrow{\beta*\beta} & (HS+Id)^2 = HS(HS+Id)+HS+Id & \xrightarrow{[HS\xi_{(S,\sigma)},H\eta^S S]+Id} & HSS+Id \\
\downarrow{\scriptstyle h*h} & & \downarrow{\scriptstyle (Hh+Id)^2=Hh(Hh+Id)+Hh+Id} & & \downarrow{\scriptstyle H(h*h)+Id} \\
TT & \xrightarrow[\alpha*\alpha]{} & (HT+Id)^2 = HT(HT+Id)+HT+Id & \xrightarrow[[HT\xi_{(T,\kappa)},H\eta T]+Id]{} & HTT+Id
\end{array}$$

The left-hand square commutes trivially. We consider the right-hand one componentwise: The right-hand component is obvious. For the middle one we remove H and obtain the following commutative square:

$$\begin{array}{ccc}
S & \xrightarrow{\eta^S S} & SS \\
\downarrow{\scriptstyle h} & & \downarrow{\scriptstyle h*h} \\
T & \xrightarrow[\eta T]{} & TT
\end{array}$$

Finally, for the left-hand component we remove H again to obtain

$$\begin{array}{ccc}
S(HS+Id) & \xrightarrow{S\xi_{(S,\sigma)}} & SS \\
\downarrow{\scriptstyle h*(Hh+Id)} & & \downarrow{\scriptstyle h*h} \\
T(HT+Id) & \xrightarrow[T\xi_{(T,\kappa)}]{} & TT
\end{array}$$

By naturality, it suffices to check that the square

$$\begin{array}{ccc}
HS+Id & \xrightarrow{\xi_{(S,\sigma)}} & S \\
\downarrow{\scriptstyle Hh+Id} & & \downarrow{\scriptstyle h} \\
HT+Id & \xrightarrow[\xi_{(T,\kappa)}]{} & T
\end{array}$$

commutes. Notice that we are not entitled to use naturality of ξ here, since we do not yet know that h is a monad morphism. Instead, we invoke the finality of T and show that all arrows in the above square are coalgebra homomorphisms for the functor $H \cdot \_ + Id$ on $[\mathcal{A}, \mathcal{A}]$. Indeed, since h is a coalgebra homomorphism, so is $Hh + Id$, and the other two morphisms are coalgebra homomorphisms by Lemma 5.3. This completes the proof. □

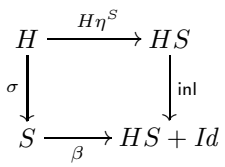
5.2. T **Gives a Final Coalgebra as a Completely Iterative Monad.** The next result is the main technical tool for our treatment of recursive program schemes in Section 6 below. Recall that we denote by $H/\mathbf{CIM}(\mathcal{A})$ the category whose objects are the pairs (S, σ), where S is completely iterative and σ is an ideal natural transformation; the morphisms in $H/\mathbf{CIM}(\mathcal{A})$ are given by the ideal monad morphisms. So $H/\mathbf{CIM}(\mathcal{A})$ is a subcategory of $H/\mathbf{Mon}(\mathcal{A})$. We show in Corollary 5.5 just below that $((T, \kappa), \alpha)$ is a final coalgebra for the same functor $\mathcal{H}$ that we have been working with. In the language of Theorem 5.4, the main point of the corollary is that if $\beta : S \longrightarrow HS + Id$ is an *ideal* monad morphism which is a coalgebra for $\mathcal{H}$ from some completely iterative monad, then the morphism into T may be taken to be an ideal monad morphism as well.

Corollary 5.5. *$\mathcal{H}$ restricts to an endofunctor on $H/\mathbf{CIM}(\mathcal{A})$, and $((T, \kappa), \alpha)$ is a final $\mathcal{H}$-coalgebra in $H/\mathbf{CIM}(\mathcal{A})$.*

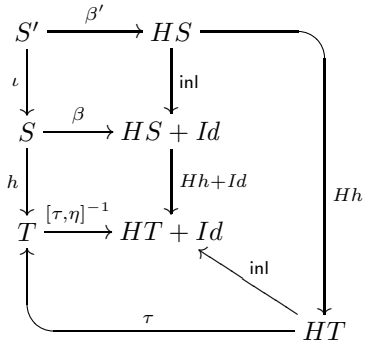
Proof. Lemma 3.5 of [Mi$_1$] gives a result propagating the complete iterative structure of monads: for any completely iterative monad S with ideal $\iota : S' \hookrightarrow S$, and any natural transformation $\sigma : H \longrightarrow S$ such that $\sigma = \iota \cdot \sigma'$ for some $\sigma' : H \longrightarrow S'$, the monad $HS + Id$ is completely iterative, too. Moreover, for any monad morphism h, $\mathcal{H}(h) = Hh + Id$ is an ideal monad morphism. Hence, $\mathcal{H}$ restricts to an endofunctor on $H/\mathbf{CIM}(\mathcal{A})$, which by abuse of notation we denote by $\mathcal{H}$ again.

Observe that (T, κ) lies in $H/\mathbf{CIM}(\mathcal{A})$ and that the coalgebra structure $\alpha = [\tau, \eta]^{-1}$ is an ideal monad morphism. Now suppose that $\beta : (S, \sigma) \longrightarrow (HS + Id, \mathsf{inl} \cdot H\eta^S)$ is an $\mathcal{H}$-coalgebra, i. e., β is an ideal monad

morphism between completely iterative monads such that the following square



commutes. By Theorem 5.4, we obtain a unique coalgebra homomorphism $h : (S, \sigma) \longrightarrow (T, \kappa)$ in $H/\mathbf{Mon}(\mathcal{A})$. Our only task is to check that h is an ideal monad morphism, so that it is a morphism in $H/\mathbf{CIM}(\mathcal{A})$. In fact, consider the following commutative diagram:



The middle square commutes since h is a coalgebra homomorphism, the upper one since β is an ideal monad morphism. The two other parts are obvious. Hence the outer shape commutes, proving that $Hh \cdot \beta' : S' \longrightarrow HT$ is a restriction of h to the ideal S' of S. □

6. Uninterpreted Recursive Program Schemes

In the classical treatment of recursive program schemes one gives an uninterpreted semantics to systems like (1.1) which are in *Greibach normal form*, i. e., every tree on the right-hand side of the system has as its head symbol a symbol from the given signature Σ. The semantics assigns to each of the new operation symbols a Σ-tree. These trees are obtained as the result of unfolding the recursive specification of the RPS. We illustrated this with an example in (1.7) in the Introduction.

We have seen in Section 2 that Σ-trees are the carrier of a final coalgebra of the signature functor H_Σ. It is the universal property of this final coalgebra which allows one to give a semantics to the given RPS. Using the technical tools developed in Section 5 we will now provide a conceptually easy and general way to give an uninterpreted semantics to recursive program schemes considered more abstractly as natural transformations, see our discussion in Section 2.3.

But before we do this, we need to say what a solution of an RPS should be. To do this we use the universal property of the monads $\mathcal{T}(H)$ as presented in Section 4. It gives an abstract version of second-order substitution.

Here are our central definitions, generalizing recursive program schemes from signatures to completely iterative monads.

Definition 6.1. Let V and H be endofunctors on $\mathcal{A}$. A *recursive program scheme* (or *RPS*, for short) is a natural transformation

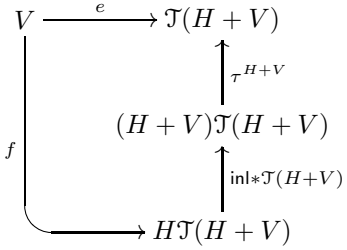
$$e : V \longrightarrow \mathcal{T}(H + V) .$$

We sometimes call V the *variables*, and H the *givens*.

The RPS e is called *guarded* if there exists a natural transformation

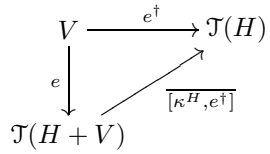
$$f : V \longrightarrow H\mathcal{T}(H + V)$$

such that the following diagram commutes:



(6.1)

A *solution* of e is an ideal natural transformation $e^\dagger : V \longrightarrow \mathcal{T}(H)$ such that the following triangle commutes:



(6.2)

Remark 6.2. Recall that $\overline{[\kappa^H, e^\dagger]}$ is the unique ideal monad morphism extending $\sigma = [\kappa^H, e^\dagger] : H + V \longrightarrow \mathcal{T}(H)$. Observe that therefore it is important to require that $e^\dagger$ be an ideal natural transformation since otherwise $\overline{\sigma}$ is not defined.

Remark 6.3.
(i) From Section 2.3, our definition is a generalization of the classical notion of RPS (to the category theoretic setting), and it extends the classical work as well by allowing infinite trees on the right-hand sides of equations.
(ii) Our notion of guardedness captures precisely the requirement that all right-hand sides of (2.3) have their root labelled by a symbol from the givens Σ. In the classical treatment of RPS this is precisely what is called Greibach normal form of an RPS, see [C].
(iii) Suppose that $H = H_\Sigma$ and $V = H_\Phi$ are signature functors of Set, and consider the recursive program scheme $e : H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$ as a set of formal equations as in (2.3). Then for any set X of syntactic variables the X-component $e^\dagger_X : H_\Phi X \longrightarrow T_\Sigma X$ of a solution assigns to any flat tree $(f, x_1, \ldots, x_n) = (f, \vec{x})$ from $H_\Phi X$ a Σ-tree over X. The commutativity of the triangle (6.2) gives the following property of solutions: apply to the right-hand side $t^f(\vec{x})$ of $f(\vec{x})$ in the given RPS the second-order substitution that replaces each operation symbol of Φ by its solution, and each operation symbol of Σ by itself—this is the action of $\overline{[\kappa^H, e^\dagger]}_X$. The resulting tree in $T_\Sigma X$ is the same tree as $e^\dagger_X(f, \vec{x})$.

Example 6.4. Let us now present two classical RPS as well as an example of RPS which is not captured with the classical setting.

(i) Recall from the Introduction the formal equations (1.1) and the ubiquitous (1.2) defining the factorial function. As explained in Example 2.8 these give rise to recursive program schemes. Thus, since both (1.1) and (1.2) are in Greibach normal form we obtain two guarded RPS's in the sense of Definition 6.1 above.
(ii) Sometimes one might wish to recursively define new operations from old ones where the new operations should satisfy certain extra properties automatically. We demonstrate this with an RPS recursively defining a new operation which is commutative. Suppose the signature Σ of givens consists of a ternary symbol F and a unary one G. Let us assume that we want to require that F satisfies the equation $F(x, y, z) = F(y, x, z)$ in any interpretation. This is modelled by the endofunctor $HX = X^3/\!\sim + X$ where $\sim$ is the smallest equivalence on X^3 with $(x, y, z) \sim (y, x, z)$. To be an H-algebra is equivalent to being an algebra A with a unary operation G_A and a ternary one F_A satisfying $F_A(x, y, z) = F_A(y, x, z)$. Suppose that one wants to define a commutative binary operation φ by the formal equation

$$\varphi(x, y) \approx F(x, y, \varphi(Gx, Gy)) \,. \tag{6.3}$$

To do it we express φ by the endofunctor V assigning to a set X the set $\{\, \{\, x, y \,\} \mid x, y \in X \,\}$ of unordered pairs of X. It is not difficult to see that the formal equation (6.3) gives rise to a guarded RPS $e : V \longrightarrow \mathcal{T}(H + V)$. In fact, to see the naturality use the description of the terminal coalgebra $\mathcal{T}(H + V)Y$ given in [AM], see Example 2.5(i). Notice that in the classical setting we are unable to demand that (the solution of) φ be a commutative operation at this stage: one would use general

facts to obtain a unique solution, and then one would need to devise a special argument to verify commutativity of that solution. Once again, our general theory insures that any solution of our RPS will be commutative.

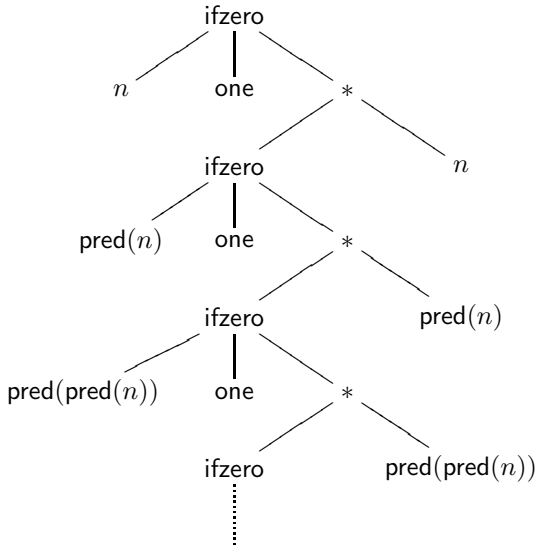
The main result of this section is the following theorem. Before we present its proof below let us illustrate the result with a few examples.

