Von der Carl-Friedrich-Gauß-Fakultät für Mathematik
und Informatik der Technischen Universität Braunschweig

genehmigte Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

Stefan Milius

Coalgebras, Monads and Semantics

28. Oktober 2005

1. Referent: Prof. Jiří Adámek, Technische Universität Braunschweig, Deutschland

2. Referent: Prof. Lawrence S. Moss, Indiana University, Bloomington, Indiana, USA

eingereicht am: 14. Juli 2005

To the memory of my mother.

About this Thesis

This is a cumulative dissertation which consists of the following papers:

[AMV$_1$]	Free Iterative Theories: a coalgebraic view
[AMV$_2$]	From Iterative Algebras to Iterative Theories
[AMV$_3$]	Elgot Algebras
[AAMV]	Infinite Trees and Completely Iterative Theories: A Coalgebraic View
[M$_1$]	Completely Iterative Algebras and Completely Iterative Monads
[M$_2$]	On Iteratable Endofunctors
[MM]	The Category Theoretic Solution of Recursive Program Schemes

These papers contain some of the results of my research obtained while I was working at the Institute of Theoretical Computer Science of the Technical University of Braunschweig, Germany, under the supervision of Professor Jiří Adámek. They have all been published in international journals and/or they have been presented at international conferences.

The current document constitutes an introduction and summary of this research. As such it contains (almost) no new results as compared to the above papers. Its aim is to provide the reader with a self-contained account of our research in a unified notation and presentation. The current document is structured in a way so as to maximize accessibility to the reader while still providing enough explanation of the techniques and tools employed in our work. All important results from the above papers are included in this document. However, in a summary, one is forced to omit material. And so many of the technical details, and certainly all the technical calculations have been omitted so that full proofs are almost never given here. We provide sketches of proofs meant to illustrate the main ideas at work in our theory. For those readers who want to delve deeper into the omitted details we have annotated our results with precise references to their places in the above papers, which we include in an appendix.

Abstract

We study mathematical structures arising in the theory of *coalgebras* which are useful for providing semantics of recursive specifications. We begin by investigating (completely) iterative algebras and (complete) Elgot algebras, and we establish the connection of these algebras to coalgebra—final coalgebras are precisely the same as free completely iterative algebras, or free complete Elgot algebras, respectively. Similarly, for the non-complete case we show that free iterative algebras, and free Elgot algebras, respectively, exist and can be constructed from finite coalgebras. Furthermore, we show that the *monads* arising from free (completely) iterative algebras are characterized by a universal property; they are free (completely) iterative monads. This generalizes and extends classical work of Calvin Elgot and Evelyn Nelson. The second main topic of this thesis is to provide a category theoretic *semantics* of so-called recursive program schemes. By exploiting the structure of free completely iterative monads and of complete Elgot algebras we are able to provide an uninterpreted as well as an interpreted semantics of recursive program schemes. We show that both are connected as expected from the classical work. Finally, we present applications of our abstract theory. We demonstrate how to recover the usual denotational semantics using ordered or metrized structures, and we present new applications defining for example operations satisfying certain equations, or fractals recursively.

Zusammenfassung

In dieser Arbeit werden mathematische Strukturen untersucht, die in der Theorie der *Koalgebren* auftauchen und welche Anwendung für die Semantik rekursiver Definitionen finden. Zunächst werden dazu (vollständig) iterative Algebren und (vollständige) Elgot Algebren betrachtet, und es wird deren Verbindung zu Koalgebren offengelegt – finale Koalgebren sind genau die freien vollständig iterativen Algebren bzw. die freien vollständigen Elgot Algebren. Analog hierzu zeigen wir, dass freie (nicht notwendig vollständige) iterative Algebren bzw. freie Elgot Algebren existieren und in kanonischer Weise aus endlichen Koalgebren konstruiert werden können. Weiterhin wird bewiesen, dass die *Monaden*, welche sich aus freien (vollständig) iterativen Algebren ergeben, durch eine universelle Eigenschaft charakterisiert sind; sie sind die freien (vollständig) iterativen Monaden. Dieser Teil der Arbeit verallgemeinert und erweitert klassische Ergebnisse von Calvin Elgot und Evelyn Nelson. Das zweite Hauptthema der vorliegenden Dissertation ist eine kategorientheoretische *Semantik* so genannter rekursiver Programschemata. Durch die Ausnutzung der Struktur freier vollständig iterativer Monaden und vollständiger Elgot Algebren sind wir in der Lage sowohl eine uninterpretierte als auch eine interpretierte Semantik rekursiver Programmschemata anzugeben. Wir zeigen auch, dass zwischen beiden Semantiken ein präziser Zusammenhang besteht, wie es auch aus den klassischen Arbeiten zu erwarten wäre. Zuletzt präsentieren wir noch einige Anwendungen unserer abstrakten Theorie. Die übliche denotationelle Semantik, welche geordnete oder metrisierte Strukturen benutzt, ergibt sich als Spezialfall aus unseren Ergebnissen. Darüberhinaus geben wir neue Anwendungen an, die zum Beispiel rekursive Definitionen von Operationen, welche gewisse Gleichungen erfüllen, oder von Fraktalen betreffen.

Acknowledgments

I would like to express my deepest gratitude to my supervisor Jiří Adámek. Without his constant help, support and encouragement the results in this thesis could never have been developed. I am very grateful for the opportunity to work with him, for all he has done to help promote my research, for example by giving me the chance to present it at numerous conferences, and to meet many great scientists.

My warmest thanks also go to Jiří Velebil, with whom I very closely collaborated, for the countless discussions, the numerous diagram chasing sessions and for always suggesting to continue our research and not to get lost in the everyday tasks that so easily get you distracted.

I also give my best thanks to Larry Moss. His work was of great inspiration to this research and our discussion in Bloomington while I was visiting for a summer school let the pieces of the puzzle concerning algebraic semantics fall into their correct place for me. The subsequent collaboration was very fruitful and of great pleasure for me.

Heartfelt thanks go to all the people who have influenced this research with their work, who expressed encouragement and with whom I had the pleasure to discuss or share ideas: Peter Aczel, Stephen Bloom, Dominique Bourn, Neil Ghani, Tom Leinster, Federico De Marchi, Alex Simpson, Dana Scott, Tarmo Uustalu, to name just a few.

Many thanks also have to go to my colleagues at the Institute of Theoretical Computer Science for all their help and support: Katja Barkowsky, Jürgen Koslowski, Frank Rust and Dietmar Wätjen.

Last but not least, I would like to extend my gratitude to my friends and family; my parents, my children and especially to my loving wife Birgit. Her belief in me and her never ending support have often helped me a great deal through the ups and downs during the last five years.

This text has been typeset using Leslie Lamport's LaTeX and the diagrams were drawn using the XY-pic macros of Kristoffer H. Rose and Ross Moore.

Contents

A youth who had begun to read geometry with Euclid, when he had learnt the first proposition, inquired, "What do I get by learning these things?" So Euclid called a slave and said "Give him threepence, since he must make a gain out of what he learns."

Joannes Stobaeus, Four Books of Extracts, Sayings and Precepts (Extracts), 5th century AD.

1 Introduction

The greatest challenge to any thinker is
stating the problem in a way that will allow a solution.
Bertrand Russell.

Recursion arises everywhere in computer science. It allows for a clear, concise and finite description of complicated or potentially infinite objects or phenomena. Hence, it is a major ingredient of any programming language or formalism describing the work of an information system. For example, a programmer often defines a function using a recursive definition; that means a definition that is self-referential—the result of the function at an argument is given by computing the same function at a modification or on parts of the given argument. The paradigm example is the recursive definition of the factorial function:

$$f(n) := \mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n \cdot f(n-1)\,. \tag{1.1}$$

In process algebra one uses recursion to specify repeated or potentially infinite system behaviour by finite means. A very basic example is a process term

$$P := a.P \tag{1.2}$$

describing a system which can perform the action a repeatedly without ever terminating. Yet another example are recursive specifications of abstract data types such as

$$\mathsf{list} ::= \mathsf{element} \mid \mathsf{element.list} \tag{1.3}$$

where $\mathsf{element}$ is some given data type and list is the recursively specified datatype of lists with entries of type $\mathsf{element}$.

Of course, every recursive definition or self-referential specification immediately raises the question what its meaning should be. It is the task of *semantics*, an important branch of theoretical computer science, to answer such questions.

There are different ways to give semantics to a recursive definition. In practise, it is often sufficient to consider a so-called operational semantics, which gives a meaning by running a recursive program on an abstract machine of some sort. For example, an interpreter of a functional programming language that can run a program such as (1.1) provides its operational semantics. In process algebra, one uses the so-called structured operational semantics to provide a labelled transition system that behaves as specified by a process term such as (1.2). Any actual implementation of a data type of lists containing entries of a data type of elements in a programming language gives an operational semantics to the recursive data type specification (1.3).

In this thesis we shall be concerned with another flavour of semantics—denotational semantics—which assigns to a recursive definition a certain mathematical object, its meaning. Such a semantics is independent of a particular implementation of a program, and so it more easily serves as a basis for formal reasoning about recursive programs or recursively specified information systems.

The most common approach to denotational semantics considers ordered structures in which it is possible to obtain the semantics of some recursive definition as a join of its finite approximations. Another approach considers structures equipped with a complete metric. Here it is possible to obtain semantics of recursive definitions using a limit of a Cauchy sequence of its finite approximations. In both approaches fixed points of certain functions play a major rôle. In fact, in ordered structures one usually employs the fixed point theorem of Alfred Tarski, Stephen Kleene, and Bronislaw Knaster, see e.g. [Ta]. And in completely metrized structures one applies the fixed point theorem of Stefan Banach, see [Ban].

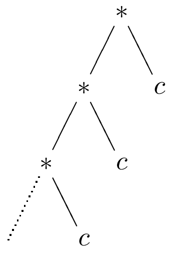
The starting point of our research is the classical work of Calvin Elgot and his collaborators [E, BE, EBT]. Their aim was to study the semantics of recursive computations at a purely algebraic level working without ordered or metrized structures. In [E] Elgot introduced *iterative theories*, which are algebraic theories in the sense of William Lawvere [La] that admit unique solutions of certain recursive specifications. Let us illustrate this on the simple theories given by operations without any equations. Suppose we have a signature Σ of operation symbols with prescribed arities in the set of natural numbers, i. e., $\Sigma = (\Sigma_n)_{n\in\mathbb{N}}$ is a sequence of sets, where Σ_n is the set of operation symbols of arity n. For example, a signature with a constant symbol c and a binary operation symbol $*$ has $\Sigma_0 = \{c\}$, $\Sigma_2 = \{*\}$, and $\Sigma_n = \emptyset$ else. For a given set of generators one can form Σ-terms over that set. Now consider a recursive system of formal equations

$$\begin{array}{rcl} x_0 & \approx & t_0(x_0,\dots,x_n,y_0,\dots,y_k) \\ & \vdots & \\ x_n & \approx & t_n(x_0,\dots,x_n,y_0,\dots,y_k) \end{array} \tag{1.4}$$

where $X = \{ x_0, \dots, x_n \}$ is a finite set of variables, $Y = \{ y_0, \dots, y_k \}$ is a set of parameters, and the t_i, $i = 1, \dots, n$, are Σ-terms over the disjoint union of X and Y. Iterative theories are defined by the property that every *guarded* system (1.4) can be solved uniquely. Guardedness here means that no term t_i, $i = 1, \dots, n$, is simply a single variable. One important example of an iterative theory is the theory T_Σ formed by all (finite and infinite) Σ-trees, i. e., rooted and ordered trees, where each inner node with n children is labelled by an n-ary operation symbol from Σ and leaves are either labelled by a constant symbol from Σ_0 or by an element from a set of generators. For example, for the signature with a constant symbol c and a binary symbol $*$ we can uniquely solve the formal equation

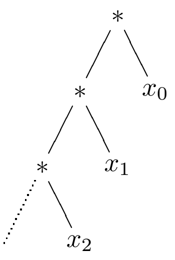
$$x \approx x * c$$

whose solution is, of course, the infinite Σ-tree



(1.5)

Another important iterative theory is the theory R_Σ, the subtheory of T_Σ formed by all *rational* Σ-trees, i. e., those Σ-trees which have (up to isomorphism) only a finite set of different subtrees—a description provided by Susanna Ginali [Gi$_2$]. For example the tree (1.5) is rational since its only subtree besides itself is the single-node tree labelled by c. But the tree



over an infinite set of generators containing all x_i, $i \in \mathbb{N}$, is not rational.

Rational trees are equivalently described as that class of Σ-trees arising as unfoldings of all guarded formal recursive equations (1.4) as infinite trees, see e.g. [C]. In fact, Calvin Elgot, Stephen Bloom and Ralph Tindell [EBT] proved that rational Σ-trees form the free iterative theory on a given signature. Similarly, the theory T_Σ of all Σ-trees is a free *completely* iterative theory on Σ, where completeness refers to the fact that all (not necessarily finite) systems of formal equations can be uniquely solved.

Algebraic theories are complicated objects to study and in fact, the original proof of Elgot and his coauthors is spread out over more than a hundred pages in the three papers [E, BE, EBT]. So it was a great step forward when Evelyn Nelson [N] and Jerzy Tiuryn [T] introduced iterative algebras for a signature to obtain a more easy approach to iterative theories. Recall first that an algebra for a signature Σ is a set A equipped with operations $\sigma_A : A^n \longrightarrow A$ for all operation symbols $\sigma \in \Sigma_n$, $n \in \mathbb{N}$. An iterative algebra A is a Σ-algebra in which every system of formal equations (1.4) has (for every interpretation of the parameters $y_0, \dots, y_k$ in A) a unique solution, i. e., there exists a unique n-tuple of elements of A that when plugged in for the variables on both sides of the equation turn the formal equations into actual identities. Classical algebras like groups or lattices are usually not iterative. However, there are enough interesting examples of iterative algebras. Denote by $T_\Sigma X$ the algebra of all Σ-trees over a set X of generators, i. e., Σ-trees whose leaves are labelled by constant symbols from Σ_0 or generators from X. Similarly, $R_\Sigma X$ is the algebra of rational Σ-trees over X. Both $T_\Sigma X$ and $R_\Sigma X$ are iterative algebras. Moreover, Evelyn Nelson proved that $R_\Sigma X$ is a free iterative algebra on X, and that the algebraic theory of free iterative algebras is the free iterative theory of Calvin Elgot.

The first, more mathematical part, of our work presented here concerns a generalization and extension of the classical work of Calvin Elgot and Evelyn Nelson using ideas from the theory of *coalgebras*. Coalgebras have only recently gained more attention from researchers in theoretical computer science. They allow a uniform investigation of state-based systems of different types at an abstract level, and they provide general methods for reasoning about systems. For example, sequential automata can be captured as coalgebras and, most prominently, labelled transition systems are coalgebras. Coalgebras are best studied using the language of category theory; the type of a class of systems to be captured as coalgebras is described by an endofunctor of Set, the category of sets and functions. For example, labelled transition systems are the coalgebras for the endofunctor $\mathcal{P}(\mathsf{Act} \times {\_})$, where Act is a fixed set of actions and $\mathcal{P}({\_})$ denotes the power set functor. A very important concept in the theory of coalgebras is that of a final coalgebra. Final coalgebras provide in many cases a semantics of behaviour of a state of a system. For example, for automata considered as coalgebras the final coalgebra consists of all formal languages over the input alphabet, and the finality principle assigns to every state of a given automaton the language accepted by it with the chosen state as the initial one. Unfortunately, for arbitrary labelled transition systems there can be no final coalgebra for cardinality reasons. However, for finitely branching labelled transition systems the final coalgebra exists and consists of (bisimilarity equivalence classes of) all possible behaviours of states of such transition systems. These behaviours can be described as equivalence classes of finitely branching extensional trees. For more details on this and other examples we refer the reader to our introductory Section 2.3.

For our work, a crucial observation is that for a signature Σ the set $T_\Sigma X$ of all Σ-trees over X is a final coalgebra for the endofunctor $H_\Sigma({\_}) + X$, where H_Σ is the canonical polynomial endofunctor associated to the signature Σ and $+$ denotes disjoint union of sets. The important message of our results is that the finality principle is sufficient to prove the classical results of Calvin Elgot. Furthermore, the fact that we work in a category theoretic setting using universal properties such as finality allows us to substantially generalize the classical results. Our proof is shorter than the classical one given by Elgot et al., and we believe that it unveils what mechanisms really are at work in the classical setting at a conceptual level.

Our overall assumptions are small indeed; we work with any endofunctor H of Set (or a more general category satisfying some rather mild side conditions) which has "enough final coalgebras", i. e., for every set X there exists a final coalgebra for $H({\_}) + X$. In the first part of our work we introduce completely iterative algebras for an endofunctor, where systems like (1.4), but not necessarily with a finite set of variables, have a unique solution. For a signature Σ, every algebra $T_\Sigma X$ of Σ-trees over a set X is completely iterative. Other examples stem from the realm of algebras on complete metric spaces. For example, the golden ratio φ can be described by the continued fraction

$$1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}} \,,$$

whence it is the unique real number from the interval $[1, 2]$ satisfying the equation

$$x = 1 + \frac{1}{x} \,.$$

Another example is the famous Cantor space c which is the unique non-empty closed subspace of the interval $[0, 1]$ such that the equation

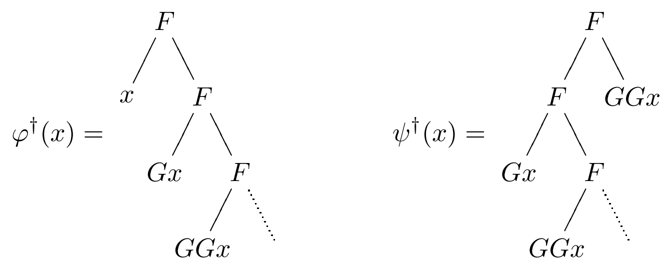
$$c = \frac{1}{3}c + \left(\frac{2}{3} + \frac{1}{3}c\right) \,, \tag{1.6}$$

holds, where we write $\frac{1}{3}c$ to mean $\{\, \frac{1}{3}x \mid x \in c \,\}$. The Cantor space arises as the unique solution of a formal equation in the completely iterative algebra formed by all non-empty closed subspaces of $[0, 1]$ metrized with the Hausdorff metric. As our first important result on completely iterative algebras we prove that a final coalgebra for $H({\_}) + X$ is precisely the same as a free completely iterative algebra on X. This characterization of final coalgebras as free algebras of some sort is a new and important result of our work. This result opens the door to our subsequent treatment of algebraic semantics using coalgebraic methods. But before that, following Evelyn Nelson's work, we show that free completely iterative algebras yield free completely iterative theories. Similarly, for every *finitary* endofunctor H of Set, i. e., H is determined by its action on finite sets, we formulate the notion of iterative algebra. Then we prove that every set X generates a free iterative algebra RX for H, and we show how to construct every RX from finite coalgebras. As a result of this construction it follows that the initial iterative algebra $R\emptyset$ gives a semantics of all states of finite systems described as coalgebras. For example, when sequential automata are considered as coalgebras, then $R\emptyset$ is the set of all regular languages, i. e., those languages accepted by finite automata. More generally, our construction yields for every polynomial endofunctor of Set the algebras of rational trees as expected. We consider this construction as another one of our main achievements. It is the key to almost all of the subsequent results on iterativity we present here and in other work. For example, we show that as in the classical setting free iterative algebras yield free iterative theories.

In the second part of this thesis we turn our attention towards applications for semantics of our previously developed theory. We show that our work provides a category-theoretic semantics of so-called recursive program schemes. Classically, a recursive program scheme (RPS) is a system of formal equations such as

$$\begin{array}{lcl} \varphi(x) & \approx & F(x, \varphi(Gx)) \\ \psi(x) & \approx & F(\varphi(Gx), GGx) \end{array} \tag{1.7}$$

where the operations φ and ψ are defined recursively from given operations F and G. The theory of such recursive program schemes is the topic of algebraic semantics, see for instance the book of Irène Guessarian [G]. Actually, one has to distinguish between the uninterpreted semantics of a recursive program scheme and the interpreted one. The uninterpreted semantics provides a meaning of recursive program schemes independently of the actual nature of the given operations. That means that the recursive program scheme is regarded as a purely syntactic construct and its classical uninterpreted semantics will give to each newly defined operation symbol a tree over the signature of givens obtained by unfolding the recursive definition. For example, the above RPS (1.7) has as its uninterpreted solution the infinite trees



(1.8)

Of course, much has been omitted here. For example, we have not explained why unfolding an RPS yields its solution, or even why the trees in (1.8) are a solution of the RPS (1.7). In order to do this one first needs to define *second-order substitution* of Σ-trees, i. e., substitution of trees for operation symbols, see [C]. Then one can formulate that a solution of an RPS is a fixed point with respect to this second-order substitution; much as solutions of systems of the form (1.4) are fixed points with respect to ordinary or first-order substitution of Σ-trees, i. e., substitution of trees for generators.

In the present thesis we shall formulate the notion of a (suitably generalized) recursive program scheme in a category theoretic setting using final coalgebras in lieu of terms or trees. As it turns out the notion of second-order substitution is elegantly expressed in our setting as a direct consequence of the universal property of free completely iterative theories. This easy but crucial observation allows us to formulate what a solution of an RPS is. We then prove that every guarded recursive program scheme (in our sense) has a unique solution. Guardedness here means that the right-hand sides of an RPS have their head symbols in the signature Σ of given operations. This is also called Greibach normal form in the classical setting. Again, our category theoretic result readily yields the classical one as a special case by working with polynomial endofunctors of Set. But our theory encompasses also applications that go beyond what can be done with the classical methods. For example, we are able to solve recursive program schemes which define operations satisfying equations like commutativity by encoding such extra requirements directly into the RPS. In the classical setting such additional requirements have to be treated separately.