Theorem 6.5. *Every guarded recursive program scheme has a unique solution.*

Examples 6.6. We present here the solutions of the RPS's of Example 6.4.

(i) The unique solution of the RPS induced by the equations (1.1) is an ideal natural transformation $e^\dagger : H_\Phi \longrightarrow \mathcal{T}(H_\Sigma)$. Equivalently, we have a natural transformation $\Phi \longrightarrow T_\Sigma \cdot J$, see Section 2.3. That is, the solution $e^\dagger$ is essentially given by two Σ-trees (one for each of the operation symbols φ and ψ) over a singleton set, say $\{ x \}$. It follows from the proof of Theorem 6.5 that those Σ-trees are the ones given in (1.6), see Example 6.15 below.

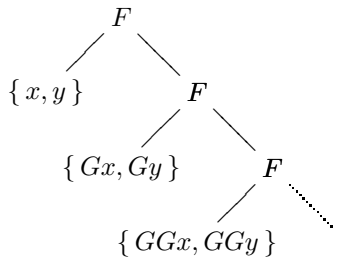
Similarly, the unique solution of the RPS induced by the equation (1.2) is essentially given by the Σ-tree over the set $\{ n \}$ below:



(6.4)

Notice that the nodes labelled by a term correspond to appropriately labelled finite subtrees.

(ii) We continue example 6.4(ii) and describe the uninterpreted solution of the guarded RPS e arising from the formal equation (6.3) defining a commutative operation. The components of $e^\dagger_X : VX \longrightarrow \mathcal{T}(H)X$ assign to an unordered pair $\{ x, y \}$ in VX the tree



where for every node labelled by F the order of the first two children cannot be distinguished; we indicate this with set-brackets in the picture above.

Remark 6.7. Notice that in the classical setting not every recursive program scheme which has a unique solution needs to be in Greibach normal form. For example, consider the system formed by the first equation in (1.1) and by the equation $\psi(x) \approx \varphi(\psi(x))$. This system gives rise to an unguarded RPS. Thus, Theorem 6.5 does not provide a solution of this RPS. However, the system is easily rewritten to an equivalent one in Greibach normal form which gives a guarded RPS that we can uniquely solve using our Theorem 6.5.

The rest of this section is devoted to the proof of Theorem 6.5. We illustrate each crucial step with the help of our examples. Before we turn to the proof of the main theorem we need to establish some preliminary lemmas.

Lemma 6.8. *Let H and K be endofunctors on $\mathcal{A}$. Suppose we have objects (S, σ) of $H/\mathbf{Mon}(\mathcal{A})$ and (R, ρ) of $K/\mathbf{Mon}(\mathcal{A})$. Let $n : H \longrightarrow K$ be a natural transformation and let $m : S \longrightarrow R$ be a monad morphism such that the following square*

$$\begin{array}{ccc} H & \xrightarrow{n} & K \\ {\scriptstyle\sigma}\downarrow & & \downarrow{\scriptstyle\rho} \\ S & \xrightarrow[m]{} & R \end{array}$$

commutes. Then $n * m + Id : HS + Id \longrightarrow KR + Id$ is a monad morphism such that the following square

$$\begin{array}{ccc} H & \xrightarrow{\quad n \quad} & K \\ {\scriptstyle \mathsf{inl}\cdot H\eta^S}\downarrow & & \downarrow{\scriptstyle \mathsf{inl}\cdot K\eta^R} \\ HS + Id & \xrightarrow[n*m+Id]{} & KR + Id \end{array}$$

commutes.

Proof. Naturality and the unit law are clear, and the preservation of the monad multiplication is a straightforward diagram chasing argument which we leave to the reader. □

Lemma 6.9. *Consider any guarded RPS e, see (6.1). There exists a unique (ideal) monad morphism*

$$\widehat{e} : \mathcal{T}(H + V) \longrightarrow \mathcal{T}(H + V)$$

such that the following triangle commutes:

$$\begin{array}{ccc} H + V & \xrightarrow{\kappa^{H+V}} & \mathcal{T}(H + V) \\ & {\scriptstyle [\kappa^{H+V}\cdot \mathsf{inl}, e]}\searrow & \downarrow{\scriptstyle \widehat{e}} \\ & & \mathcal{T}(H + V) \end{array}$$

There is also a unique (ideal) monad morphism

$$\overline{e} : \mathcal{T}(H + V) \longrightarrow H\mathcal{T}(H + V) + Id$$

such that the following diagram commutes:

$$\begin{array}{ccc} H + V & \xrightarrow{\kappa^{H+V}} & \mathcal{T}(H + V) \\ {\scriptstyle [H\eta^{H+V}, f]}\downarrow & & \downarrow{\scriptstyle \overline{e}} \\ H\mathcal{T}(H + V) & \xrightarrow[\mathsf{inl}]{} & H\mathcal{T}(H + V) + Id \end{array} \tag{6.5}$$

Proof. We get $\widehat{e}$ from Theorem 4.5. Indeed, it is easily checked that

$$[\kappa^{H+V} \cdot \mathsf{inl}, e] : H + V \longrightarrow \mathcal{T}(H + V)$$

is an ideal natural transformation using that e is guarded. As for $\overline{e}$, recall that H "embeds" into $\mathcal{T}(H + V)$ via the natural transformation

$$H \xrightarrow{\mathsf{inl}} H + V \xrightarrow{\kappa^{H+V}} \mathcal{T}(H + V)\,.$$

Since κ^{H+V} is an ideal natural transformation so is this one. Hence, $(\mathcal{T}(H + V), \kappa^{H+V} \cdot \mathsf{inl})$ is an object of $H/\mathbf{CIM}(\mathcal{A})$ and so it follows from Corollary 5.5 that $H\mathcal{T}(H + V) + Id$ carries the structure of a completely iterative monad. The natural transformation $\mathsf{inl} \cdot [H\eta^{H+V}, f] : H + V \longrightarrow H\mathcal{T}(H + V) + Id$, see (6.5), is obviously ideal. Thus, we obtain $\overline{e}$ as desired from another application of Theorem 4.5. □

Remark 6.10. In the leading example of a classical RPS for given signatures, the formation of the morphism $\overline{e}$ corresponds to the formation of a flat system of equations, where for every tree there is a recursion variable. More precisely, suppose we have signatures Σ and Φ, and an RPS as in (2.3) which is in Greibach normal form. The component of $\overline{e}$ at some set Z of syntactic variables can be seen as a set of formal equations. Here is a description of $\overline{e}_Z$: For every tree $t \in T_{\Sigma+\Phi}Z$, we have a recursion variable $\underline{t}$. And for each recursion variable we have one formal equation:

$$\begin{array}{lcll} \underline{t} & \approx & x & \text{if } t \text{ is a single node tree with root label } x \in Z, \text{ or} \\ \underline{t} & \approx & \sigma(\underline{t_1}, \ldots, \underline{t_n}) & \text{for some } n \in \mathbb{N} \text{ and some } \sigma \in \Sigma_n \text{ otherwise,} \end{array}$$

where the tree $s = \sigma(t_1, \ldots, t_n)$ is the result of the following second-order substitution applied to t: every symbol of Φ is substituted by its right-hand side in the given RPS, and every symbol of Σ by itself. Since the given RPS is guarded the head symbol of s is a symbol of Σ for all trees t.

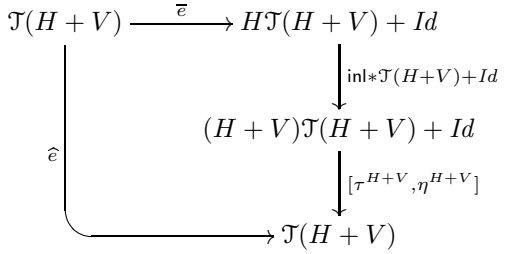
Observe that forming the right-hand sides of this system corresponds to the application of one step of Kleene's computation rule, see [G].

Example 6.11. For the guarded RPS of (1.1) the flat system obtained from $\overline{e}$ for $Z = \{ x \}$ includes the equations of the system (1.7) from the Introduction.

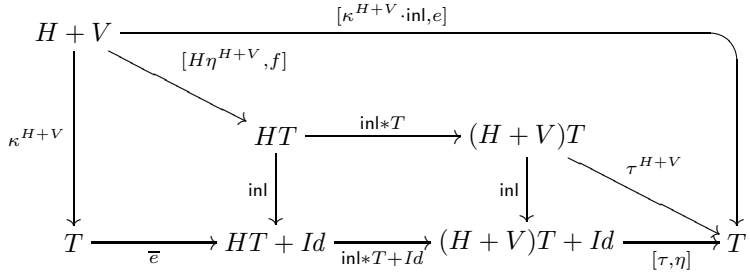
We also give a fragment of the flat system obtained as the extension of the RPS (1.2), see also Example 6.4(i) . Here the set of syntactic variables is $Z = \{n\}$ and the formal equations described by $\overline{e}_Z$ include the following ones:

$$\begin{aligned}
\underline{f(n)} &\approx \mathsf{ifzero}(\underline{n}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(n)) * n}) \\
\underline{n} &\approx n \\
\underline{\mathsf{one}} &\approx \mathsf{one} \\
\underline{f(\mathsf{pred}(n)) * n} &\approx \underline{\mathsf{ifzero}(\mathsf{pred}(n), \mathsf{one}, f(\mathsf{pred}(\mathsf{pred}(n)))) * \underline{n}} \\
\underline{f(\mathsf{pred}(n))} &\approx \mathsf{ifzero}(\underline{\mathsf{pred}(n)}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(\mathsf{pred}(n)) * \mathsf{pred}(n))}) \\
&\vdots
\end{aligned} \tag{6.6}$$

Lemma 6.12. *The following diagram commutes:*



Proof. By the first part of Lemma 6.9, it suffices to show that the composite in the above diagram is an ideal monad morphism extending $[\kappa^{H+V} \cdot \mathsf{inl}, e]$. For the extension property, consider the following commutative diagram, where we write T for $\mathcal{T}(H+V)$:



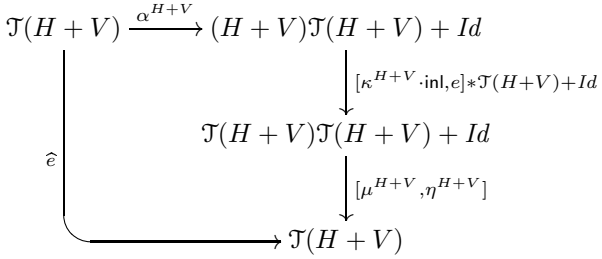
The left-hand part commutes by Lemma 6.9. For the left-hand component of the upper part notice that

$$\tau^{H+V} \cdot (\mathsf{inl} * T) \cdot H\eta^{H+V} = \tau^{H+V} \cdot (H+V)\eta^{H+V} \cdot \mathsf{inl} = \kappa^{H+V} \cdot \mathsf{inl}\,.$$

The right-hand component of this part commutes since e is guarded, see (6.1), and the remaining parts are trivial.

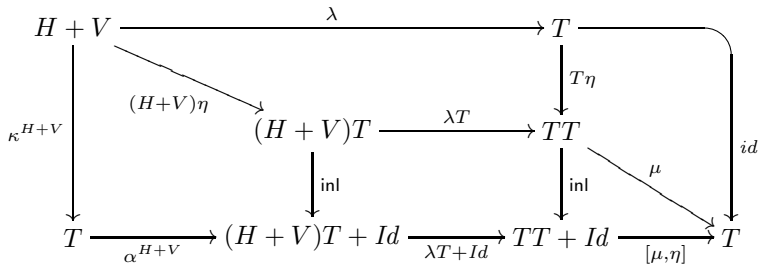
We show that all parts of the lower edge in the above diagram are monad morphisms. For $\overline{e}$, see Lemma 6.9. For $\mathsf{inl} * T + Id$, apply Lemma 6.8 to $n = \mathsf{inl}$ and $m = 1_T$. And the for the last part, $[\tau, \eta] = [\mu \cdot \kappa T, \eta]$, notice that it is the component at $(\mathcal{T}(H+V), \kappa^{H+V})$ of the natural transformation ξ of Lemma 5.2 applied to $(H+V)/\mathbf{Mon}(\mathcal{A})$. □

Lemma 6.13. *The following diagram*



commutes.