In practise one is often more interested in the interpreted semantics of an RPS. This semantics considers an RPS with the givens from a signature Σ together with a suitable Σ-algebra A, whose operations $\sigma_A : A^n \longrightarrow A$, $\sigma \in \Sigma_n$, $n \in \mathbb{N}$, provide an interpretation of all the given function symbols. Here is the standard example in the subject. Let Σ be the signature of given operation symbols with a constant one, a unary symbol pred, a binary symbol $*$ and a ternary one ifzero. The interpretation we have in mind is the set $\mathbb{N}$ of 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

$$f(n) \approx \mathsf{ifzero}(n, \mathsf{one}, f(\mathsf{pred}(n)) * n)\,. \tag{1.9}$$

Then (1.9) is a recursive program scheme defining the factorial function. In general an interpreted RPS is supposed to define new operations on the algebra A in a canonical way such that the formal recursive definitions become valid identities in the given algebra. So by "suitable algebra" we mean, of course, one in which recursive program schemes can be given a semantics. For example, for the recursive program scheme (1.7) we are only interested in those Σ-algebras A, where $\Sigma = \{F, G\}$, in which the program scheme (1.7) has a *solution*, i. e., we can canonically obtain new operations φ_A and ψ_A on A so that the formal equations (1.7) become valid identities. Before we can actually get to an interpreted semantics the question we have to address is:

$$\text{What } \Sigma\text{-algebras are suitable for semantics?} \tag{1.10}$$

As we mentioned already, several answers have been proposed in the literature. In the classical theory one works with complete posets (cpo) in lieu of sets, see [G]. Here algebras have an additional cpo structure making all operations continuous. Another approach works with complete metric spaces. Here we have an additional complete metric making all operations contracting. 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 computation as a join (or limit, respectively) of finite approximations.

Also (completely) iterative algebras are suitable for semantics. However, they are not the unifying concept one would hope for. While avoiding extra structure they do not subsume continuous algebras on cpos which have *least* (but not necessarily unique) solutions of recursive equations.

The iteration theories of Stephen Bloom and Zoltán Ésik [BÉ] were introduced to overcome this problem, and iteration algebras, see [BÉ], Chapter 7, are algebras that admit canonical (but in general, non-unique) solutions of formal systems of equations.

Analyzing all the above types of algebras we find an interesting common feature which make continuous, metrizable, completely iterative and iteration algebras fit for use in semantics of recursive program schemes: these algebras allow for an evaluation of all Σ-trees. More precisely, for every continuous, metrizable or completely iterative algebra A we obtain a canonical map $T_\Sigma A \longrightarrow A$ which provides for any Σ-tree over A its result of evaluation in A. For completely iterative algebras this follows from the freeness of $T_\Sigma A$ as a completely iterative algebra. It is then easy to give semantics to recursive program schemes in A. For example, for (1.7) one can simply take the tree unfolding which yields the infinite trees of (1.8), and then for any argument $x \in A$ evaluate these infinite trees in A.

Actually, we do not need to be able to evaluate all infinite trees: all recursive program schemes unfold to *algebraic trees*, see [C]. And, as we have seen, another important subclass are the algebras $R_\Sigma X$ of rational trees over X. With this in mind, we can restate problem (1.10) more formally:

$$\begin{gathered}\text{What } \Sigma\text{-algebras have a suitable evaluation of all trees?}\\ \text{Or all rational trees?}\end{gathered} \tag{1.11}$$

The answer we provide in this thesis is: Σ-algebras carrying a certain additional structure providing canonical solutions of every system of formal equations (1.4). We have chosen to name these algebras (complete) *Elgot algebras* to honour Calvin Elgot whose work has been a great inspiration for us. In contrast to iterative algebras, in an Elgot algebra solutions need not be unique. Instead, there is an operation choosing a solution which is required to satisfy two simple and well-motivated axioms which stem canonically from Elgot's iterative theories. The first of these axioms states that solutions are stable under systematic renaming of the variables of X in system like (1.4). And the second axiom states that simultaneous recursion can be performed sequentially, very similar to what is known as Bekić-Scott law from the theory of least fixed points in complete partial orders. We prove that every free iterative algebra on a set X is also a free Elgot algebra on that set. Moreover, the axioms of Elgot algebras ensure that the evaluation map $R_\Sigma A \longrightarrow A$ obtained by the freeness of $R_\Sigma A$ behaves well with respect to substitution of rational trees for generators. This fact can be expressed in a mathematical precise way in the language of category theory: Elgot algebras for a signature Σ are precisely the Eilenberg-Moore algebras for the monad given by the free iterative theory R_Σ. Hence, one may say that Elgot algebras are precisely those algebras having a canonical evaluation of all rational trees.

Similarly, complete Elgot algebras give canonical solutions to every (not necessarily finite) system of formal equations, where the operation of assigning solutions satisfies the same two simple axioms as above. For a signature Σ we prove that a free complete Elgot algebra on X is precisely the same as a final coalgebra for $H_\Sigma(\_) + X$. And moreover, complete Elgot algebras are precisely the same as the Eilenberg-Moore algebras for the monad given by the free completely iterative theory T_Σ. More elementary, one may say that complete Elgot algebras for a signature Σ are precisely those Σ-algebras with a evaluation map $T_\Sigma A \longrightarrow A$ of all Σ-trees in A. Of course, we introduce (complete) Elgot algebras more generally for endofunctors H, and all our results are proved for this more general category theoretic setting. Basic examples of (complete) Elgot algebras include all continuous algebras, metrizable algebras, and, of course, all (completely) iterative algebras. Using structures for semantics that come with a choice of solutions satisfying certain axioms is not new. Our two axioms mentioned above are very similar to two of the axioms of iteration theories of Stephen Bloom and Zoltán Ésik [BÉ]. In fact, for a signature Σ with a distinguished constant symbol $\perp$ the rational trees form an iteration theory whose iteration algebras are certain Elgot algebras for Σ satisfying an extra extensionality property.

Once we have established complete Elgot algebras as a suitable class of algebras which can serve as interpretations of givens for an RPS we can present our interpreted RPS semantics. We prove that for every guarded RPS which is given together with a complete Elgot algebra A we can provide a canonical solution in A. Moreover, if A happens to be a completely iterative algebra, this canonical solution is unique. Finally, a fundamental result in the theory of recursive program schemes states that the uninterpreted and interpreted semantics of recursive program schemes are connected via the interpretation of givens in the algebra A:

$$\begin{tikzcd}
\text{Recursive program scheme} \arrow[r, squiggly, "\text{uninterpreted semantics}"] \arrow[dr, squiggly, "\text{interpreted semantics}"'] & \Sigma\text{-trees} \arrow[d, squiggly, "\text{evaluation}"] \\
& \text{Algebra } A
\end{tikzcd} \tag{1.12}$$

More detailed, to take first the tree-unfolding of a given recursive program scheme and then compute the resulting Σ-trees in A is the same as to compute the new operations provided by the interpreted semantics of the recursive program scheme. We give a mathematical precise formulation of this consistency between uninterpreted and interpreted RPS solutions, and we show how this can easily be derived in our abstract theory.

We believe that our results in this area generalize and extend the previous work on this topic. 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 function $f : [0,1] \longrightarrow [0,1]$ such that

$$f(x) = \frac{1}{4}\left(x + f\left(\frac{1}{2}\sin x\right)\right). \tag{1.13}$$

Another example: there exists a unique contracting function φ on the set of all non-empty closed subspaces of the Euclidean interval $[0,1]$ satisfying

$$\varphi(x) = \frac{1}{3}\varphi(x) \cup \left(\frac{2}{3} + \frac{1}{3}x\right), \tag{1.14}$$

for every non-empty closed subspace x of $[0,1]$.

Finally, we also provide examples of recursive program schemes and their solutions which cannot be treated within the classical theory: recursive definitions of operations satisfying equations like commutativity.

To sum up, this thesis contributes substantially to a new coalgebraic perspective to recursion which has been developed by researchers in recent years. Our approach of using the theory of coalgebras and the language of category theory to study recursion does not only produce generalizations of well-known results. It also provides new tools and offers new insights into the general mechanisms at work in the semantics of recursion at a concise and conceptual level. Furthermore, we claim that our abstract categorical approach allows to bring several well-known approaches to semantics of recursion under one roof. And, as we have seen, our work is starting to produce new results in semantics.

The Structure of this Text

This summary is structured as follows: in Section 2 we will briefly recall some notions and results from category theory which are perhaps less familiar; in particular we discuss locally finitely presentable categories and *monads*. For the convenience of the reader we also provide a bit of introduction to the theories of algebras and coalgebras for an endofunctor. The material of the papers summarized in this document begins in Section 3. We study first iterative and completely iterative algebras. This section presents the results of the first parts of our papers [M_1] and [AMV_2], the latter of which is joint work with Jiří Aámek and Jiří Velebil.

Then we start to study the monads of free (completely) iterative algebras and we characterize in Section 4 the Eilenberg-Moore algebras for these monads—they are precisely (complete) Elgot algebras. We argue that Elgot algebras capture important aspects of different types of algebras commonly used in semantics of recursion. In particular, we show that continuous algebras and algebras on complete metric spaces are complete Elgot algebras. These are the results of [AMV_3], which is joint work with Jiří Aámek and Jiří Velebil, again. In Section 5, we show that much more complicated systems of recursive definitions than the ones of Section 3 can be solved uniquely in (completely) iterative algebras. These results are the basis for the work in Section 6 where the monads of free (completely) iterative algebras are shown to be the free (completely) iterative monads generated by an endofunctor. So Sections 5 and 6 present the second parts with the main results of the papers [M_1, AMV_2] and the results of [M_2]. While these results are mathematically quite pleasing they also are an important step towards our semantics of (generalized) recursive program schemes in Section 7, which is the topic of the joint paper with Larry Moss [MM]. In fact, the universal property of free completely iterative monads yields as a special case the so-called second-order substitution, which will allow us to define what a solution of a recursive program scheme in our abstract setting is. We will then present the uninterpreted as well as an interpreted semantics of recursive program schemes. Our main results in Section 7 are that every guarded recursive program scheme can be given a unique uninterpreted solution. Furthermore, using complete Elgot algebras as interpretations of givens we provide a canonical interpreted semantics, and we show that interpreted and uninterpreted solutions are consistent with each other. Finally, we illustrate our results with several applications.

We conclude this summary with some discussion of presently ongoing and directions for future research in the last Section 8.

Related Work

As we have mentioned already it was the idea of Calvin Elgot to study the semantics of recursive definitions at a purely algebraic level. He introduced iterative algebraic theories in [E], and later he proved with his coauthors that free iterative theories exist [BE] and are given by rational trees [EBT], see also the work of Susanna Ginali [Gi_1, Gi_2] for the first proof of this fact. Similarly, free completely iterative theories exist and are given by all infinite trees.

Iterative algebras were introduced and studied by Jerzy Tiuryn [T] and Evelyn Nelson [N] with the aim of providing a more easy approach to Elgot's iterative theories. Nelson proved that rational trees yield free iterative algebras and that those give a free iterative theory.

The first categorical accounts of infinite trees as monads of final coalgebras appear independently and almost at the same time in the work of Larry Moss [Mo_1], of Neil Ghani, Christoph Lüth, Federico De Marchi and John Power [$GLMP_1$, $GLMP_2$], and of Peter Aczel, Jiří Adámek and Jiří Velebil [AAV]. Furthermore, in [Mo_1] and [AAV] it is proved that those monads are completely iterative. And in [AAMV] we established the universal property of the free completely iterative monads. The categorical treatment of rational trees and iterative theories started with our work [AMV_1]. We proved there that every endofunctor of Set admits a free iterative monad and we provided a coalgebraic construction of this monad. In [GLM_1] the authors gave a general construction based on our ideas that allows to obtain (monads of) other syntactic objects than rational trees with a coalgebraic construction.

The axioms of (complete) Elgot algebras as presented in Section 4 below were inspired by the axioms of iteration theories of Stephen Bloom and Zoltán Ésik [BÉ]. Other approaches using solutions and axioms are traced monoidal categories of André Joyal, Ross Street and Dominic Verity [JSV], see the traced cartesian categories in [Ha] for the connection, or fixed-point theories for domains, see the work of Samuel Eilenberg [Ei] or Gordon Plotkin [P], etc.