Proof. By the first part of Lemma 6.9, it suffices to show that the composite in the above diagram is an ideal monad morphism extending $[\kappa^{H+V} \cdot \mathsf{inl}, e]$. Let us write λ for $[\kappa^{H+V} \cdot \mathsf{inl}, e]$ and T for $\mathcal{T}(H+V)$. Now consider the following commutative diagram



which establishes the extension part. To see that the morphism of the lower edge is a monad morphism, recall from Theorem 5.4 that α^{H+V} is the structure map of a final coalgebra of a functor on $(H+V)/\mathbf{Mon}(\mathcal{A})$, whence a monad morphism, and $[\mu \cdot \lambda T, \eta]$ is the component at (T, λ) of the natural transformation ξ, see Lemma 5.2. □

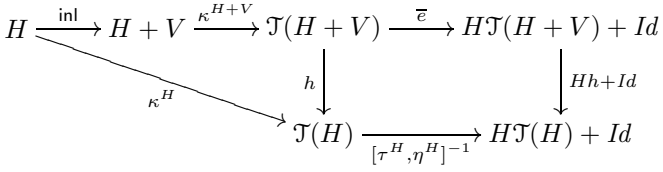
Proof of Theorem 6.5. Consider $\overline{e}$ from Lemma 6.9. It is a coalgebra structure for the functor

$$\mathcal{H} : H/\mathbf{CIM}(\mathcal{A}) \longrightarrow H/\mathbf{CIM}(\mathcal{A})$$

In fact, $\overline{e}$ is a morphism in $H/\mathbf{CIM}(\mathcal{A})$; it is an ideal monad morphism and by (6.5) we have

$$\overline{e} \cdot \kappa^{H+V} \cdot \mathsf{inl} = \mathsf{inl} \cdot [H\eta^{H+V}, f] \cdot \mathsf{inl} = \mathsf{inl} \cdot H\eta^{H+V}\,. \tag{6.7}$$

Now apply Corollary 5.5 to obtain a unique $\mathcal{H}$-coalgebra homomorphism from the above coalgebra $\overline{e}$ to the final one. In more detail, we obtain a unique ideal monad morphism $h : \mathcal{T}(H+V) \longrightarrow \mathcal{T}(H)$ such that the following diagram commutes:



(6.8)

Now let

$$e^\dagger \equiv V \xrightarrow{\mathsf{inr}} H+V \xrightarrow{\kappa^{H+V}} \mathcal{T}(H+V) \xrightarrow{h} \mathcal{T}(H)\,. \tag{6.9}$$

We shall prove that $e^\dagger$ uniquely solves e.

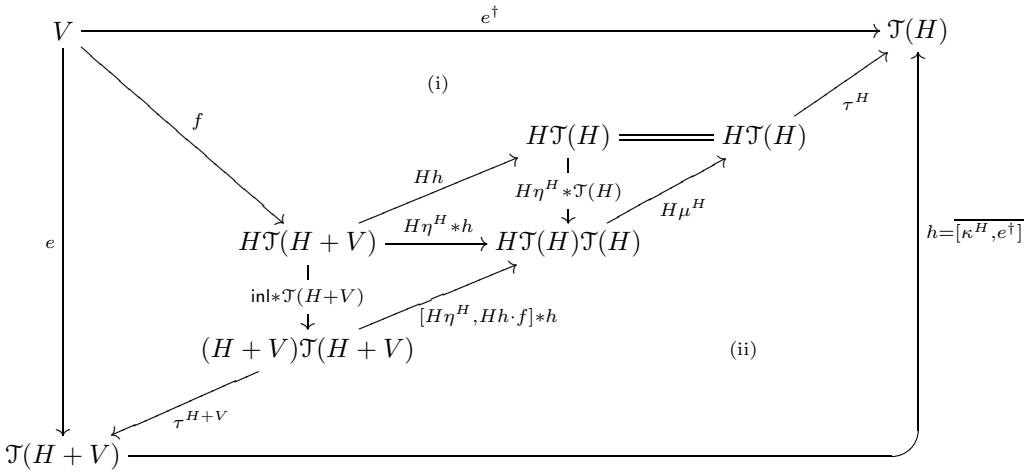
(a) $e^\dagger$ is a solution of e. Since h is an ideal monad morphism and κ^{H+V} is an ideal natural transformation, we see that $e^\dagger$ is an ideal natural transformation. Next observe that by definition we have

$$h = \overline{[\kappa^H, e^\dagger]},$$

and we also get

$$\begin{aligned} e^\dagger &= h \cdot \kappa^{H+V} \cdot \mathsf{inr} && \text{(by (6.9))} \\ &= [\tau^H, \eta^H] \cdot (Hh + Id) \cdot \overline{e} \cdot \kappa^{H+V} \cdot \mathsf{inr} && \text{(by (6.8))} \\ &= [\tau^H, \eta^H] \cdot (Hh + Id) \cdot \mathsf{inl} \cdot f && \text{(by (6.5))} \\ &= \tau^H \cdot Hh \cdot f. \end{aligned} \tag{6.10}$$

Now we can conclude that the following diagram commutes:



(6.11)

Indeed, part (i) commutes by (6.10). For part (ii) observe first that from (3.7) and (6.10) we get the equation

$$[\kappa^H, e^\dagger] \equiv H + V \xrightarrow{[H\eta^H, Hh \cdot f]} H\mathcal{T}(H) \xrightarrow{\tau^H} \mathcal{T}(H)\,. \tag{6.12}$$

Now apply Lemma 4.6 to $H+V$ and H and $\sigma = [\kappa^H, e^\dagger]$ using that (6.12) induces the ideal monad morphism $\overline{[\kappa^H, e^\dagger]} = h$. The other parts of (6.11) are obvious.

(b) Uniqueness of solutions. Suppose that $s : V \longrightarrow \mathcal{T}(H)$ is a solution of e. Since solutions are ideal natural transformations by definition, there exists a natural transformation $s' : V \longrightarrow H\mathcal{T}(H)$ such that $s = \tau^H \cdot s'$. We shall show below that the ideal monad morphism h from (6.8) is equal to

$$\overline{[\kappa^H, s]} : \mathcal{T}(H+V) \longrightarrow \mathcal{T}(H) \tag{6.13}$$

using coinduction, i. e., we show that $\overline{[\kappa^H, s]}$ is a coalgebra homomorphism. Then, since $(\mathcal{T}(H), \kappa^H)$ is a final $\mathcal{H}$-coalgebra, we can conclude that $h = \overline{[\kappa^H, s]}$ and therefore

$$e^\dagger = \overline{[\kappa^H, e^\dagger]} \cdot e = h \cdot e = \overline{[\kappa^H, s]} \cdot e = s,$$

where the last equality holds since s is a solution of e.

For the coinduction argument, replace h in Diagram (6.8) by $\overline{[\kappa^H, s]}$ and check that the modified diagram commutes. In fact, for the left-hand triangle we obtain:

$$\overline{[\kappa^H, s]} \cdot \kappa^{H+V} \cdot \mathsf{inl} = [\kappa^H, s] \cdot \mathsf{inl} = \kappa^H\,. \tag{6.14}$$

To verify the modified version of the right-hand square of (6.8), use the freeness of the completely iterative monad $\mathcal{T}(H+V)$. Thus, it is sufficient that this diagram of ideal monad morphisms commutes when precomposed with the universal arrow κ^{H+V}. Furthermore we consider the components of the coproduct $H+V$ separately. Let us write x as a short notation for $\overline{[\kappa^H, s]}$. Then, for the left-hand coproduct component we obtain the following equations:

$$\begin{aligned}
&[\tau^H, \eta^H] \cdot (Hx + Id) \cdot \overline{e} \cdot \kappa^{H+V} \cdot \mathsf{inl} \\
&\quad = [\tau^H, \eta^H] \cdot (Hx + Id) \cdot \mathsf{inl} \cdot H\eta^{H+V} && \text{(see (6.7))} \\
&\quad = \tau^H \cdot Hx \cdot H\eta^{H+V} \\
&\quad = \tau^H \cdot H\eta^H && \text{(since } x \cdot \eta^{H+V} = \eta^H\text{)} \\
&\quad = \kappa^H && \text{(see (3.7))} \\
&\quad = x \cdot \kappa^{H+V} \cdot \mathsf{inl} && \text{(see (6.14))}
\end{aligned}$$

In order to prove the right-hand component commutative we use a diagram similar to Diagram (6.11). Just replace in (6.11) $Hh \cdot f$ by s', all other occurrences of h by x, and $e^\dagger$ by s. We prove that part (i) in this modified diagram commutes. In fact, this follows from the fact that all other parts and the outward square commute: the outward square commutes since s is a solution of e; part (ii) in the modified diagram commutes by Lemma 4.6 again, and all other parts are clear. Thus, we proved that the following equation holds:

$$s = \tau^H \cdot Hx \cdot f. \tag{6.15}$$

From this we obtain the equations

$$\begin{aligned}
&[\tau^H, \eta^H] \cdot (Hx + Id) \cdot \overline{e} \cdot \kappa^{H+V} \cdot \mathsf{inr} \\
&\quad = \tau^H \cdot Hx \cdot f && \text{(similar as in (6.10))} \\
&\quad = s && \text{(by (6.15))} \\
&\quad = x \cdot \kappa^{H+V} \cdot \mathsf{inr} && \text{(since } x = \overline{[\kappa^H, s]}\text{)}\,.
\end{aligned}$$

This establishes that h from (6.9) is equal to (6.13). □

Remark 6.14. Recall that the formation of $\overline{e}$ corresponds, in the leading example of an RPS for given signatures, to the formation of a flat system of equations, see Remark 6.10. Now the map h_Z of (6.8) assigns to every variable $\underline{t} \in T_{\Sigma+\Phi}Z$ of the flat system the Σ-tree given by unfolding the recursive specification given by this flat system, i. e., h_Z is the unique solution of the flat equation morphism $\overline{e}_Z$ in the cia $T_\Sigma Z$.

Example 6.15. In equations (1.1) in the beginning of this paper, we introduced a guarded RPS as they are classically presented. It induces an RPS (in our sense), as discussed in Example 6.4(i). The unique solution of the flat equation morphism $\overline{e}_Z$ corresponding to the flat system described in Example 6.11 is $h_Z : T_{\Sigma+\Phi}Z \longrightarrow T_\Sigma Z$ (see also (1.7) from the Introduction). The definition of $e^\dagger$ in (6.9) means that we only consider the solution for the recursion variables $\underline{\varphi(x)}$ and $\underline{\psi(x)}$ in that system. These solutions are precisely the Σ-trees described in Example 6.6(i).

Similarly, for the guarded RPS induced by (1.2) the solution is obtained by considering only the unique solution for the variable $\underline{f(x)}$ of the flat system (6.6). Clearly, this yields the desired tree (6.4).

7. Interpreted Recursive Program Schemes

We have seen in the previous section that for every guarded recursive program scheme we can find a unique uninterpreted solution. In practice, however, one is more interested in finding *interpreted* solutions. In the classical treatment of recursive program schemes, this means that a recursive program scheme defining new operation symbols of a signature Φ from given ones in a signature Σ comes together with some Σ-algebra A. An interpreted solution of the recursive program scheme in question is, then, an operation on A for each operation symbol in Φ such that the formal equations of the RPS become valid identities in A.

Of course, in general an algebra A will not admit interpreted solutions. We shall prove in this section that any Elgot algebra $(A, a, (\_)^*)$ as defined in Section 3 admits an interpreted solution of any guarded recursive program scheme. Moreover, if A is a completely iterative algebra, interpreted solutions are unique. We also define the notion of a *standard* interpreted solution and prove that uninterpreted solutions and standard interpreted ones are consistent with one another as expected. This is a fundamental result for algebraic semantics.

We turn to applications after proving our main results. In Subsection 7.1 we study the computation tree semantics of RPS's arising from the computation tree Elgot algebras of Section 3.3. Then, in Subsection 7.2 we prove that in the category CPO our interpreted program scheme solutions agree with the usual denotational semantics obtained by computing least fixed points. Similarly, we show in Subsection 7.3 for the category of CMS that our solutions are the same as the ones computed using Banach's Fixed Point Theorem. Furthermore, we present new examples of recursive program scheme solutions pertaining to fractal self-similarity. We are not aware of any previous work connecting recursion to implicitly defined sets.