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

Basic properties of infinite trees are presented by Bruno Courcelle in [C]. Our work [AAMV, M_1] gives a categorical description of second-order substitution of infinite trees which we use in order to formulate solutions of recursive program schemes abstractly.

Neil Ghani, Christoph Lüth and Federico De Marchi [GLM_2] 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 connection to (generalized) second-order substitution as presented in [AAMV, M_1] is new in our work.

Complete metric spaces as a basis for the semantics of recursive program schemes have been studied by André Arnold and Maurice Nivat, see e. g. [AN]. Stephen 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 Pierre America and Jan Rutten [ARu] and by Jiří Adámek and Jan Reitermann [ARe]. We build on this with our treatment of self-similar objects. These have also recently been studied in a categorical framework by Tom Leinster, see [Le_1, Le_2, Le_3].

2 Preliminaries

We will need to use some very simple notions of category theory,
an esoteric subject noted for its difficulty and irrelevance.
Moore and Seiberg [MS]

In this section we collect most of the basic notions and standard results used frequently throughout this summary. We assume that the reader is acquainted with some basics of category theory like categories, functors, natural transformations, limits, colimits, etc., see any standard text on category theory, e.g. the excellent textbook [ML] by Saunders MacLane. Everything else necessary to understand the work in the subsequent sections shall be mentioned here. This section is not thought as a general introduction to any of the theories treated in the following subsections. All we want to do here is to provide a little bit of intuition for the basic material used in our subsequent work, set up notation and recall the usual results. We will mostly not give proofs but provide references where arguments are not obvious. Readers familiar with category theory, algebras and coalgebras should skip this section and start reading Section 3 immediately.

2.1 Locally Finitely Presentable Categories

For some of our work it will be necessary to work in a category in which it makes sense to speak about a "finite object" in the same way as one can talk about finite sets in the category Set of (small) sets and functions. Moreover, we shall be interested in working with finitary endofunctors, i. e., those functors which are determined by their action on "finite objects". Such an environment is provided by locally finitely presentable categories in the sense of Peter Gabriel and Friedrich Ulmer [GU]. Before we give the definition let us recall the notions of filtered diagram and finitely presentable object.

Definition 2.1.

(i) A category $\mathcal{D}$ is called *filtered* if each finite subcategory has a cocone in $\mathcal{D}$, or more precisely

(a) $\mathcal{D}$ is non-empty,

(b) for each pair D_1, D_2 of objects of $\mathcal{D}$ there exists a cospan in $\mathcal{D}$, i.e, an object D and two morphisms $f_i : D_i \longrightarrow D$, $i = 1, 2$, and

(c) for each pair of parallel morphisms $f, g : D \longrightarrow D'$ in $\mathcal{D}$ there exists a coequating morphism, i. e., a morphism $h : D' \longrightarrow D''$ in $\mathcal{D}$ such that $h \cdot f = h \cdot g$.

(ii) A *filtered diagram* is a diagram $D : \mathcal{D} \longrightarrow \mathcal{A}$ whose scheme $\mathcal{D}$ is a filtered category, and a *filtered colimit* is a colimit of a filtered diagram.

(iii) A functor $F : \mathcal{A} \longrightarrow \mathcal{B}$ is called *finitary* provided that it preserves filtered colimits.

(iv) An object X of a category $\mathcal{A}$ is called *finitely presentable* if its covariant hom-functor $\mathcal{A}(X, -)$ is finitary.

Notation 2.2. For two categories $\mathcal{A}$ and $\mathcal{B}$ we shall denote by

$$[\mathcal{A}, \mathcal{B}] \qquad \text{and} \qquad \mathsf{Fin}[\mathcal{A}, \mathcal{B}]$$

the category of all endofunctors and of finitary ones, respectively.

Remark 2.3. The following more explicit characterization of finite presentability of an object X will be used frequently in our work: X is finitely presentable if and only if for any filtered colimit C and any morphism $f : X \longrightarrow C$, there exists an essentially unique factorization through an injection of the colimit. More detailed, there exists an injection $c_i : C_i \longrightarrow C$ of the colimit and a morphism $f' : X \longrightarrow C_i$ such that the triangle

$$\begin{tikzcd} & C_i \arrow[d, "c_i"] \\ X \arrow[ur, "f'"] \arrow[r, "f"'] & C \end{tikzcd}$$

commutes, and furthermore, whenever two factorizations f' and f'' of f through any colimit injection $c_j : C_j \longrightarrow C$ satisfy $c_j \cdot f' = c_j \cdot f''$, then f' and f'' can be coequated in the diagram, i. e., there exists a morphism $c_{jk} : C_j \longrightarrow C_k$ in the diagram such that $c_{jk} \cdot f' = c_{jk} \cdot f''$.

Next, we list some examples of finitely presentable objects. More examples can be found in [AR].

Examples 2.4.

(i) A set X is finitely presentable in Set if and only if it is a finite set.

(ii) Similarly, in Pos the category of partially ordered sets and order preserving maps an object is finitely presentable if and only if it is a finite poset.

(iii) A group is finitely presentable in the category Grp of groups and their homomorphisms if and only if it can be presented by finitely many generators and finitely many equations. More generally, an object in a variety of finitary (many-sorted) algebras is finitely presentable if and only if it can be presented by finitely many generators and equations.

(iv) Consider the category CPO of complete partial orders and continuous functions between them. So objects of CPO are partially ordered sets (not necessarily with a least element) in which every ascending chain has a join, and morphisms are functions preserving such joins. In CPO, only the empty cpo is finitely presentable.

The following result about finitely presentable objects will often be used, see [AR], Proposition 1.3.

Proposition 2.5. *A finite colimit of finitely presentable objects is finitely presentable.*

Definition 2.6. A category $\mathcal{A}$ is called *locally finitely presentable* (or, *lfp*, for short) if

(i) it is cocomplete,

(ii) it has (up to isomorphism) only a set of finitely presentable objects, and

(iii) every object of $\mathcal{A}$ is a filtered colimit of finitely presentable objects.

Examples 2.7.

(i) Set is lfp since every set is a directed union of its finite subsets, and the collection of all finite sets is up to isomorphism a (countable) set.

(ii) Pos, Grp, and every variety of finitary (many-sorted) algebras are lfp.

(iii) CPO is not lfp.

(iv) A poset considered as a category is lfp if and only if it is a complete algebraic lattice.

Next, we list properties of lfp categories used in this summary. We omit all proofs as they can be found in [AR].

Notation 2.8. For any lfp category $\mathcal{A}$ we denote by $\mathcal{A}_{\mathrm{fp}}$ the full subcategory of $\mathcal{A}$ given by (a representing set of) finitely presentable objects.

Proposition 2.9. *Every object A of an lfp category $\mathcal{A}$ is a colimit of the canonical filtered diagram*

$$D_A : \mathcal{A}_{\mathrm{fp}}/A \longrightarrow \mathcal{A}, \qquad (X \longrightarrow A) \longmapsto X\,,$$

of finitely presentable objects, i. e., $\mathcal{A}_{\mathrm{fp}}$ is a dense subcategory.

Theorem 2.10. *Any lfp category is complete and well-powered as well as cocomplete and cowell-powered.*

Theorem 2.11. (Adjoint Functor Theorem)
A functor between lfp categories is a right adjoint if and only if it preserves limits and filtered colimits.

Proposition 2.12. *Let $\mathcal{A}$ and $\mathcal{B}$ be lfp categories.*

(i) The categories $[\mathcal{A}_{\mathrm{fp}}, \mathcal{B}]$ and $\mathsf{Fin}[\mathcal{A}, \mathcal{B}]$ are equivalent.

(ii) For two lfp categories $\mathcal{A}$ and $\mathcal{B}$ the category $\mathsf{Fin}[\mathcal{A}, \mathcal{B}]$ is lfp.

In fact, the first item follows from the fact that $\mathcal{A}$ is a free cocompletion under filtered colimits of $\mathcal{A}_{\mathrm{fp}}$.

Remark 2.13. Analogously to lfp categories there exists the concept of locally λ-presentable category, for λ a regular cardinal, see [AR]. A category $\mathcal{A}$ is locally λ-presentable if it is cocomplete and has (up to isomorphism) a set of λ-presentable objects such that any object of $\mathcal{A}$ is a λ-filtered colimit of objects from that set.

Here the notions of λ-filtered colimit and λ-presentability are defined as expected: a category $\mathcal{D}$ is called λ-filtered if every subcategory with less than λ morphisms has a cocone in $\mathcal{D}$, colimits of diagrams whose domain is λ-filtered are called λ-filtered colimits, etc. Notice that a functor F is called λ-*accessible* if it preserves λ-filtered colimits, and F is called *accessible* if it is λ-accessible for some regular cardinal λ. Observe that Set is locally λ-presentable for every regular cardinal λ. Recall that an endofunctor $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ is called *bounded* if there exists a cardinal number λ such that for every set X any element $x \in HX$ lies in the image of Hm for some set $m : Y \hookrightarrow X$ of X of cardinality less that λ. Accessible endofunctors of Set are precisely the bounded ones, see [AT], Proposition III.4.3.

In this text we will not work with arbitrary locally λ-presentable categories. But we shall often mention and use accessible endofunctors (of Set).

2.2 Algebras

In this subsection we shall recall some basics on algebras for an endofunctor. Their purpose within our work is to provide a categorical framework for the notion of a set equipped with operations. We will provide here enough intuition to make this precise. In Sections 3 and 4 we will study certain algebras suitable for semantics of recursive programs, i.e., algebras in which it is possible to define new operations from given ones by means of a recursive specification.

Another view on algebras for an endofunctor is as implementations of abstract data types. More precisely, an *initial* algebra provides a canonical semantics of an abstract data type whose constructors are described by means of a suitable endofunctor of a category.

Definition 2.14. Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be an endofunctor of a category $\mathcal{A}$. By an H-*algebra* we mean a pair (A, a) where A is an object of $\mathcal{A}$, the *carrier* of the algebra, and

$$a : HA \longrightarrow A$$

is a morphism of $\mathcal{A}$, the *structure* of the algebra.

An *(H-algebra) homomorphism* from an algebra (A, a) to an algebra (B, b) is a morphism $h : A \longrightarrow B$ of $\mathcal{A}$ preserving the algebraic structure, i. e., such that the square

$$\begin{tikzcd} HA \arrow[r, "a"] \arrow[d, "Hh"'] & A \arrow[d, "h"] \\ HB \arrow[r, "b"'] & B \end{tikzcd} \tag{2.15}$$

commutes.

Notation 2.15. Observe that the identity on a carrier of an algebra is a homomorphism and that homomorphisms compose. Thus, H-algebras and their homomorphisms organize themselves in a category which we will denote by

$$\mathsf{Alg}\, H\,.$$

Remark 2.16. An important concept is that of an initial algebra. An H-algebra (I, i) is initial if for every H-algebra (A, a) there exists precisely one homomorphism from I to A, i. e., (I, i) is an initial object of $\mathsf{Alg}\, H$. Notice that, being a colimit, an initial algebra is uniquely determined up to isomorphism.

Examples 2.17.