Definition 7.1. Let $(A, a, (\_)^*)$ be an Elgot algebra for H and let $e : V \longrightarrow \mathcal{T}(H + V)$ be an RPS. An *interpreted solution* of e in A is a structure of a V-algebra $e^\ddagger_A : VA \longrightarrow A$, such that

(i) the $(H + V)$-algebra $[a, e^\ddagger_A] : (H + V)A \longrightarrow A$ is the structure morphism of an Elgot algebra $(A, [a, e^\ddagger_A], (\_)^+)$ for $H + V$; and
(ii) the triangle below

$$\begin{array}{ccc} VA & \xrightarrow{\;e^\ddagger_A\;} & A \\ {\scriptstyle e_A}\downarrow & \nearrow_{[a,e^\ddagger_A]} & \\ \mathcal{T}(H+V)A & & \end{array} \tag{7.1}$$

commutes, where the diagonal arrow denotes the evaluation morphism associated to the Elgot algebra in (i), see Theorem 3.14.

Remark 7.2.

(i) The subscript A in $e^\dagger_A$ is only present to remind us of the codomain A. That is, $e^\dagger_A$ is not a component of any natural transformation.

(ii) Recall from Corollary 3.15 that the triangle

$$\begin{array}{ccc} (H+V)A & \xrightarrow{\kappa_A^{H+V}} & \mathcal{T}(H+V)A \\ & \searrow^{[a,e^\dagger_A]} & \downarrow{\scriptstyle \widetilde{[a,e^\dagger_A]}} \\ & & A \end{array} \tag{7.2}$$

commutes. It follows that for an interpreted solution $e^\dagger_A$, we have

$$a = \widetilde{[a, e^\dagger_A]} \cdot \kappa_A^{H+V} \cdot \mathsf{inl}, \tag{7.3}$$

(iii) In our leading example where $H = H_\Sigma$ and $V = H_\Phi$ are signature functors on Set, the commutativity of (7.1) states precisely that an interpreted solution provides operations on A which turn the formal equations of the given recursive program scheme into actual identities. More precisely, suppose that e is a recursive program scheme given by formal equations (2.3). The interpreted solution $e^\dagger_A$ gives for each n-ary operation symbol f of the signature Φ an operation $f_A : A^n \longrightarrow A$. And the commutativity of (7.1) gives the following property of f_A: take for any $\vec{a} \in A^n$ the right-hand side $t^f(\vec{a})$ in the given recursive program scheme, then evaluate $t^f(\vec{a})$ in A using the given operations for Σ and the ones provided by $e^\dagger_A$ for Φ on A—this is the action of $\widetilde{[a, e^\dagger_A]}$. The resulting element of A is the same as $f_A(\vec{a})$.

(iv) The requirement that $[a, e^\dagger_A]$ be the structure morphism of an Elgot algebra may seem odd at first. However, we need to assume this in order to be able to use $\widetilde{[a, e^\dagger_A]}$ in (7.1). Furthermore, the requirement has a clear practical advantage: operations defined recursively by means of an interpreted solution of an RPS may be used in subsequent recursive definitions. For example, for the signature functors on Set as in (iii) above the Elgot algebra with structure map $[a, e^\dagger_A]$ has operations for all operation symbols of $\Sigma + \Phi$. Thus, it can be used as an interpretation of givens for any further recursive program scheme with signature $\Sigma + \Phi$ of givens.

Theorem 7.3. *Let $(A, a, (\_)^*)$ be an Elgot algebra for H and let $e : V \longrightarrow \mathcal{T}(H + V)$ be a guarded RPS. Then the following hold:*

(i) *there exists an interpreted solution $e^\dagger_A$ of e in A,*

(ii) *if A is a completely iterative algebra, then $e^\dagger_A$ is the unique interpreted solution of e in A.*

We will present the proof after a technical lemma. It follows from the proof of Theorem 7.3 that uninterpreted solutions correspond to certain interpreted ones in a canonical way. We shall make this precise at the end of this subsection, and prove what could be called "Fundamental Theorem of Algebraic Semantics".

Lemma 7.4. *Let $(A, a, (\_)^*)$ be an Elgot algebra, let e be a guarded RPS, and let $\widehat{e}$ be as in Lemma 6.9. Then the following are in one-to-one correspondence:*

(i) *the interpreted solutions $e^\dagger_A$ of e in A.*

(ii) *the evaluation morphisms $\beta : \mathcal{T}(H + V)A \longrightarrow A$ such that the two diagrams*

$$\begin{array}{ccccc} HA & \xrightarrow{\mathsf{inl}_A} & (H+V)A & \xrightarrow{\kappa_A^{H+V}} & \mathcal{T}(H+V)A \\ & \searrow^{a} & & & \downarrow{\scriptstyle \beta} \\ & & & & A \end{array} \tag{7.4}$$

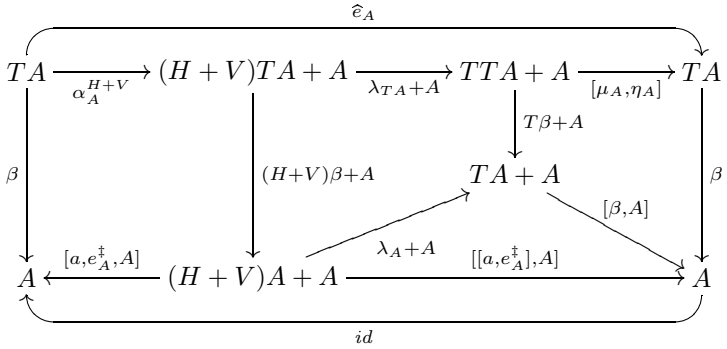
$$\begin{array}{ccc} \mathcal{T}(H+V)A & \xrightarrow{\widehat{e}_A} & \mathcal{T}(H+V)A \\ & \searrow^{\beta} & \downarrow{\scriptstyle \beta} \\ & & A \end{array} \tag{7.5}$$

commute.

In more detail, if $e^\dagger_A$ is an interpreted solution of e in A, then the evaluation morphism $\beta = \widetilde{[a, e^\dagger_A]}$ has the properties of (ii). And if β makes the diagrams in (ii) commute, then $\beta \cdot \kappa_A^{H+V} \cdot \mathsf{inr}$ is an interpreted solution to e in A. Finally, these two operations are mutually inverse.

Proof. Let us write T as a short notation for $\mathcal{T}(H+V)$ and λ for $[\kappa^{H+V} \cdot \mathsf{inl}, e] : H + V \longrightarrow T$.

(i) $\Rightarrow$ (ii): As in our statement, take the evaluation morphism $\beta = \widetilde{[a, e^\dagger_A]}$. Then (7.4) is (7.3). For (7.5), consider the following commutative diagram



The upper part commutes by Lemma 6.13, the right-hand part commutes since β is an Eilenberg-Moore algebra structure, and the left-hand part commutes since β is given as the solution of α_A^{H+V} in the Elgot algebra $(A, [a, e^\dagger_A], (\_)^+)$, see Theorem 3.13. For the middle part simply use the naturality of λ. The lowest part is obvious. Finally, for the commutativity of the lower triangle it suffices to show that the equation

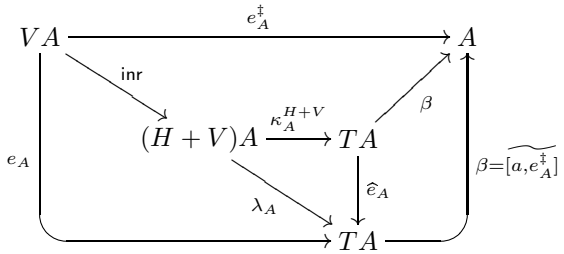
$$\beta \cdot \lambda_A = [a, e^\dagger_A]$$

holds. We consider the components separately: the left-hand one is (7.3), and the right-hand one is the triangle (7.1). Thus, the outer shape of the above diagram commutes, viz. the desired triangle (7.5).

(ii) $\Rightarrow$ (i): Define

$$e^\dagger_A \equiv VA \xrightarrow{\mathsf{inr}} (H+V)A \xrightarrow{\kappa_A^{H+V}} \mathcal{T}(H+V)A \xrightarrow{\beta} A\,. \tag{7.6}$$

It follows from the commutativity of (7.4) and Theorem 3.14 that $[a, e^\dagger_A] = \beta \cdot \kappa_A^{H+V}$ is the structure map of an Elgot algebra for $H + V$ on A so that the equation $\beta = \widetilde{[a, e^\dagger_A]}$ holds. For Diagram (7.1) consider the following commutative diagram:



The inner triangle commutes due to the definition of $\widehat{e}$, see Lemma 6.9, the right-hand one due to (7.5), and the other two parts are clear. Thus, the outer shape commutes, so we have (7.1).

Finally, we check that the operations of going from interpreted solutions $e^\dagger_A$ to evaluation morphisms β are mutually inverse. In fact, we have

$$\widetilde{[a, e^\dagger_A]} \cdot \kappa_A^{H+V} \cdot \mathsf{inr} = e^\dagger_A$$

by the commutativity of the right-hand component of Diagram (7.2). And for the interpreted solution $e^\dagger_A$ defined by (7.6), we have already seen above that the equation $\beta = \widetilde{[a, e^\dagger_A]}$ holds. This completes the proof. □

Remark 7.5. Lemma 7.4 above states that interpreted solutions correspond to suitable evaluation morphisms β such that the diagrams (7.4) and (7.5) commute. In particular, if $H = H_\Sigma$ and $V = H_\Phi$ are signature functors on Set, the existence of β means that all trees over the signature $\Sigma + \Phi$ can be evaluated in A. The commutativity of (7.4) states that for trees in $T_{\Sigma+\Phi}A$ operation symbols of Σ are interpreted according to the given Σ-algebra structure a, while the commutativity of (7.5) states that the operation symbols of Φ are interpreted by operations that satisfy the equations given by the recursive program scheme e.

Proof of Theorem 7.3. (i) Given an Elgot algebra $(A, a, (\_)^*)$ and a guarded recursive program scheme $e : V \longrightarrow \mathcal{T}(H+V)$ consider $\overline{e} : \mathcal{T}(H+V) \longrightarrow H \cdot \mathcal{T}(H+V) + Id$ from Lemma 6.9. Its component at A yields a flat equation morphism

$$g \equiv \mathcal{T}(H+V)A \longrightarrow H\mathcal{T}(H+V)A + A\,, \tag{7.7}$$

with respect to $(A, a, (\_)^*)$ and we take its solution

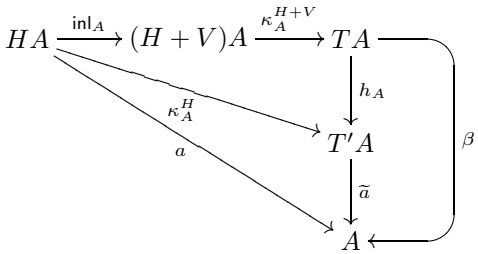
$$\beta \equiv \mathcal{T}(H+V)A \xrightarrow{g^*} A \tag{7.8}$$

By Lemma 7.4, it suffices to prove that β is an evaluation morphism such that the diagrams (7.4) and (7.5) commute. We first check that β is the structure of an Eilenberg-Moore algebra for $\mathcal{T}(H+V)$, hence an evaluation morphism. To this end we first establish the equation

$$\beta = \widetilde{a} \cdot h_A\,, \tag{7.9}$$

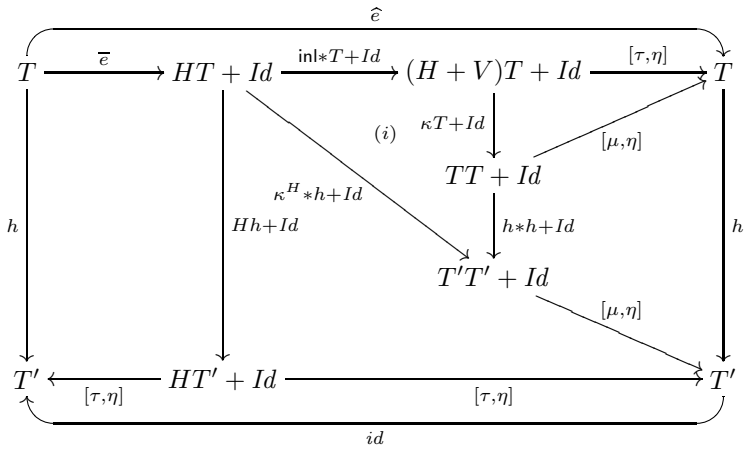
where $h : \mathcal{T}(H+V) \longrightarrow \mathcal{T}(H)$ is the monad morphism that we obtain from the recursive program scheme e. (See (6.8). It is also useful to recall that $h = \overline{[\kappa^H, e^\dagger]}$.) Recall that $\widetilde{a} = (\alpha_A)^*$, see Theorem 3.13, and that h_A is a homomorphism of $(H(\_) + A)$-coalgebras, see Diagram (6.8). Thus, by the Functoriality of $(\_)^*$, we obtain $g^* = (\alpha_A)^* \cdot h_A$, i. e., the desired equation (7.9). Now since $h : \mathcal{T}(H+V) \longrightarrow \mathcal{T}(H)$ is a monad morphism, and $\widetilde{a}$ is the structure morphism of an Eilenberg-Moore algebra for $\mathcal{T}(H)$, $\beta = \widetilde{a} \cdot h_A$ is an Eilenberg-Moore algebra for $\mathcal{T}(H+V)$. In fact, this follows from a general fact from category theory, see e. g. Proposition 4.5.9 in [Bo].