(i) Consider the endofunctor of Set given by $HX = X + 1$. Algebras for H are unary algebras with a constant, and homomorphisms are maps preserving the constant and the unary operation. An initial algebra is the algebra $\mathbb{N}$ of natural numbers together with the constant 0 and the successor function. Notice that the elements of the initial algebra can be thought of as being build up inductively starting with the constant, then applying formally the unary operation to form new elements. In fact, the initiality of $(\mathbb{N}, 0, \mathsf{succ})$ can be understood as induction principle: to define a function f from $\mathbb{N}$ to a set A it suffices to specify its value at 0, i. e., an element $a = f(0)$, and a function $\alpha : A \longrightarrow A$ which will tell us that $f(n+1) = \alpha(a')$ if we already have defined $f(n) = a'$ for some natural number n. Indeed, (A, a, α) yields an H-algebra. Thus, by the initiality there exists a function $f : \mathbb{N} \longrightarrow A$ and the fact that it is an algebra homomorphism is equivalent to saying that $f(0) = a$ and $f(n+1) = \alpha(f(n))$, for any natural number $n \geq 1$, hold. The uniqueness of f yields an induction proof principle: to prove that two functions $f, g : \mathbb{N} \longrightarrow A$ are equal it suffices to establish that both f and g are algebra homomorphisms: then $f(0) = g(0)$ and whenever $f(n) = g(n)$, then also $f(n+1) = \alpha(f(n)) = \alpha(g(n)) = g(n+1)$. Thus, $f = g$ as desired.

(ii) Binary algebras and their homomorphisms are precisely the algebras for the set endofunctor given by $HX = X \times X$. In this case the initial algebra is carried by the empty set.

(iii) Lists. Consider the functor on Set given by $HX = 1 + S \times X$ for a fixed set S. The H-algebras are algebras A with a constant and unary operations α_s for each $s \in S$—the latter can, of course, be presented as one map $\alpha : S \times A \longrightarrow A$. Homomorphisms are, again, the structure preserving maps. An initial algebra is given by the set S^* of finite lists over S with the constant given by the empty list and with the unary operations given by concatenation:

$$\begin{array}{rcl} \mathsf{cons} : S \times S^* & \longrightarrow & S^* \\ (s, \langle s_1, \ldots, s_n \rangle) & \longmapsto & \langle s, s_1, \ldots, s_n \rangle . \end{array}$$

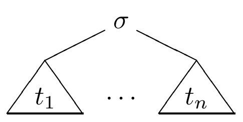
As before the initiality gives rise to an induction principle, i. e., the initiality is precisely the fact that functions out of S^* can be uniquely defined by structural induction. We leave it to the reader to work out the details.

(iv) Trees.[1] For two sets of labels L and M consider the functor $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ defined by $HX = L \times X \times X + M$. Algebras of H have a binary operation for each element $l \in L$ and constants for each $m \in M$, and homomorphisms preserve these operations and constants. An initial H-algebra is the algebra T of all binary trees with inner nodes labelled by elements of L and leaves labelled by elements of M. The algebra structure $L \times T \times T + M \longrightarrow T$ assigns to an element of M the single-node tree labelled by that element, and to a triple (l, t_1, t_2) the binary tree obtained by joining the trees t_1 and t_2 with a common root node which is labelled by l.

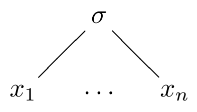
(v) The previous examples (i) to (iv) are subsumed by general algebras for a signature. Let Σ be a signature (or, ranked alphabet), i. e., Σ gives for each natural number n a set Σ_n of operation symbols of *arity* n. We form the polynomial functor $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$ associated to Σ by defining

$$H_\Sigma X = \Sigma_0 + \Sigma_1 \times X + \Sigma_2 \times X^2 + \cdots \tag{2.16}$$

on objects and similarly on morphisms. It is not difficult to see that $\mathsf{Alg}\, H_\Sigma$ is the category of Σ-algebras, i. e., general algebras for the signature Σ, and their homomorphisms. An initial algebra is the algebra F_Σ of all finite Σ-trees (or, Σ-terms), i. e., finite ordered trees where each node with n children is labelled by an n-ary operation symbol. The algebra structure map $\varphi_\Sigma : H_\Sigma F_\Sigma \longrightarrow F_\Sigma$ is given by tree tupling, more precisely, on the n-th coproduct component of $H_\Sigma F_\Sigma$, φ_Σ assigns to a tuple $(\sigma, t_1, \ldots, t_n)$ where σ is an n-ary operation symbol and $t_1, \ldots, t_n$ are finite Σ-trees from F_Σ the tree



Finally, notice that the elements of $H_\Sigma X$ for a set (of variables) X can be understood as *flat trees*, i. e., trees whose root node is labelled by an n-ary operation symbol and where this root has n-children which are leaves labelled by elements from X:



We will sometimes use the notation $(\sigma, \vec{x})$ for elements of $H_\Sigma X$.

[1]Trees are throughout be understood as rooted ordered ones up to isomorphism.

(vi) Algebras with operations that satisfy certain equations can sometimes be expressed as algebras for an endofunctor. For example, the algebras with a binary commutative operation are the algebras for the functor $\mathcal{P}_2$ on Set assigning to a set X the set of unordered pairs

$$\mathcal{P}_2 : X \longmapsto \{ \{ x, y \} \mid x, y \in X \} .$$

Of course, H-algebra homomorphisms are precisely the morphisms preserving the binary operation. The initial algebra is carried by the empty set.

(vii) More generally, algebras for finitary endofunctors of Set can be understood as algebras for a signature satisfying equations. Recall that a functor $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ is finitary (i. e., it preserves filtered colimits, see Definition 2.1) if and only if 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, which are fully described by their kernel equivalence whose pairs can be presented in the form of so-called *basic equations*

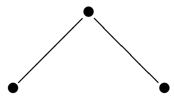
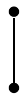
$$\sigma(x_1, \dots, x_n) = \varrho(y_1, \dots, y_m)$$

for $\sigma \in \Sigma_n$, $\varrho \in \Sigma_m$ and $(\sigma, \vec{x}), (\varrho, \vec{y}) \in H_\Sigma X$ for some set X including all x_i and y_j. It is not difficult to prove that the algebras for H are precisely the Σ-algebras satisfying the basic equations given by ε. The initial algebra I of H is given by the quotient $I_\Sigma/{\sim}$ where $\sim$ is the smallest congruence on I_Σ such that $I_\Sigma/{\sim}$ is an H-algebra. More explicitly, for finite Σ-trees s and t we have $s \sim t$ if and only if t can be obtained from s by finitely many applications of the basic equations describing ε. For example, the functor H which assigns to a set X the set $\{ \{ x, y \} \mid x, y \in X \} + 1$ of unordered pairs of X plus an extra element is a quotient of $H_\Sigma X = X \times X + 1$ expressing one binary operation b and a constant c, where ε_X is presented by commutativity of b; i. e., by the basic equation $b(x, y) = b(y, x)$. And I is the algebra of all finite unordered binary trees.

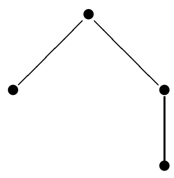
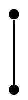
(viii) It follows from (vii) above that a finitary variety of Σ-algebras for a signature forms a category of algebras for a finitary endofunctor of Set if and only if the variety can be defined by basic equations. We already saw in (vi) above algebras with a commutative binary operation as algebras for the endofunctor $\mathcal{P}_2$. However, the category of semigroups (i. e., algebras with one associative binary operation) does not form a category of algebras for a finitary set endofunctor.

(ix) Consider the finite power set functor $\mathcal{P}_{\mathrm{fin}}$ assigning to a set X the set of its finite subsets. We do not give an explicit description of $\mathsf{Alg}\,\mathcal{P}_{\mathrm{fin}}$, but we describe the initial algebra. The functor $\mathcal{P}_{\mathrm{fin}}$ is given as a quotient $\varepsilon : H_\Sigma \longrightarrow \mathcal{P}_{\mathrm{fin}}$ where Σ has for each natural number n a unique n-ary operation symbol, i. e., $H_\Sigma X = \coprod_n X^n$, and $\varepsilon_X(x_1, \dots, x_n) = \{ x_1, \dots, x_n \}$ for every n-tuple of elements of X. Thus, the initial $\mathcal{P}_{\mathrm{fin}}$-algebra is given by a quotient of the initial H_Σ-algebra modulo basic equations as in (vii) above. We give alternative descriptions here.

An initial $\mathcal{P}_{\mathrm{fin}}$-algebra I is carried by the set of all finite constructive sets. Then $I = \mathcal{P}_{\mathrm{fin}} I$ so that the structure of the initial algebra is the identity map. Before we give another (isomorphic) description of I we need to recall the following notions. A *bisimulation* between two unordered trees s and t is a relation $\sim$ between their sets of nodes with the following property: if $n \sim m$ are related nodes of s and t, respectively, then for each child n' of n there exists a child m' of m with $n' \sim m'$, and vice versa, for each child m' of m there exists a child n' of n with $n' \sim m'$. The trees s and t are called *bisimilar* if there exists a bisimulation between them relating their root nodes. For example, the two trees

 and  (2.17)

are bisimilar, whereas

 and 

are not. An unordered tree t is called *strongly extensional* if for each node two distinct subtrees rooted at children of that node are never bisimilar. Notice that, the greatest bisimulation on a tree t is always an equivalence relation and the quotient of t modulo this equivalence is a strongly extensional tree. For example, consider the two trees in (2.17) above. The left-hand one is not strongly extensional, but the right-hand one is and in fact, it is the stronly extensional quotient of the left-hand tree.

Now coming back to the initial algebra I of $\mathcal{P}_{\mathrm{fin}}$, one can prove that I consists of all unordered finite strongly extensional trees.

Remark 2.18. Despite the fact, that all our previously presented examples are algebras for set endofunctors it is useful to work in an abstract category $\mathcal{A}$. Firstly, in our proofs and constructions it will be clearer what concepts and structure really have to be employed to make things work. Secondly, there may later arise applications that call for the use of other ground categories than Set.

In fact, we shall later see applications in semantics of computation where we deal with algebras (e. g. for a signature) having extra structure. For example, continuous algebras are carried by a complete partial order rather that just a set. Likewise, one may be interested to work with algebras on complete metric spaces. Those algebras can conveniently be treated by considering algebras for a functor on the category of complete partial orders and continuous maps or the category of complete metric spaces and non-expanding maps, respectively.

Next we give some basic results on algebras, which we shall frequently use throughout our work. First of all, constructions like products of algebras etc. can be performed on the level of the underlying category $\mathcal{A}$. Recall that a functor $F : \mathcal{A} \longrightarrow \mathcal{B}$ is said to *create* limits if for any diagram $D : \mathcal{D} \longrightarrow \mathcal{A}$ and for every limiting cone $(L, (\ell_i)_{i\in\mathcal{D}})$ of $F \cdot D$ there exists a unique cone $(A, (a_i)_{i\in\mathcal{D}})$ with $FA = L$ and $Fa_i = \ell_i$, for all $i \in \mathcal{D}$, and this cone is a limit of D. For the forgetful functor $U : \mathsf{Alg}\, H \longrightarrow \mathcal{A}$ creation of limits means that for each diagram D of algebras and homomorphisms the limit L of the diagram $U \cdot D$ in $\mathcal{A}$ can uniquely be equipped with an algebra structure so that all limit projection become homomorphisms and the resulting cone is a limit in $\mathsf{Alg}\, H$ of D.