We check that the diagrams (7.4) and (7.5) commute. Let us use T as a short notation for $\mathcal{T}(H+V)$ and T' for $\mathcal{T}(H)$. In order to see that (7.4) commutes, consider the following commutative diagram



The upper part commutes due to the left-hand triangle of (6.8), the lower triangle by Corollary 3.15, and the right-hand part is (7.9).

To see that (7.5) commutes, consider the following diagram



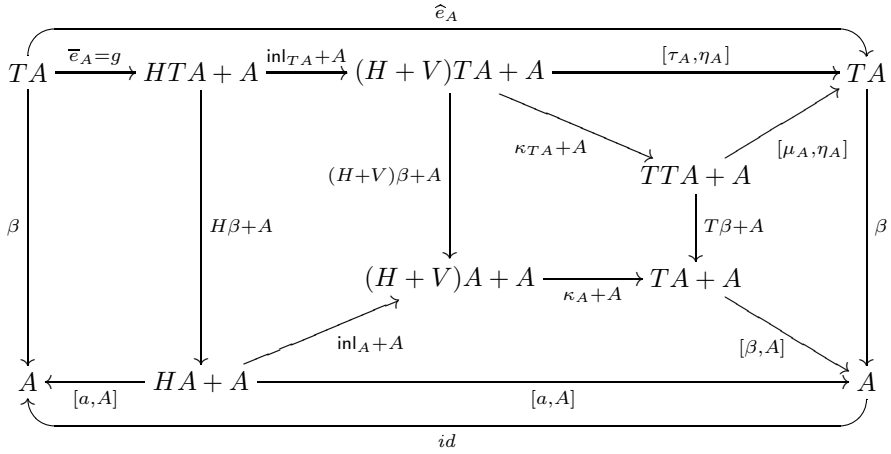
(7.10)

All of its inner parts commute. The upper part is Lemma 6.12, for the left-hand square, see (6.8), and for the right-hand part use that h is a monad morphism. That part (i) commutes follows from commutativity of the left-hand triangle of (6.8) and naturality. The remaining inner part commutes due to Corollary 3.15: $\mu \cdot \kappa T = \tau$. Thus, the outer shape of diagram (7.10) commutes. We obtain the equations

$$\begin{aligned} \beta \cdot \widehat{e}_A &= \widetilde{a} \cdot h_A \cdot \widehat{e}_A && \text{(see (7.9))} \\ &= \widetilde{a} \cdot h_A && \text{(see (7.10))} \\ &= \beta && \text{(see (7.9))} \end{aligned}$$

This completes the proof of part (i).

(ii) Let $(A, a, (\_)^*)$ be a cia. We show that the solution $e^\ddagger_A$ defined in (i) is unique. By Lemma 7.4, it suffices to prove that any evaluation morphism $\beta : \mathcal{T}(H+V)A \longrightarrow A$ for which diagrams (7.4) and (7.5) commute is a solution of g, see (7.7). To this end consider the following diagram, where we write $T = \mathcal{T}(H+V)$ for short once more:



Its outer shape commutes due to (7.5), the right-hand part since β is an Eilenberg-Moore algebra structure, the upper-right triangle follows from Corollary 3.15, and the lower right-hand part follows from (7.4). Thus, since all other parts are obviously commutative, the left-hand inner square commutes. But this shows that β is a solution of g, see (7.7). By the uniqueness of solutions, we have $\beta = g^*$. □

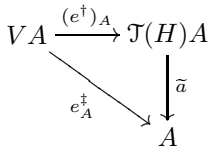
Definition 7.6. For any guarded RPS e and any Elgot algebra $(A, a, (\_)^*)$, let $e^\ddagger_A$ be the interpreted solution obtained from the proofs of Theorem 7.3 and Lemma 7.4 as stated below

$$e^\ddagger_A \equiv VA \xrightarrow{\mathsf{inr}} (H+V)A \xrightarrow{\kappa_A^{H+V}} \mathcal{T}(H+V)A \xrightarrow{g^*} A\,,$$

where g is the flat equation morphism of (7.7). We call this the *standard interpreted solution* of e in A.

Finally, we prove the "Fundamental Theorem of Algebraic Semantics", which establishes that uninterpreted and interpreted solutions are connected in the "proper way".

Theorem 7.7. *Let $(A, a, (\_)^*)$ be an Elgot algebra and consider its evaluation morphism $\widetilde{a} : \mathcal{T}(H)A \longrightarrow A$. Let e be any guarded recursive program scheme, let $e^\ddagger_A : VA \longrightarrow A$ be the standard interpreted solution of e in A of Theorem 7.3, and let $e^\dagger : V \longrightarrow \mathcal{T}(H)$ be the (uninterpreted) solution of Theorem 6.5. Then the triangle*



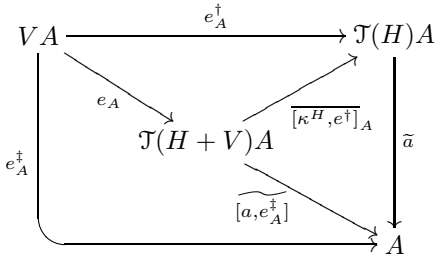
(7.11)

commutes.

Furthermore, the standard interpreted solution $e^\ddagger_A$ is uniquely determined by the commutativity of the above triangle.

Remark 7.8. Notice that $(e^\dagger)_A$ is the component at A of the natural transformation $e^\dagger$. And once again, $e^\ddagger_A$ is not the component at A of any natural transformation but merely a morphism from VA to A.

Proof. In fact, we have the commutative diagram



The lower left part commutes due to the definition of an interpreted solution, see (7.1); the top triangle is the definition of an uninterpreted solution, see (6.2); and the triangle on the right commutes by (7.9) since $h = \overline{[\kappa^H, e^\dagger]}$ and $\beta = \widetilde{[a, e_A^\ddagger]}$. The overall outside is what we want.

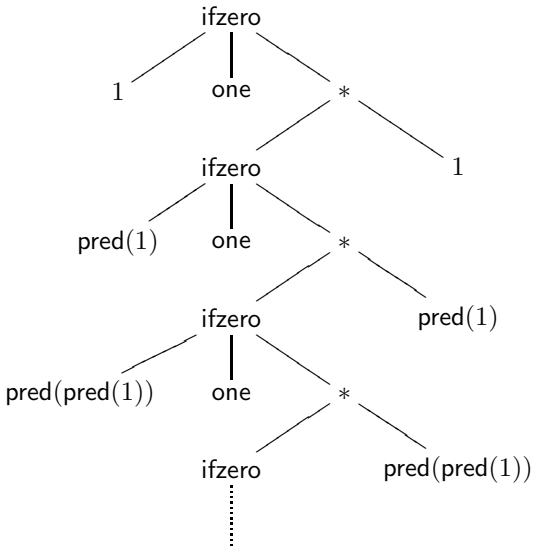
It is obvious that $e_A^\ddagger$ is uniquely determined by the commutativity of the triangle (7.11) as neither $\widetilde{a}$ nor $e^\dagger$ depend on $e_A^\ddagger$. □

7.1. **Interpreted Solutions in Computation Tree Elgot Algebras.** In this section, we work through some details concerning the factorial example of equation (1.2), repeated as

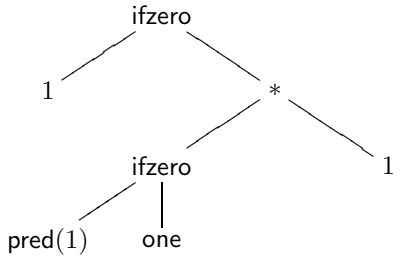
$$f(n) \approx \mathsf{ifzero}(n, \mathsf{one}, f(\mathsf{pred}(n) * n))$$

and studied further in Example 6.4(i). Recall that we work with the signature Σ containing a constant one, one unary symbol pred, a binary symbol $*$ and a ternary one ifzero. Let H_Σ be the associated endofunctor on Set. Let $\uparrow$ be an object which is not a natural number, and let $\mathbb{N}_\uparrow$ be the H_Σ-algebra with carrier $\{0, 1, 2, \ldots\} \cup \{\uparrow\}$ and whose operations are the strict extensions of the standard operations. (For example $\uparrow * 3 = \uparrow$ in this algebra.)

We shall use the computation tree Elgot algebra structure $(\mathbb{N}_\uparrow, a, (\_)^*)$ from Section 3.3. That structure used a particular element of the carrier set in connection with the conditional ifzero, and in our structure we take 0. Before looking at the interpreted solution to the factorial RPS, it might be useful to spell out the associated evaluation morphism $\widetilde{a} : \mathcal{T}(H_\Sigma)\mathbb{N}_\uparrow \longrightarrow \mathbb{N}_\uparrow$. Let t be a finite or infinite Σ-tree over $\mathbb{N}_\uparrow$; so the leaves of t might be labelled with natural number or $\uparrow$, but not with formal variables. Here is a pertinent example:



We got this by taking the uninterpreted solution of our RPS, as depicted in (6.4), and then substituting the number 1 for the formal variable n. Note that the nodes labelled ifzero have three children. Here is how we define $\widetilde{a}(t)$. We look for a finite subtree u of t with the property that if a node belongs to u and is labelled by a function symbol other than ifzero, then all its children belong to u as well; and if the node is labeled by ifzero, then either the first and second children belong to u, or else the first and third children do. For such a finite subtree u, we can evaluate the nodes in a bottom-up fashion using the H_Σ-algebra structure. We require that for a conditional node x, the first child evaluates to 0 (from our Elgot algebra structure) iff the second child is in u. If such a finite u exists, then we can read off an element of $\mathbb{N}_\uparrow$. This element is $\widetilde{a}(t)$. If no finite u exists, we set $\widetilde{a}(t) = \uparrow$. Returning to our example above, the finite subtree would be



And for our example tree t, $\widetilde{a}(t) = 1$.

We are now in a position to discuss the interpreted solution of our RPS. Recall that the signature Φ of recursively defined symbols contains only the unary symbol f. The corresponding signature functor is H_Φ, and $H_\Phi(\mathbb{N}_\uparrow)$ is the set $\{f(0), f(1), \ldots\} \cup \{f(\uparrow)\}$. The RPS itself is a natural transformation $e : H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$. The uninterpreted solution is the natural transformation $e^\dagger : H_\Phi \longrightarrow \mathcal{T}(H_\Sigma)$ corresponding to the tree shown in (6.4). We are concerned here with the interpreted solution $e^\ddagger_{\mathbb{N}_\uparrow} : H_\Phi(\mathbb{N}_\uparrow) \longrightarrow \mathbb{N}_\uparrow$ of our RPS. In light of the Fundamental Theorem 7.7, this is $\widetilde{a} \cdot (e^\dagger)_{\mathbb{N}_\uparrow}$. We show by an easy induction on $n \in N$ that this interpreted solution takes $f(n)$ to $n!$, and that it takes $f(\uparrow)$ to $\uparrow$.

We could also establish this same result directly, without Theorem 7.7. To do this, we follow the proof of Theorem 7.3. We turn our RPS e into a related natural transformation $\overline{e} : \mathcal{T}(H_\Sigma + H_\Phi) \longrightarrow H_\Sigma \mathcal{T}(H_\Sigma + H_\Phi) + Id$. Then $\overline{e}_{\mathbb{N}_\uparrow}$ is a flat equation morphism in the Elgot algebra $\mathbb{N}_\uparrow$, and its solution is the interpreted solution of our RPS. Here is a fragment of $\overline{e}_{\mathbb{N}_\uparrow}$:

$$\begin{aligned}
\underline{f(0)} &\approx \mathsf{ifzero}(\underline{0}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(0)) * 0}) \\
\underline{f(1)} &\approx \mathsf{ifzero}(\underline{1}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(1)) * 1}) \\
\underline{f(\mathsf{pred}(1)) * 1} &\approx \underline{\mathsf{ifzero}(\mathsf{pred}(1), \mathsf{one}, f(\mathsf{pred}(\mathsf{pred}(1))))} * \underline{1} \\
\underline{f(\mathsf{pred}(1))} &\approx \mathsf{ifzero}(\underline{\mathsf{pred}(1)}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(\mathsf{pred}(1)) * \mathsf{pred}(1))}) \\
\underline{\mathsf{pred}(1)} &\approx \mathsf{pred}(\underline{1}) \\
\underline{\mathsf{one}} &\approx 1
\end{aligned}$$

One can see that for each natural number n, the solution to this flat equation morphism assigns to $\underline{f(n)}$ the number $n!$.

7.2. **Interpreted Solutions in CPO.** We shall show in this subsection that if we have $\mathcal{A} = \mathsf{CPO}$ as our base category, then interpreted solutions of guarded RPSs e in an Elgot algebra $(A, a, (\_)^*)$ are given as least fixed points of a continuous operator on a function space. In this way we recover denotational semantics from our categorical interpreted semantics of recursive program schemes.

Example 7.9. We study the RPS of equation (1.2) as formalized in Example 6.4(i). As we know, the intended interpreted solution is the factorial function on the natural numbers $\mathbb{N}$.

This time we turn the natural numbers into an object of CPO so as to obtain a suitable Elgot algebra in which we can find an interpreted solution of (1.2). Let $\mathbb{N}_\perp$ be the flat CPO obtained from the discretely ordered $\mathbb{N}$ by adding a bottom element $\perp$, i. e., $x \leq y$ iff $x = \perp$ or $x = y$. We equip $\mathbb{N}_\perp$ with the strict operations $\mathsf{one}_{\mathbb{N}_\perp}$, $\mathsf{pred}_{\mathbb{N}_\perp}$, and $*_{\mathbb{N}_\perp}$. These are all strict and hence continuous. In addition, we use the continuous function

$$\mathsf{ifzero}_{\mathbb{N}_\perp}(n, x, y) \quad = \quad \begin{cases} \perp & \text{if } n = \perp \\ x & \text{if } n = 0 \\ y & \text{else} \end{cases}$$

Indeed, this is what we saw in (3.4) for the computation tree semantics, except we write $\perp$ for $\uparrow$. Hence we have a continuous Σ-algebra with $\perp$. Therefore $\mathbb{N}_\perp$ is an Elgot algebra for $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$, see Example 3.8(iii).

The standard interpreted solution $e^\ddagger_{\mathbb{N}_\perp} : H_\Phi \mathbb{N}_\perp \longrightarrow \mathbb{N}_\perp$ will certainly be *some* function or other on $\mathbb{N}_\perp$. But how do we know that this function is the desired factorial function? Usually one would simply regard the RPS (1.2) itself as a continuous function R on $\mathsf{CPO}(\mathbb{N}_\perp, \mathbb{N}_\perp)$ acting as

$$f(\_) \longmapsto \mathsf{ifzero}_{\mathbb{N}_\perp}(\_, 1, f(\mathsf{pred}_{\mathbb{N}}(\_) *_{\mathbb{N}_\perp} \_), \_)\,;$$

i. e., R is the operator described in (1.5) in the introduction. That means that we interpret all the operation symbols of Σ in the algebra $\mathbb{N}_\perp$. The usual denotational semantics assigns to the formal equation (1.2) with the interpretation in $\mathbb{N}_\perp$ the least fixed point of R. Clearly this yields the desired factorial function. And it is not difficult to work out that the least fixed point of R coincided with the standard interpreted solution $e^\ddagger_{\mathbb{N}_\perp}$ obtained from Theorem 7.3. We shall do this shortly in greater generality.

In general, any recursive program scheme can be turned into a continuous operator R on the function space $\mathsf{CPO}(VA, A)$. Theorem 7.10 below shows that the least fixed point of R is the same as the interpreted solution obtained from Theorem 7.3.

We assume throughout this subsection that H, V and $H+V$ are locally continuous (and, as always, iteratable) endofunctors of CPO. We also consider a fixed guarded RPS $e : V \longrightarrow \mathcal{T}(H+V)$, and an H-algebra (A, a) with a least element $\bot$. By Example 3.8(ii), we know that this carries the structure of an Elgot algebra $(A, a, (\_)^*)$, where $(\_)^*$ assigns to every flat equation morphism a least solution. As before, we will use the notation $\widetilde{a} : \mathcal{T}(H)A \longrightarrow A$ for the induced evaluation morphism. Furthermore, for any continuous map $f : VA \longrightarrow A$ we have an Elgot algebra on A with structure $[a, f] : (H+V)A \longrightarrow A$. Due to Corollary 3.15, its evaluation morphism satisfies the equation

$$\widetilde{[a,f]} \cdot \kappa_A^{H+V} = [a, f]\,. \tag{7.12}$$

Theorem 7.10. *The following function R on $\mathsf{CPO}(VA, A)$*

$$f \longmapsto VA \xrightarrow{e_A} \mathcal{T}(H+V)A \xrightarrow{\widetilde{[a,f]}} A \tag{7.13}$$

is continuous. Its least fixed point is the standard interpreted solution $e_A^\ddagger : VA \longrightarrow A$ of Theorem 7.3.

Proof. (i) To see the continuity of R is suffices to prove that the function

$$\widetilde{(\_)} : \mathsf{CPO}(HA, A) \longrightarrow \mathsf{CPO}(\mathcal{T}(H)A, A)$$

is continuous. Let us write T for $\mathcal{T}(H)$. Recall that for any continuous map $a : HA \longrightarrow A$, the evaluation morphism $\widetilde{a}$ is the least solution of the flat equation morphism $\alpha_A : TA \longrightarrow HTA + A$, i. e., $\widetilde{a}$ is the least fixed point of the continuous function

$$F(a, -) : \mathsf{CPO}(TA, A) \longrightarrow \mathsf{CPO}(TA, A) \qquad f \longmapsto [a, A] \cdot (Hf + A) \cdot \alpha_A\,.$$

Observe that F is continuous in the first argument a, and so F is a continuous function on the product $\mathsf{CPO}(HA, A) \times \mathsf{CPO}(TA, A)$. It follows from standard arguments that taking the least fixed point in the second argument yields a continuous map $\mathsf{CPO}(HA, A) \longrightarrow \mathsf{CPO}(TA, A)$. But this is precisely the desired one $\widetilde{(\_)}$.

(ii) We prove that $e_A^\ddagger$ is the least fixed point of R. Notice that the least fixed point of R is the join t of the following increasing chain in $\mathsf{CPO}(VA, A)$:

$$t_0 = \mathsf{const}_\bot : VA \longrightarrow A, \qquad t_{i+1} \equiv VA \xrightarrow{e_A} \mathcal{T}(H+V)A \xrightarrow{\widetilde{[a,t_i]}} A, \qquad \text{for } i \geq 0.$$

Furthermore, recall that the interpreted solution $e_A^\ddagger$ is defined by (7.6) as

$$e_A^\ddagger \equiv \beta \cdot \kappa_A^{H+V} \cdot \mathsf{inr}\,,$$

where $\beta = g^*$ is the least solution of the flat equation morphism g which is obtained from the component at A of the $\mathcal{H}$-coalgebra $\overline{e}$, see Lemma 6.9 and Theorem 7.3. By Example 3.8(ii), the solution β of g is the join of the chain

$$\beta_0 = \mathsf{const}_\bot, \qquad \beta_{i+1} = [a, A] \cdot H(\beta_i + A) \cdot g, \quad \text{for } i \geq 0.$$

Observe that $e_A^\ddagger$ is a fixed point of R, see (7.1). Thus, we have $t \sqsubseteq e_A^\ddagger$. To show the reverse inequality we will prove by induction on i the inequalities

$$\beta_i \sqsubseteq \widetilde{[a,t]}\,, \quad i \in \mathbb{N}\,. \tag{7.14}$$

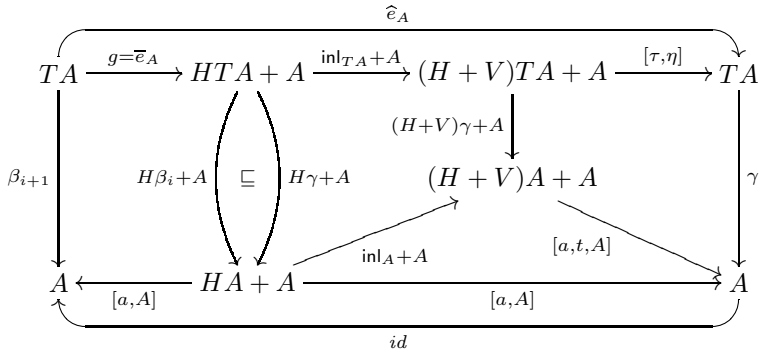
This implies that $\beta \sqsubseteq \widetilde{[a,t]}$ and therefore

$$e_A^\ddagger = \beta \cdot \kappa_A^{H+V} \cdot \mathsf{inr} \sqsubseteq \widetilde{[a,t]} \cdot \kappa_A^{H+V} \cdot \mathsf{inr} = t\,,$$

where the last equality follows from (7.12).

We complete the proof with the induction proof to establish (7.14). The base case is clear. For the induction step we write T for $\mathcal{T}(H+V)$ and γ as a short notation for $\widetilde{[a,t]}$ and we consider the following

diagram



Its upper part commutes due to Lemma 6.12, the left-hand part is the definition of β_{i+1}, and the inequality follows from the induction hypothesis. For the right-hand part recall that $\gamma = \widetilde{[a,t]} : TA \longrightarrow A$ is a homomorphism of Elgot algebras, see Theorem 3.13. Hence, γ is an algebra homomorphism by Proposition 3.6. Finally, the remaining parts of the above diagram clearly commute. Thus we obtain the inequality $\beta_{i+1} \sqsubseteq \gamma \cdot \widehat{e}_A$. Finally, since t is a fixed point of R it is an interpreted solution of e in A. By Lemma 7.4, it follows that $\gamma \cdot \widehat{e}_A = \gamma$, and this establishes the desired inequality. □

Remark 7.11. The result of Theorem 7.10 implies a concrete formula

$$e^\ddagger_A = \bigsqcup_{n<\omega} e^\ddagger_n$$

for the interpreted solution of the guarded RPS e in the continuous algebra A. In fact, the least fixed point of R is the join of the ascending chain

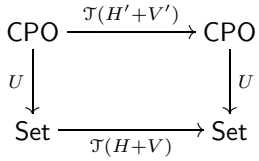
$$\bot \sqsubseteq R(\bot) \sqsubseteq R^2(\bot) \sqsubseteq \cdots$$

where $\bot = \mathsf{const}_\bot$ is the least element of $\mathsf{CPO}(VA, A)$. Thus, with $e^\ddagger_0 = \mathsf{const}_\bot$ and

$$e^\ddagger_{n+1} \equiv VA \xrightarrow{e_A} \mathcal{T}(H+V)A \xrightarrow{\widetilde{[a,e^\ddagger_n]}} A$$

we obtain the above formula for $e^\ddagger_A$.

Remark 7.12. Suppose that H, V and $H+V$ are iteratable endofunctors of Set, which have locally continuous liftings H' and V' to CPO. Then we have a commutative square



see Example 2.5(iv). Furthermore, assume that the guarded RPS $e : V \longrightarrow \mathcal{T}(H+V)$ has a lifting $e' : V' \longrightarrow \mathcal{T}(H'+V')$; i. e., a natural transformation e' such that $U * e' = e * U$. Now consider any CPO-enrichable H-algebra (A, a) as an Elgot algebra, see Example 3.8(iii). Then we can apply Theorem 7.10 to obtain the standard interpreted solution $e^\ddagger_A$ of e in the algebra A as a least fixed point of the above function R of (7.13).

Example 7.13.

(i) Suppose we have signatures Σ and Φ. Then the signature functors H_Σ and H_Φ have locally continuous liftings H'_Σ and H'_Φ. Since the lifting of $H_\Sigma + H_\Phi$ is a lifting of $H_{\Sigma+\Phi}$ we know that $\mathcal{T}(H'_\Sigma + H'_\Phi)$ assigns to any cpo X the algebra $T_{\Sigma+\Phi}X$ with the cpo structure induced by X, see Example 2.5(v). More precisely, to compare a tree t to a tree s replace all leaves labelled by a variable from X by a leaf labelled by some extra symbol $\star$ to obtain relabelled trees t' and s'. Then t is less than s iff t' and s' are isomorphic as labelled trees, and for any leaf of t labelled by a variable x the corresponding leaf in s is labelled by a variable y with $x \sqsubseteq y$ in X.