Theorem 2.19. *The forgetful functor $U : \mathsf{Alg}\, H \longrightarrow \mathcal{A}$ creates all limits and all colimits that H preserves.*

In general, colimits of algebras are not formed on the level of $\mathcal{A}$, e. g., coproducts of binary algebras are not carried by the disjoint union of the carriers. However, there are results on the cocompleteness of $\mathsf{Alg}\, H$, see [Li$_2$, AK$_1$] or [AK$_2$].

The next result states that an initial algebra is the least fixed point of H, or the least solution of the recursive domain equation

$$HX \simeq X\,.$$

Lemma 2.20. (Lambek [L])
If (I, i) is an initial H-algebra, then its structure is an isomorphism.

Proof. Clearly, (HI, Hi) is an H-algebra. Thus there exists a unique homomorphism j from (I, i) to (HI, Hi). Since $i : HI \longrightarrow I$ is a homomorphism, we obtain the commutative diagram

$$\begin{tikzcd}
HI \arrow[r, "i"] \arrow[d, "Hj"'] & I \arrow[d, "j"] \\
HHI \arrow[r, "Hi"] \arrow[d, "Hi"'] & HI \arrow[d, "i"] \\
HI \arrow[r, "i"'] & I
\end{tikzcd}$$

Thus, we have $i \cdot j = id$ by the initiality, because id is the only homomorphism from the initial algebra to itself. The commutativity of the upper square now gives $j \cdot i = Hi \cdot Hj = id$. □

Another very important concept is that of a free algebra on a given object of generators.

Definition 2.21. Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be an endofunctor, and let X be an object of $\mathcal{A}$. An H-algebra (FX, φ_X) together with a morphism $\eta_X : X \longrightarrow FX$ is called a *free H-algebra on X* if for every H-algebra (A, a) and every morphism $f : X \longrightarrow A$ there exists a unique homomorphic extension $\widehat{f} : FX \longrightarrow A$, i. e., a unique

H-algebra homomorphism $\widehat{f} : (FX, \varphi_X) \longrightarrow (A, a)$ with $\widehat{f} \cdot \eta_X = f$:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & FX \arrow[d, "\widehat{f}"'] & HFX \arrow[l, "\varphi_X"'] \arrow[d, "H\widehat{f}"] \\
& A & HA \arrow[l, "a"]
\end{tikzcd}$$

Remark 2.22.

(i) In a category $\mathcal{A}$ with finite coproducts, it is easy to show that for every object X an initial algebra for the functor $H(\_) + X$ is precisely a free H-algebra on X. More detailed, if FX is a free H-algebra on X, then its algebraic structure $\varphi_X : HFX \longrightarrow FX$ and the universal arrow $\eta_X : X \longrightarrow FX$ yield the structure of the initial algebra for $H(\_) + X$, and conversely.

Observe that the free H-algebra $F0$ on the initial object of $\mathcal{A}$ is precisely the initial H-algebra.

(ii) Functors H having free algebras on every object X are called *varietors* in [AT]. For a varietor the assignment $X \longmapsto FX$ of a set to a free algebra on that set yields a left-adjoint to the forgetful functor $U : \mathsf{Alg}\, H \longrightarrow \mathcal{A}$. All functors of Example 2.17 are varietors of Set. However, not every set functor is a varietor. The (unbounded) power set functor $\mathcal{P} : \mathsf{Set} \longrightarrow \mathsf{Set}$ does not have an initial algebra, whence no free algebras. In fact, there can be no fixed point of $\mathcal{P}$ due to the famous Cantor Theorem.

(iii) Varietors do not enjoy nice closure properties. For example, for $\mathcal{A} = \mathsf{Set}$ varietors need not compose, nor need a coproduct of two varietors be a varietor, see [AT], IV.4.4. Therefore, one often works with a more nicely behaved subclass of varietors, for example finitary (or, more generally, accessible) functors of Set, see Theorem 2.25 below.

Examples 2.23.

(i) For a polynomial endofunctor $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$ induced by a signature a free algebra on a set X is carried by the set $F_\Sigma X$ of all finite Σ-trees over X, i. e., Σ-trees where leaves can be labelled either by a generator from X or by a constant symbol from Σ. We leave it to the reader to work out the special cases for the functors of Examples 2.17(i)–(iv). Observe that the freeness of $F_\Sigma X$ specializes to substitution of terms for variables. In fact, let X and Y be sets of variables. Then any map $s : X \longrightarrow F_\Sigma Y$ gives a substitution of variables of X by finite Σ-trees over Y. The unique induced homomorphism $\widehat{s} : F_\Sigma X \longrightarrow F_\Sigma Y$ performs for any finite Σ-tree t in $F_\Sigma X$ the substitution s, i. e., every leaf labelled by $x \in X$ is replaced by $s(x) \in F_\Sigma Y$.

Notice also that for any Σ-algebra A we obtain a canonical homomorphism $\alpha : F_\Sigma A \longrightarrow A$ extending the identity on A. It provides evaluations of all finite Σ-trees over A in the algebra A. In fact, one easily verifies that α is obtained by canonically extending the computation of all flat Σ-trees over A which is provided by the algebraic structure $a : H_\Sigma A \longrightarrow A$.

(ii) For an arbitrary finitary functor H of Set which comes as a quotient $\varepsilon : H_\Sigma \longrightarrow H$ we have described an initial algebra in Example 2.17(vii). Analogously, a free algebra FX on a set X is a quotient $F_\Sigma X/{\sim_X}$ of the free H_Σ-algebra on X. In fact, notice that we obtain a natural transformation $\varepsilon' = \varepsilon + id : H_\Sigma(\_) + X \longrightarrow H(\_) + X$ exhibiting $H(\_) + X$ as a quotient of the polynomial functor $H_\Sigma(\_) + X$. Observe that the basic equations for ε' are precisely the same as those for ε, and recall Remark 2.22(i) to obtain FX as the quotient of $F_\Sigma X$ modulo basic equations; more detailed, $s \sim_X t$ holds for Σ-trees s and t over X if s is obtained from t by application of finitely many basic equations provided by ε.

(iii) A free $\mathcal{P}_2$-algebra on a set X where $\mathcal{P}_2$ is the unordered pair functor from Example 2.17(v) is carried by the set of binary unordered trees with leaves labelled in X. The algebra structure is given by tree pairing.

(iv) A free $\mathcal{P}_{\mathrm{fin}}$-algebra on a set X is carried by the set of finitely branching strongly extensional finite trees with leaves partially labelled in X.

A free H-algebra on an object X can in many categories be inductively constructed. We describe this first for the case of an endofunctor H on Set. For a set X we define by transfinite induction the following chain indexed by ordinal numbers:

$$\begin{array}{lrcl}
\text{Initial step:} & H_0X & = & X \\
 & h_{0,1} & \equiv & H_0X = X \xrightarrow{\mathsf{inr}} HX + X = H_1X \\
\text{Isolated step:} & H_{i+1}X & = & HH_iX + X \\
 & h_{i+1,j+1} & \equiv & H_{i+1}X = HH_iX + X \xrightarrow{Hh_{i,j}+X} HH_j + X = H_{j+1}X \\
\text{Limit step:} & H_jX & = & \operatorname*{colim}_{i<j} H_iX \quad \text{with injections } h_{i,j} : H_iX \longrightarrow H_jX,\ i < j,
\end{array}$$

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

$$\begin{tikzcd}
H_iX \arrow[r, "h_{i,j}"] \arrow[d, "h_{i,i+1}"'] & H_jX \arrow[d, dotted, "h_{j,j+1}"] \\
H_{i+1}X = HH_iX + X \arrow[r, "Hh_{i,j}+X"'] & HH_jX + X = H_{j+1}X
\end{tikzcd} \qquad i < j\,.$$

The above chain is said to *converge* if $h_{i,i+1}$ is an isomorphism for some ordinal number i.

Theorem 2.24. ([AT], Theorem IV.4.2)
Let H be an endofunctor of Set. *Then H has a free algebra on X if and only if the above chain (H_iX) converges. Furthermore, if $h_{i,i+1}$ is invertible for some i, then $(H_iX, h_{i,i+1}^{-1})$ is a free H-algebra on X.*

This result does not indicate after how many steps the free algebra (if it exists) is constructed. However, the following theorem gives a bound for endofunctors H that preserve colimits of λ-chains, i. .e, diagrams $\lambda \longrightarrow \mathsf{Set}$ for an infinite ordinal λ.

Theorem 2.25. (Adámek [A$_1$])
Let H be an endofunctor that preserves colimits of λ-chains. Then the free algebra chain converges after λ steps, and the pair $(H_\lambda X, h_{\lambda,\lambda+1}^{-1})$ is a free H-algebra on X.

This last result is not specific for Set. In fact, its proof works in any category $\mathcal{A}$ in which finite coproducts and colimits of α-chains, $\alpha \leq \lambda$, exist. In particular, if $\mathcal{A}$ is an lfp category and H is a finitary functor, then H preserves colimits of ω-chains, thus the free algebra construction stops after ω steps. Finally, notice that in the light of Remark 2.22(i) the above construction easily specializes to a construction of an initial algebra. In fact, just take for X the empty set (or, the initial object of $\mathcal{A}$).

2.3 Coalgebras

In this subsection we recall coalgebras for an endofunctor. Whereas algebras can be thought of as sets of data elements equipped with operations that "construct" new data elements (see the examples of natural numbers, lists and trees as initial algebras) coalgebras have a strong flavour of dynamic systems where a set of states is equipped with "observations" that allow to access certain information about these states albeit not the states themselves. The type of the system, i. e., the type of the possible observations of the systems, is described by the endofunctor under consideration. Most prominently, sequential automata and labelled transition systems can be considered as coalgebras, see Examples 2.29 below. Another perspective on coalgebras is the one viewing them as infinitary data structures like streams or infinite trees. Coalgebraic principles are then useful to define operations on such data structures as well as proving properties about such operations. Again, it is not our aim to provide a general introduction to the theory of coalgebras but merely collect those parts which will be essential to our work. For a general introduction to the field of coalgebra the reader may consult any of the texts [JR, R$_2$, Gu, A$_2$].

Formally, coalgebras are dual as objects to algebras in the sense that the structure is turned around.

Definition 2.26. Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be an endofunctor of a category $\mathcal{A}$. An H-coalgebra is a pair (A, a) consisting of a *carrier* A, an object of $\mathcal{A}$, and the *structure* $a : A \longrightarrow HA$. A coalgebra homomorphism h

from (A, a) to (B, b) is a morphism $h : A \longrightarrow B$ such that the square

$$\begin{tikzcd} A \arrow[r, "a"] \arrow[d, "h"'] & HA \arrow[d, "Hh"] \\ B \arrow[r, "b"'] & HB \end{tikzcd} \tag{2.18}$$

commutes.

Remark 2.27.

(i) Coalgebras for a functor H and their homomorphisms form a category

$$\mathsf{Coalg}\, H\,.$$

Coalgebras are dual to algebras in the sense that the dual of $\mathsf{Coalg}\, H$ is the category of algebras for the functor $H^{\mathrm{op}} : \mathcal{A}^{\mathrm{op}} \longrightarrow \mathcal{A}^{\mathrm{op}}$:

$$(\mathsf{Coalg}\, H)^{\mathrm{op}} = \mathsf{Alg}\, H^{\mathrm{op}}\,.$$

(ii) Dually to the situation in algebra, in coalgebra the concept of a *final* (or, *terminal*) coalgebra plays a major rôle. An H-coalgebra (T, t) is final if for any H-coalgebra (A, a) there exists a unique coalgebra homomorphism

$$[\![a]\!] : A \longrightarrow T\,.$$

This notation is to indicate that a final coalgebra is used to provide semantics of computations in dynamical systems. In fact, we shall see in the Examples 2.29 how $[\![a]\!]$ assigns to an element of A thought of as a state of a system its semantics as an element of the terminal coalgebra. This element can often be thought of as the *behaviour* of the state, i. e., the information which can be observed of the state using (repeatedly) all the observations provided by the coalgebra structure a.

Notice that a final H-coalgebra (T, t) is a terminal object of $\mathsf{Coalg}\, H$, and as such it is uniquely determined up to isomorphism.

Dualizing Lambek's Lemma 2.20 we obtain the following

Lemma 2.28. *If (T, t) is a final coalgebra, then t is an isomorphism.*

Consequently, a final H-coalgebra is the greatest fixed point of H, or the greatest solution of the domain equation

$$X \simeq HX\,.$$

Examples 2.29.

(i) Consider the functor given by $HX = X + 1$ on sets. Coalgebras $\alpha : A \longrightarrow A + 1$ are just partial unary algebras and coalgebra homomorphisms are the strict algebra homomorphisms $h : (A, \alpha) \longrightarrow (B, \beta)$, i. e.,

$$\beta \text{ is defined on } h(a) \text{ if and only if } \alpha \text{ is defined on } a.$$

Of course, we can also view (A, α) as a deterministic automaton (with unary input) with halting states, i. e., states which do not have a successor state. Usually, one writes

$$a \longrightarrow a' \qquad \text{and} \qquad a{\downarrow}$$

if a can make a state transition to a', or if a is a halting state, respectively. The coalgebra homomorphisms are functions $h : A \longrightarrow B$ preserving halting states and transitions:

(a) $a{\downarrow}$ implies $h(a){\downarrow}$, and

(b) $a \longrightarrow a'$ implies $h(a) \longrightarrow h(a')$

These functions are called simulations in automata theory. Notice that due to the determinism preservation of halting states and transitions implies their reflection by h.

A final H-coalgebra is carried by

$$\overline{\mathbb{N}} = \mathbb{N} + \{\infty\}$$

with the structure map $\mathsf{pred} : \overline{\mathbb{N}} \longrightarrow \overline{\mathbb{N}} + 1$

$$\mathsf{pred}(n) = \begin{cases} n-1 & \text{if } n \in \mathbb{N} \setminus \{0\} \\ * & \text{if } n = 0 \\ \infty & \text{if } n = \infty \end{cases}$$

where $*$ denotes the unique element of 1. For a coalgebra (A, α) the uniquely determined map $[\![\alpha]\!] : A \longrightarrow \overline{\mathbb{N}}$ assigns to every state a of A its behaviour. More precisely, $[\![\alpha]\!]$ assigns $*$ to any halting state, and to any other state a the number of steps the automaton A can make from a before a halting state is reached—this number may be infinity.

The universal property can also be understood in terms of a definition and a proof principle called corecursion and coinduction, respectively. We do not give any details of this here. The interested reader should consult the aforementioned introductory texts [JR, R$_2$, Gu, A$_2$].

(ii) Streams. Coalgebras for the functor given by $HX = S \times X$ for a set S are deterministic automata with output in S. Indeed, a coalgebra structure $\alpha = \langle o, s\rangle : A \longrightarrow S \times A$ gives for each state $a \in A$ a pair consisting of the output of state a and its successor state. Coalgebra homomorphisms are, again, simulations, i. e., maps preserving (and therefore reflecting) transitions and output. In fact, h is a coalgebra homomorphism from $(A, \langle o, s\rangle)$ to $(A', \langle o', s'\rangle)$ if and only if $o'(h(a)) = o(a)$ and $s'(h(a)) = h(s(a))$. A final coalgebra for H is carried by the set of streams (i. e., infinite sequences) S^ω with the head and tail function as structure

$$\langle \mathsf{hd}, \mathsf{tl}\rangle : S^\omega \longrightarrow S \times S^\omega .$$

For a coalgebra (A, α) the mapping $[\![\alpha]\!] : A \longrightarrow S^\omega$ assigns to a state of A the stream of outputs produced by the computation that starts at this state.

(iii) Deterministic Automata. The application of the theory of coalgebras to automata theory has been studied by Jan Rutten [R$_1$]. Here we will just indicate how automata can be understood as coalgebras. Deterministic automata with an input alphabet Σ are usually presented by a set of states Q, a next state function $\delta : Q \times \Sigma \longrightarrow Q$ and a subset $F \subseteq Q$ of final states, equivalently, a function $f : Q \longrightarrow 2$ where $2 = \{0, 1\}$ with $f(q) = 1$ if and only if $q \in F$. We disregard initial states since they are inessential when automata are treated as coalgebras. Currying δ, we get a function $\overline{\delta} : Q \longrightarrow Q^\Sigma$ defined by $\overline{\delta}(q) = \delta(q, -) : \Sigma \longrightarrow Q$. Now an automaton is presented by one function

$$\alpha = \langle \delta, f\rangle : Q \longrightarrow Q^\Sigma \times 2 ,$$

thus, it is a coalgebra for the functor given by $HX = X^\Sigma \times 2$. The coalgebra homomorphisms are the functional bisimulations: h is a coalgebra homomorphism from $(Q, \langle \delta, f\rangle)$ to $(Q', \langle \delta', f'\rangle)$ if

(a) q is final in Q if and only if $h(q)$ is final in Q', i. e., $f'(h(q)) = f(q)$, and

(b) h preserves (and therefore reflects) state transitions, i. e.,

$$h(\delta(q, s)) = \delta'(h(q), s)$$

for all $q \in Q$ and $s \in \Sigma$.

A final coalgebra is carried by the set

$$T = \mathcal{P}(\Sigma^*)$$

of all formal languages over Σ with the following structure $\tau : \langle \delta, f\rangle : T \longrightarrow T^\Sigma \times 2$:

$$\begin{aligned} f(L) &= 1 \qquad \text{if the empty word lies in } L \\ \delta(L) &= (s \longmapsto L_s = \{ w \in \Sigma^* \mid sw \in L \}) \end{aligned}$$

Now for an automaton considered as a coalgebra (A, α) the map $[\![\alpha]\!] : A \longrightarrow T$ assigns to each state a the language the automaton accepts when we choose a as the initial state of the automaton. This is the reason why we said that initial states are inessential here. The map $[\![\alpha]\!]$ summarizes the semantics of the automaton regardless of what state is the initial one.

(iv) Finitely branching nondeterministic automata (with unary input) are simply coalgebras for the finite power set functor $\mathcal{P}_{\text{fin}}$. Equivalently, these are finitely branching directed graphs. However, it may be misleading to think about $\mathcal{P}_{\text{fin}}$-coalgebras that way because coalgebra homomorphisms are stronger than the usual graph homomorphisms. In fact, h is a coalgebra homomorphism from a graph G to a graph H considered as $\mathcal{P}_{\text{fin}}$-coalgebras if and only if h is a graph homomorphism that reflects edges: if $h(a) \longrightarrow b$ is an edge in H then there exists an edge $a \longrightarrow a'$ in G with $h(a') = b$. Switching back to the system theoretic view of $\mathcal{P}_{\text{fin}}$-coalgebras the homomorphisms are precisely the functional bisimulations, i. e., the maps that preserve and reflect transitions.

A terminal $\mathcal{P}_{\text{fin}}$-coalgebra is the set of all (finite and infinite) strongly extensional finitely branching trees where the coalgebra structure assigns to a tree the finite set of its maximum proper subtrees, see [W$_2$].

(v) Labelled transition systems (lts) are an important tool in the semantics of process algebra. An lts is a triple $(L, \longrightarrow, \mathsf{Act})$ where Act is a set of (atomic) actions, L is a set of states and $\longrightarrow$ is a family of relations $\xrightarrow{a}$ on L indexed by elements a of Act. As usual, one writes

$$s \xrightarrow{a} s'$$

if s and s' are related by $\xrightarrow{a}$, and we think of this as a state transition from s to s' "consuming" a. Of course, every transition system can be summarized by a map

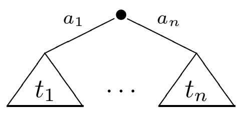
$$\alpha : L \longrightarrow \mathcal{P}(\mathsf{Act} \times L)\,,$$

where $(a, s') \in \alpha(s)$ if and only if $s \xrightarrow{a} s'$, i. e., lts are precisely the coalgebras for the endofunctor $\mathcal{P}(\mathsf{Act} \times -)$ of Set. Coalgebra homomorphism are functional (strong) bisimulations, i. e., those maps that preserve and reflect transitions: a map $h : L \longrightarrow L'$ between state sets of two lts is a coalgebra homomorphism if and only if

(a) $s \xrightarrow{a} t$ implies $h(s) \xrightarrow{a} h(t)$, and

(b) if $h(s) \xrightarrow{a} t'$, then there exists a state t in L with $s \xrightarrow{a} t$ and $h(t) = t'$.

It is clear that there can be no final coalgebra as the functor $\mathcal{P}(\mathsf{Act} \times -)$ does not have fixed points. If however, we were to put a bound on the branching breadth of lts this situation can be remedied. For example, finitely branching transition systems are coalgebras for $\mathcal{P}_{\text{fin}}(\mathsf{Act} \times -)$, and this functor has a final coalgebra in Set. It is carried by the set T of all finitely branching strongly extensional trees with edges labelled in Act. The coalgebra structure $T \longrightarrow \mathcal{P}_{\text{fin}}(\mathsf{Act} \times T)$ assigns to any tree



the finite set of pairs (a_i, t_i), $i = 1, \ldots, n$. The map $[\![\alpha]\!] : L \longrightarrow T$ assigns to every state its behaviour in the following way: first take the tree unfolding t of the state according to the transitions of L. Since this may not be strongly extensional one identifies bisimilar children of any node in t so that a strongly extensional tree is obtained. Notice that here the notion of bisimulation takes into account the labels of edges. We leave the technical details to the reader.

(vi) Infinite Σ-trees. Let Σ be a signature of operation symbols with prescribed arity and form the polynomial functor $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$. The H_Σ-coalgebras are automata with inputs from the set of natural numbers and with output from Σ, where for each state the maximum possible input number is the arity of the output of that state. More precisely, if $\alpha : A \longrightarrow H_\Sigma A$ is a coalgebra we can think of A as the set of states and α assigns to every state a its reaction $(\sigma, a_1, \ldots, a_n)$ where $\sigma \in \Sigma_n$, for some natural number n is the output of a and $a_1, \ldots, a_n$ are the successor states corresponding to the transitions that will occur if at state a we input a number $k = 1, \ldots, n$ into the automaton. The coalgebra homomorphisms are the usual automata simulations, and we leave it to the reader to work this out. Recall that the initial Σ-algebra is formed by all finite Σ-trees, see Example 2.17(iv). If we take instead all (finite and infinite) Σ-trees we get the final coalgebra T_Σ whose structure map is the inverse of tree-tupling which is sometimes called "parsing". It assigns to a Σ-tree t whose root is labelled by an n-ary operation symbol σ the $(n+1)$-tuple $(\sigma, t_1, \ldots, t_n)$ where t_i, $i = 1, \ldots n$, are the maximum proper subtrees of t.

(vii) Final coalgebras for finitary set functors. Suppose that H is a finitary endofunctor of Set described as a quotient $\varepsilon : H_\Sigma \longrightarrow H$ as in Example 2.17(vii).