Now consider any system as in (2.3) which is in Greibach normal form, and form the associated guarded RPS $e : H_\Phi \longrightarrow T_{\Sigma+\Phi}$. Then e has a lifting $e' : H'_\Phi \longrightarrow \mathcal{T}(H'_\Sigma + H'_\Phi)$. In fact, for any cpo X the component $e'_X = e_X : H_\Phi X \longrightarrow T_{\Sigma+\Phi}X$ is a continuous map since the order in $H_\Phi X$ is given similarly as for $T_{\Sigma+\Phi}X$ on the level of variables only: $(f, \vec{x}) \sqsubseteq (g, \vec{y})$ holds for elements of $H_\Phi X$ if $f = g \in \Phi_n$ and $x_i \sqsubseteq y_i$, $i = 1, \ldots, n$, holds in X.

Let (A, a) be a CPO-enrichable H_Σ-algebra; i. e., a continuous Σ-algebra with a least element $\bot$. We wish to consider the continuous function R on $\mathsf{CPO}(H_\Phi A, A)$ which assigns to any continuous algebra structure $\varphi : H_\Phi A \longrightarrow A$ the algebra structure $R(\varphi) = \widetilde{[a, \varphi]} \cdot e'_A$. The structure $R(\varphi) : H_\Phi A \longrightarrow A$ gives to each n-ary operation symbol f of Φ the operation $t^f_A : A^n \longrightarrow A$ which is obtained as follows: take the term t^f provided by the right-hand side of f in our given RPS, then interpret all operation symbols of Σ in t^f according to the given algebraic structure a and all operation symbols of Φ according to φ; the action of t^f_A is evaluation of that interpreted term.

Theorem 7.10 states that the standard interpreted solution $e^\ddagger_A$ of e in the algebra A can be obtained by taking the least fixed point of R; in other words the standard interpreted solution $e^\ddagger_A$ gives the usual denotational semantics.

(ii) Apply the previous example to the RPS of Example 6.4(i). Then Theorem 7.10 states that the interpreted solution of the RPS (1.2) in the Elgot algebra $\mathbb{N}_\bot$ is obtained as the least fixed point of the function R of Example 7.9. That is, the standard interpreted solution gives the desired factorial function.

(iii) Recall the guarded RPS $e : V \longrightarrow \mathcal{T}(H + V)$ from Example 6.4(ii) whose uninterpreted solution we have described in Example 6.6(ii). Consider again the algebra $\mathbb{N}_\bot$ together with the following two operations:

$$F_{\mathbb{N}_\bot}(x, y, z) = \begin{cases} x & \text{if } x = y \\ z & \text{else} \end{cases} \qquad G_{\mathbb{N}_\bot}(x) = \begin{cases} \lfloor \frac{x}{2} \rfloor & \text{if } x \in \mathbb{N} \\ \bot & x = \bot \end{cases} \tag{7.15}$$

Since the first operation obviously satisfies $F_{\mathbb{N}_\bot}(x, y, z) = F_{\mathbb{N}_\bot}(y, x, z)$ we have defined an H-algebra. It is not difficult to check that the set functor H has a locally continuous lifting H' to CPO and that $\mathbb{N}_\bot$ is a continuous H'-algebra. In fact, the existence of the lifting H' follows from the fact that the unordered pair functor $V : \mathsf{Set} \longrightarrow \mathsf{Set}$ can be lifted to CPO; the lifting assigns to a cpo $(X, \leq)$ the set of unordered pairs with the following order: $\{ x, y \} \sqsubseteq \{ x', y' \}$ iff either $x \leq x'$ and $y \leq y'$, or $x \leq y'$ and $y \leq x$. Thus, we have defined an Elgot algebra for $H : \mathsf{Set} \longrightarrow \mathsf{Set}$, see Example 3.8(iii). The standard interpreted solution $e^\ddagger_{\mathbb{N}_\bot} : V\mathbb{N}_\bot \longrightarrow \mathbb{N}_\bot$ is given by one commutative binary operation $\varphi_{\mathbb{N}_\bot}$ on $\mathbb{N}_\bot$. We leave it to the reader to verify that for natural numbers n and m, $\varphi_{\mathbb{N}_\bot}(n, m)$ is the natural number represented by the greatest common prefix in the binary representation of n and m, e. g., $\varphi_{\mathbb{N}_\bot}(12, 13) = 6$. Notice that we do not have to prove separately that $\varphi_{\mathbb{N}_\bot}$ is commutative. In Example 6.4(ii) we have encoded that extra property directly into the RPS e so that any solution must be commutative.

(iv) Least fixed points are RPS solutions. Let A be a poset with joins of all subsets which are at most countable, and let $f : A \longrightarrow A$ be a function preserving joins of ascending chains. Take f and binary joins to obtain an algebra structure on A of the signature functor $H_\Sigma X = X + X \times X$ expressing a binary operation symbol F and a unary one G. Obviously, this functor has a lifting $H' : \mathsf{CPO} \longrightarrow \mathsf{CPO}$ and A is a CPO-enrichable algebra, i. e., A is an Elgot algebra. Turn the formal equations (1.1) into a recursive program scheme $e : H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$ as demonstrated in Section 2.3. The RPS e has a lifting $e' : V' \longrightarrow \mathcal{T}(H' + V')$, where V' denotes the lifting of H_Φ. The standard interpreted solution $e^\ddagger_A : V'A \longrightarrow A$ gives two continuous functions φ_A and ψ_A on A. Clearly, we have $\varphi_A(a) = \bigvee_{n \in \mathbb{N}} f^n(a)$, and in particular $\varphi_A(\bot)$ is the least fixed point of f.

7.3. **Interpreted Solutions in CMS.** Recall the category CMS of complete metric spaces from Example 2.2(iii), and let $H, V : \mathsf{CMS} \longrightarrow \mathsf{CMS}$ be contracting endofunctors. We shall show in this subsection that for any guarded RPS $e : V \longrightarrow \mathcal{T}(H + V)$ we can find a unique interpreted solution in any non-empty H-algebra A. More precisely, assume that we have such a guarded RPS e, and let (A, a) be a non-empty H-algebra. Then A is a cia, and, in particular it carries the structure of an Elgot algebra. Notice that for any non-expanding map $f : VA \longrightarrow A$ we obtain an algebra structure $[a, f] : (H + V)A \longrightarrow A$, thus we have the evaluation morphism

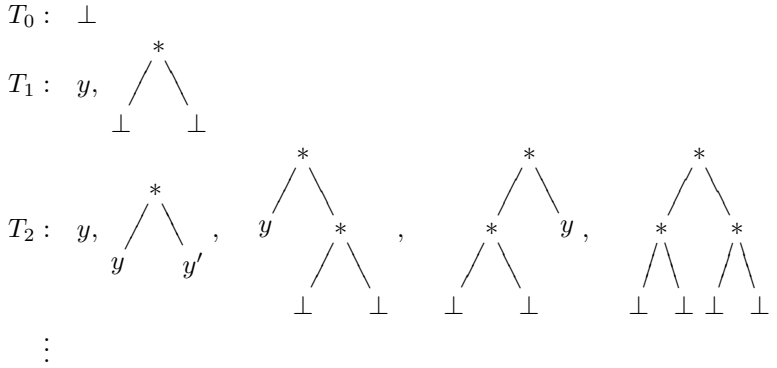
$$\widetilde{[a, f]} : \mathcal{T}(H + V)A \longrightarrow A\,.$$

As in CPO, the RPS e induces a function R on $\mathsf{CMS}(VA, A)$, see (7.13). The standard procedure for obtaining an interpreted solution would be to prove that R is a contracting map, and then invoke Banach's Fixed Point theorem to obtain a unique fixed point of R. Here we simply apply Theorem 7.3. Notice, however, that we cannot completely avoid Banach's Fixed Point theorem: it is used in the proof that final coalgebras exist for contracting functors, see [ARe].

Corollary 7.14. *The unique interpreted solution $e^\dagger_A : VA \longrightarrow A$ of e in A as obtained in Theorem 7.3 is the unique fixed point of the function R on $\mathsf{CMS}(VA, A)$ defined by (7.13).*

Proof. In fact, being a fixed point of R is equivalent to being an interpreted solution of e in the cia A, whose unique existence we have by Theorem 7.3. □

Remark 7.15. Let H_Σ be a signature functor on Set and denote by H' a lifting to CMS as described in Example 3.2(v). For a complete metric space Y the final coalgebra $\mathcal{T}(H')Y$ of $H'(\_) + Y$ is the set $T_\Sigma Y$ of all Σ-trees over Y equipped with a suitable complete metric. This metric can be described as follows. Recall from [ARe] that $\mathcal{T}(H')Y$ is obtained as T_ω after ω steps of the final coalgebra chain for $H'(\_) + Y$, see Construction 2.3. That means the metric on $T_\Sigma Y$ is the smallest metric such that all projections $t_{\omega,i} : T_\Sigma Y = T_\omega \longrightarrow T_i$ are non-expanding. We illustrate this with an example adapted from [ARe]. Let $H_\Sigma X = X \times X$ be the functor expressing one binary operation symbol $*$. Then we can represent $T_0 = 1$ by a single node tree labelled with $\bot$ and $T_{i+1} = T_i \times T_i + Y$ by trees which are either single node trees labelled in Y, or which are composed by joining two trees from T_i with a root labelled by $*$:



The distance on T_1 is that of Y for single node trees and 1 otherwise. The distance on T_2 is again that of Y between single node trees, and 1 between single node trees and all other trees. Furthermore, the distance between trees of different shapes is $\frac{1}{2}$, and finally, $d_{T_2}(y * y', z * z') = \frac{1}{2} \max\{\, d_Y(y, z), d_Y(y', z') \,\}$ as well as $d_{T_2}(y * t, y' * t) = d_{T_2}(t * y, t * y') = \frac{1}{2} d_Y(y, y')$, where $t = \bot * \bot$, etc. In general, the distance on T_{i+1} is that of Y between single node trees, it is 1 between single node trees and trees of height at least 1, and otherwise we have $d_{T_{i+1}}(s * t, s' * t') = \frac{1}{2} \max\{\, d_{T_i}(s, s'), d_{T_i}(t, t') \,\}$. For the metric on $T_\Sigma Y$, we have

$$d_{T_\Sigma Y}(s_1, s_2) = \sup_{i<\omega} d_{T_i}(t_{\omega,i}(s_1), t_{\omega,i}(s_2)).$$

This is the smallest metric for which the projections are non-expanding. (One may also verify directly that this definition gives a complete metric space structure and that $H'(\_) + Y$ preserves the limit, so that we indeed have a final coalgebra.) Finally notice that the metric of $T_\Sigma Y$ depends on the choice of the lifting H'. For example, if we lift the functor H_Σ as $H'(X, d) = (X^2, \frac{1}{3} d_{\max})$, the factor $\frac{1}{2}$ would have to be replaced by $\frac{1}{3}$ systematically.

Example 7.16.

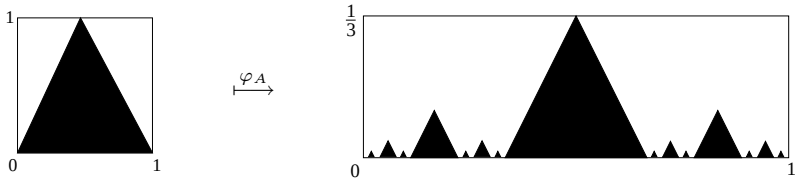
(i) Consider the endofunctor $H' : \mathsf{CMS} \longrightarrow \mathsf{CMS}$ obtained by lifting the signature functor $H_\Sigma X = X \times X + X$ expressing a binary operation F and a unary one G as described in Example 2.5(v). The Euclidean interval $I = [0, 1]$ together with the operations $F(x, y) = \frac{x+y}{4}$ and $G(x) = \frac{\sin(x)}{2}$ is an H'-algebra, whence a cia. Use only the first equation in (1.1) to obtain a guarded RPS $e : Id \longrightarrow \mathcal{T}(H_\Sigma + Id)$ where Id expresses the unary operation symbol φ. Let V' be contracting lifting of Id with a contraction factor of $\varepsilon = \frac{1}{2}$. Then e gives rise to a guarded RPS $e' : V' \longrightarrow \mathcal{T}(H' + V')$ in CMS. The unique interpreted solution of e' in I consists of a function $\varphi_I : I \longrightarrow I$ satisfying $\varphi_I(x) = \frac{1}{4}(x + \varphi_I(\frac{1}{2} \sin x))$, that is, φ_I is the unique function f satisfying (1.3).

(ii) Self-similar sets are solutions of interpreted program schemes. Recall from Example 3.2(v) that for any complete metric space (X, d) we obtain the complete metric space $(C(X), h)$ of all non-empty compact subspaces of X with the Hausdorff metric. Furthermore, contractive mappings of X yield structures of cias on $C(X)$. Now consider the functor H' on CMS with $H(X, d) = (X^3, \frac{1}{3} d_{\max})$, where $d_{\max}$ is the maximum metric. It is a lifting of the signature functor H_Σ on Set expressing one ternary operation α. Let $A = [0, 1] \times [0, 1]$, be equipped with the usual Euclidean metric. Consider the contracting maps $f(x, y) = (\frac{1}{3}x, \frac{1}{3}y)$, $g(x, y) = (\frac{1}{3}x + \frac{1}{3}, \frac{1}{3}y)$, and $h(x, y) = (\frac{1}{3}x + \frac{2}{3}, \frac{1}{3}y)$ of A.

Then it follows that $\alpha_A : C(A)^3 \longrightarrow C(A)$ with $\alpha(D, E, F) = f[D] \cup g[E] \cup h[F]$ is a $\frac{1}{3}$-contracting map, whence a structure of an H'-algebra. The formal equation

$$\varphi(x) \approx \alpha(\varphi(x), x, \varphi(x))$$

gives rise to a guarded RPS $e : Id \longrightarrow \mathcal{T}(H_\Sigma + Id)$, where the identity functor expresses the operation φ. If we take the lifting of Id to CMS which is given by $V'(X, d) = (X, \frac{1}{3}d)$, then e gives rise to a natural transformation $e' : V' \longrightarrow \mathcal{T}(H' + V')$. Its interpreted solution in the algebra $C(A)$ is a $\frac{1}{3}$-contracting map $\varphi_A : C(A) \longrightarrow C(A)$ which maps a non-empty compact subspace U of A to a space of the following form: $\varphi_A(U)$ has three parts, the middle one is a copy of U scaled by $\frac{1}{3}$, and the left-hand and right-hand one look like copies of the whole space $\varphi_A(U)$ scaled by $\frac{1}{3}$. For example, we have the assignment



(iii) Coming back to Example 3.2(vi) let us consider $(C(I), \alpha_I)$, where $I = [0, 1]$ is the Euclidean interval, $C(I)$ is the set of all non-empty closed subsets of I, and α_I is the structure of a cia arising from $f(x) = \frac{1}{3}x$ and $g(x) = \frac{1}{3}x + \frac{2}{3}$ as described is Example 3.2(v). The formal equation

$$\varphi(x) \approx \alpha(\varphi(x), x)$$

gives similarly as in (i) above a guarded RPS $e : Id \longrightarrow \mathcal{T}(H_\Sigma + Id)$, where $H_\Sigma X = X \times X$ now expresses the binary operation α. Again, we have liftings $V'(X, d) = (X, \frac{1}{3}d)$ and $H'(X, d) = (X^2, \frac{1}{3}d_{\max})$ of Id and H_Σ, respectively. So the RPS e lifts to the guarded RPS $e' : V' \longrightarrow \mathcal{T}(H' + V')$ in CMS. Its unique interpreted solution is given by the $\frac{1}{3}$-contracting map $\varphi_I : C(I) \longrightarrow C(I)$ satisfying $\varphi_I(t) = \alpha_I(\varphi_I(t), t) = f[\varphi_I(t)] \cup g[t]$ for every non-empty closed subset t of the interval I.

8. Conclusions and Future Work

We have presented a general and conceptually clear way of treating the uninterpreted and the interpreted semantics of recursive program schemes in a category theoretic setting. For this we have used recent results on complete Elgot algebras and results from the theory of coalgebras. We have shown that our theory readily specializes to the classical setting yielding denotational semantics using complete partial orders or complete metric spaces. We have presented new applications of recursive program scheme solutions including fractal self-similarity and also applications which cannot be handled by the classical methods; defining operations satisfying equations like commutativity. Another new application, recursively defined functions on non-wellfounded sets, will be treated in a future paper.

Now one must go forward in reinventing algebraic semantics with category theoretic methods. We strongly suspect that there is much to be said about the relation of our work to operational semantics. We have not investigated higher-order recursive program schemes using our tools, and it would be good to know whether our approach applies in that area as well. The paper [MU] addresses variable binding and infinite terms coalgebraically, and this may well be relevant. Back to the classical theory, one of the main goals of the original theory is to serve as a foundation for program equivalence. It is not difficult to prove the soundness of fold/unfold transformations in an algebraic way using our semantics; this was done in [Mo_2] for uninterpreted schemes. We study the equational properties of our very general formulation of recursion in [MM_2]. One would like more results of this type. The equivalence of interpreted schemes in the natural numbers is undecidable, and so one naturally wants to study the equivalence of interpreted schemes in *classes of interpretations*. The classical theory proposes classes of interpretations, many of which are defined on ordered algebras, see [G]. It would be good to revisit this part of the classical theory to see whether Elgot algebras suggest tractable classes of interpretations.

Another path of future research is the study of algebraic trees with categorical methods. In the setting of trees over a signature Σ the solutions of recursive program schemes form the theory of algebraic trees, a subtheory of the theory of all trees on Σ. Moreover, algebraic trees are closed under second-order substitution and they form an iterative theory in the sense of Elgot [E]. Similar results should be possible to obtain in our generalized categorical setting. First promising steps in the direction of categorically studying algebraic trees have been taken in [AMV_4].

Acknowledgments

We are grateful to Jiří Adámek for many conversations on this and related matters, and for his enthusiasm for our project.

References

[A] Peter Aczel, *Non-Well-Founded Sets*. CSLI Lecture Notes Number 14, CSLI Publications, Stanford, 1988.

[AAV] P. Aczel, J. Adámek and J. Velebil, A Coalgebraic View of Infinite Trees and Iteration, *Electron. Notes Theor. Comput. Sci.* 44 (2001), no. 1, 26 pp.

[AAMV] P. Aczel, J. Adámek, S. Milius and J. Velebil, Infinite Trees and Completely Iterative Theories: A Coalgebraic View, *Theoret. Comput. Sci.* 300 (2003), 1–45.

[AK] J. Adámek and V. Koubek, On the greatest fixed point of a set functor, *Theoret. Comput. Sci.* 150 (1995), 57–75.

[AM] J. Adámek and S. Milius, Terminal Coalgebras and Free Iterative Theories, accepted for publication in *Inform. and Comput.*

[AMV$_1$] J. Adámek, S. Milius and J. Velebil, Free Iterative Theories: A Coalgebraic View, *Math. Structures Comput. Sci.* 13 (2003), 259–320.

[AMV$_2$] J. Adámek, S. Milius and J. Velebil, From Iterative Algebras to Iterative Theories, extended abstract appeared in *Electron. Notes Theor. Comput. Sci.* 106 (2004), 3–24, full version submitted and available at the URL `http://www.iti.cs.tu-bs.de/~milius`.

[AMV$_3$] J. Adámek, S. Milius and J. Velebil, On coalgebra based on classes, *Theoret. Comput. Sci.* 316 (2004), 3-23.

[AMV$_3$] J. Adámek, S. Milius and J. Velebil, Elgot Algebras, preprint, 2005, available at the URL `http://www.iti.cs.tu-bs.de/~milius`, extended abstract to appear in *Electron. Notes Theor. Comput. Sci.*

[AMV$_4$] J. Adámek, S. Milius and J. Velebil, Algebraic Trees? Who knows. . . , working draft, May 2005.

[AP] J. Adámek and H. E. Porst, On tree coalgebras and coalgebra presentations, *Theoret. Comput. Sci.* 311 (2004), 257–283.

[ARe] J. Adámek and J. Reitermann, Banach's Fixed-Point Theorem as a Base for Data-Type Equations, *Appl. Categ. Structures* 2 (1994), 77–90.

[AT] J. Adámek and V. Trnková, *Automata and Algebras in Categories*. Kluwer Academic Publishers, 1990.

[ARu] P. America and J. J. M. M. Rutten, Solving Reflexive Domain Equations in a Category of Complete Metric Spaces, *J. Comput. System Sci.* 39 (1989), 343–375.

[AN] A. Arnold and M. Nivat, The metric space of infinite trees. Algebraic and topological properties, *Fund. Inform.* III, no. 4 (1980), 445–476.

[B] M. F. Barnsley, *Fractals Everywhere*, Academic Press 1988.

[Ba] M. Barr, Terminal coalgebras in well-founded set theory, *Theoret. Comput. Sci.* 114 (1993), 299–315.

[BM] J. Barwise and L. S. Moss, *Vicious Circles*, CSLI Publications, Stanford, 1996.

[Bl] S. L. Bloom, All Solutions of a System of Recursion Equations in Infinite Trees and Other Contraction Theories, *J. Comput. System Sci.* 27 (1983), 225–255.

[BE] S. L. Bloom and Z. Ésik, *Iteration Theories: The equational logic of iterative processes,* EATCS Monographs on Theoretical Computer Science, Berlin: Springer-Verlag (1993).

[Bo] F. Borceux, *Handbook of Categorical Algebra 2: Categories and Structures*, Vol. 51 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1994.

[C] B. Courcelle, Fundamental properties of infinite trees, *Theoret. Comput. Sci.* 25 (1983), no. 2, 95–169.

[E] C. C. Elgot, Monadic Computation and Iterative Algebraic Theories, in: *Logic Colloquium '73* (eds: H. E. Rose and J. C. Shepherdson), North-Holland Publishers, Amsterdam, 1975.

[EBT] C. C. Elgot, S. L. Bloom and R. Tindell, On the Algebraic Structure of Rooted Trees, *J. Comput. System Sci.* 16 (1978), 361–399.

[GLMP$_1$] N. Ghani, C. Lüth, F. De Marchi and A. J. Power, Algebras, coalgebras, monads and comonads, *Electron. Notes Theor. Comput. Sci.* 44 (2001), no. 1, 18 pp.

[GLMP$_2$] N. Ghani, C. Lüth, F. De Marchi and A. J. Power, Dualising initial algebras, *Math. Structures Comput. Sci.* 13 (2003), no. 2, 349–370.

[GLM] N. Ghani, C. Lüth and F. De Marchi, Solving Algebraic Equations using Coalgebra, *Theor. Inform. Appl.* 37 (2003), 301–314.

[ADJ] J. A. Goguen, S. W. Thatcher, E. G. Wagner and J. B. Wright, Initial Algebra Semantics and Continuous Algebras, *J. ACM* 24 (1977), 68–95.

[G] I. Guessarian, *Algebraic Semantics*. Lecture Notes in Comput. Sci., Vol. 99, Springer, 1981.

[L] J. Lambek, A Fixpoint Theorem for Complete Categories, *Math. Z.* 103 (1968), 151–161.

[L$_1$] T. Leinster, General self-similarity: an overview, e-print math.DS/0411343 v1.

[L$_2$] T. Leinster, A general theory of self-similarity I, e-print math.DS/041344.

[L$_3$] T. Leinster, A general theory of self-similarity II, e-print math.DS/0411345.

[ML] S. MacLane, *Categories for the working mathematician*, 2nd edition, Springer Verlag, 1998.

[MU] R. Matthes and T. Uustalu, Substitution in Non-Wellfounded Syntax with Variable Binding. In H. P. Gumm, (ed.), *Electron. Notes Theor. Comput. Sci.*, 82 (2003), no. 1, 15 pp.

[Mi$_1$] S. Milius, On Iteratable Endofunctors, *Electron. Notes Theor. Comput. Sci.* 69 (2002), 18 pp.

[Mi$_2$] S. Milius, Completely Iterative Algebras and Completely Iterative Monads, *Inform. and Comput.* 196 (2005), 1–41.

[MM] S. Milius and L. S. Moss, The Category Theoretic Solution of Recursive Program Schemes, extended abstract, in Proceedings of the 1st International Conference on Algebra and Coalgebra in Computer Science (CALCO 2005), *Lecture Notes in Comput. Sci.* 3629 (2005), 293–312.

[MM_2] S. Milius and L. S. Moss, Equational Properties of Solutions to Recursive Program Schemes, working draft.
[Mo_1] L. S. Moss, Parametric Corecursion, *Theoret. Comput. Sci.* 260 (2001), no. 1–2, 139–163.
[Mo_2] L. S. Moss, The Coalgebraic Treatment of Second-Order Substitution and Uninterpreted Recursive Program Schemes, preprint, 2002.
[Mo_3] L. S. Moss, Uniform Functors on Sets, submitted.
[N] M. Nivat, On the Interpretation of Recursive Polyadic Program Schemes, *Symposia Mathematica XV* (1975), 255–281.
[W] J. Worrell, On the Final Sequence of a Finitary Set Functor, *Theoret. Comput. Sci.* 338 (2005), 184–199.

Institute of Theoretical Computer Science, Technical University, Braunschweig, Germany
E-mail address: milius@iti.cs.tu-bs.de

Department of Mathematics, Indiana University, Bloomington, IN, USA
E-mail address: lsm@cs.indiana.edu