In joint work with Jiří Adámek [AM$_1$] we have proved that the final H-coalgebra T is given by the quotient $T_\Sigma/{\sim^*}$ where $\sim^*$ 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 $s \sim^* t$ for two Σ-trees s and t if and only if for all $n < \omega$, $\partial_n s$ can be obtained from $\partial_n t$ by finitely many applications of the basic equations describing ε, more shortly, $s \sim^* t$ if and only if $\partial_n s \sim \partial_n t$ for all natural numbers n, where $\sim$ is the congruence of Example 2.17(vii). For example, for the endofunctor given by $HX = \{ \{ x, y, \} \mid x, y \in X \} + 1$ of Example 2.17(vii) the terminal coalgebra is the coalgebra of all unordered binary trees.

Now we will list some results from the general theory of coalgebras that we will later frequently use. Dually to Theorem 2.19, we have

Theorem 2.30. *The forgetful functor $U : \mathsf{Coalg}\, H \longrightarrow \mathcal{A}$ creates all colimits and all limits that H preserves.*

Remark 2.31. Of course, dualizing the concept of a free algebra yields the important concept of a cofree coalgebra. While this plays a major rôle in the general theory of coalgebras we will not mention anything about it here since it plays no rôle for what we are about to present.

Just as initial (or, more generally, free) algebras can be constructed inductively as a colimit of a chain, a final H-coalgebra can be constructed dually as a limit of an (op-)chain. Again, let us first consider an endofunctor H of Set. Dual to the initial algebra chain (i. e., the free algebra chain for the case $X = 0$, see 2.22(i)) we define by transfinite induction the following chain indexed by all ordinal numbers:

Initial step: $T_0 = 1$, $\quad t_{1,0} \equiv H1 \xrightarrow{!} 1$,

Isolated step: $T_{i+1} = HT_i$, $\quad t_{i+1,j+1} \equiv HT_i \xrightarrow{Ht_{i,j}} HT_j$

Limit step: $T_j = \lim\limits_{i<j} T_i$ $\quad$ with limit cone $t_{j,i} : T_j \longrightarrow T_i$, $\quad i < j$,

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

$$\begin{tikzcd} T_{i+1} = HT_i \arrow[d, "t_{i+1,i}"'] & HT_j = T_{j+1} \arrow[l, "Ht_{j,i}"'] \arrow[d, dotted, "t_{j+1,j}"] \\ T_i & T_j \arrow[l, "t_{j,i}"] \end{tikzcd} \qquad i < j\,.$$

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

Theorem 2.32. ([AK$_3$])
Let H be an endofunctor of Set. *Then the above chain T_i converges if and only if H has a final coalgebra. Moreover, if i is an ordinal number such that $t_{i+1,i}$ is an isomorphism, then $(T_i, t^{-1}_{i+1,i})$ is a final coalgebra.*

This result does not indicate after how many steps a final coalgebra (if it exists) can be obtained. However, dualizing Theorem 2.25 we see that for a λ-continuous functor, i. e., a functor that preserves limits of λ^{op}-chains, a final coalgebra is obtained in λ steps.

Corollary 2.33. *For a λ-continuous functor H, the above final coalgebra chain T_i converges after λ steps and $(T_\lambda, t^{-1}_{\lambda+1,\lambda})$ is a final H-coalgebra.*

Again, this last result holds in every category $\mathcal{A}$ having limits of α^{op}-chains, $\alpha \leq \lambda$, and for every λ-continuous functor on $\mathcal{A}$.

Unfortunately, this result is not as satisfactory as Theorem 2.25 before since many everyday functors are continuous only for large enough λ, for example, the finite power set functor is not ω-continuous but only ω_1-continuous where ω_1 is the first uncountable cardinal. However, for finitary set functors James Worrell [W$_1$, W$_2$] has proved that the final coalgebra construction converges after $\omega \cdot 2$ steps.

Theorem 2.34. *If H is a finitary endofunctor of* Set, *then the above final coalgebra chain converges after $\omega \cdot 2$ steps, i. e., $(T_{\omega 2}, t^{-1}_{\omega 2+1,\omega 2})$ is a final H-coalgebra.*

In fact, this result was proved for all accessible (equivalently, bounded) set functors. More precisely, if H is a λ-accessible set functor, see Remark 2.13, then the final coalgebra chain converges after $\lambda \cdot 2$ steps.

Example 2.35. To illustrate the result of Theorem 2.34 let us consider again the finite power set functor $\mathcal{P}_{\text{fin}}$ of Set. The sets T_i, $i < \omega$, consist of all finite strongly extensional trees of depth i. One can show that the limit T_ω can be described as the set of all countable unordered trees modulo the following equivalence $\approx$: we have $s \approx t$ if for all natural numbers n the strongly extensional quotients of the trees obtained by cutting s and t, respectively, at level n agree. Then for every $i < \omega$, $T_{\omega+i} \hookrightarrow T_\omega$ is the subset given by equivalence classes of $\approx$ given by all those unordered trees which are finitely branching up to level i, i. e., every node that can be reached from the root by a path with less than i steps has only finitely many children. Thus, in the second limit step we get the intersection of all $T_{\omega+i}$, $i < \omega$, whence $T_{\omega+\omega}$ is the set of all equivalence classes of $\approx$ given by finitely branching trees. One readily proves that the finitely branching strongly extensional trees form a set of representatives of the equivalence classes of $T_{\omega+\omega}$ as desired.

2.4 Monads

In this subsection we recall basic facts about monads, a categorical notion which is central to our work. Formally, monads are endofunctors $M : \mathcal{A} \longrightarrow \mathcal{A}$ with extra structure. One may think of this structure as an abstract way of expressing that MX consists of syntactic objects of some kind, e. g., terms over a signature with variables from a set X. In fact, the axioms of monads capture the essence of substitution of other syntactic objects for variables. Monads are also a tool that allows to capture algebras with operations that satisfy equations, e. g., monoids, groups, etc. The action of the monad on objects can then be thought of as assigning a free algebra to an object of generators. Actually, monads on Set are equivalent to algebraic theories in the sense of Lawvere [La] and Linton [Li$_1$].

Definition 2.36. A *monad* is a triple

$$\mathbb{M} = (M, \eta, \mu)$$

consisting of a functor $M : \mathcal{A} \longrightarrow \mathcal{A}$ and two natural transformations $\eta : Id \longrightarrow M$, called the *unit*, and $\mu : MM \longrightarrow M$, called the *multiplication*, such that the following diagrams

$$\begin{tikzcd}
M \arrow[r, "\eta M"] \arrow[dr, equal] & MM \arrow[d, "\mu"] & M \arrow[l, "M\eta"'] \arrow[dl, equal] \\
& M &
\end{tikzcd}
\qquad
\begin{tikzcd}
MMM \arrow[r, "\mu M"] \arrow[d, "M\mu"'] & MM \arrow[d, "\mu"] \\
MM \arrow[r, "\mu"'] & M
\end{tikzcd}$$

commute. The corresponding equations are often called left and right unit laws, and associativity.

A *finitary monad* is a monad $\mathbb{M}$ whose underlying functor is finitary.

Remark 2.37. An equivalent presentation of a monad is as a *Kleisli-triple*

$$(M, \eta, \widehat{(-)}),$$

where M is an object assignment of $\mathcal{A}$, η a family of morphisms $\eta_X : X \longrightarrow MX$ indexed by objects of $\mathcal{A}$ and $\widehat{(-)}$ is a function assigning to any $f : X \longrightarrow MY$ a morphism $\widehat{f} : MX \longrightarrow MY$ subject to three axioms: for any $f : X \longrightarrow MY$ and $g : Y \longrightarrow MZ$ we have

(i)	$\widehat{f} \cdot \eta_X = f$	(extension)
(ii)	$\widehat{\eta_X} = id$	(unit)
(iii)	$\widehat{\widehat{f} \cdot g} = \widehat{f} \cdot \widehat{g}$	(composition)

It is an easy exercise to check that Kleisli-triples and monads are equivalent presentations, see [Ma], Sec. 3, Ex. 12. In fact, any Kleisli-triple gives a monad: let $\mu_X = \widehat{id_{MX}}$ and check naturality of η and μ and the three monad laws. Conversely, every monad yields a Kleisli-triple: let $\widehat{f} = \mu_Y \cdot Mf$ and check the above three laws.

Examples 2.38.

(i) The identity functor on every category is a monad with $\eta = \mu = id$.

(ii) On Set the list monad $((\_)^*, \eta, \mu)$ is given as follows: to a set X assign the set of lists X^* of elements of X, the unit is given by $\eta_X : x \longmapsto \langle x \rangle$, and multiplication is flattening: to a list $\langle l_1, \dots, l_n \rangle$ of lists over X, μ_X assigns the list obtained by concatenating the constituent lists l_i, $i = 1, \dots, n$.

(iii) For every varietor H, the assignment $X \longmapsto FX$ of a free algebra on an object of generators gives a monad (F, η, μ) with the unit given by the universal arrows $\eta_X : X \longrightarrow FX$ and the components of the multiplications obtained as the unique homomorphic extensions $\mu_X : FFX \longrightarrow FX$ of the identity of FX.

(iv) The power set functor $\mathcal{P} : \mathsf{Set} \longrightarrow \mathsf{Set}$ carries a the structure of a monad: for a set X the unit is $\eta_X : x \longrightarrow \{ x \}$ and the multiplication $\mu_X : \mathcal{P}\mathcal{P}X \longrightarrow \mathcal{P}X$ assigns to a subset of $\mathcal{P}X$ the union of its elements.

(v) Let X be a poset considered as a category. Then a monad is precisely the same as a closure operator. In fact, $M : X \longrightarrow X$ is a monotone map with $x \leq Mx$ (due to the unit), $M \cdot M = M$ (due to the multiplication and the unit laws).

(vi) Let Σ be a finitary signature and let E be a finite set of equations. This defines a finitary variety $\mathcal{V}$ of Σ-algebras. The assignment $X \longmapsto MX$ of a free algebra in $\mathcal{V}$ to the set X of generators is a monad. The unit is given by the universal arrows $\eta_X : X \longrightarrow MX$. The multiplication μ_X is, again, given by the unique homomorphic extension of id_{MX}.

Definition 2.39. A monad morphism between monads (M, η, μ) and (M', η', μ') is a natural transformation $m : M \longrightarrow N$ such that the following diagrams

$$\begin{tikzcd}
Id \arrow[r, "\eta"] \arrow[dr, "\eta'"'] & M \arrow[d, "m"] \\
 & M'
\end{tikzcd}
\qquad
\begin{tikzcd}
MM \arrow[r, "m*m"] \arrow[d, "\mu"'] & M'M' \arrow[d, "\mu'"] \\
M \arrow[r, "m"'] & M'
\end{tikzcd}$$

commute. Notice that we denote here by $*$ the parallel composition of natural transformations, i. e., $m * m = Mm \cdot mM = mM' \cdot M'm$.

Notation 2.40. Just as (finitary) endofunctors on a category $\mathcal{A}$ form the categories $\mathsf{Fin}[\mathcal{A}, \mathcal{A}]$ and $[\mathcal{A}, \mathcal{A}]$, respectively, (finitary) monads and monad morphisms form categories

$$\mathsf{M}(\mathcal{A}) \qquad \text{and} \qquad \mathsf{FM}(\mathcal{A}),$$

respectively.

Remark 2.41. There is, of course, a canonical notion of morphisms of Kleisli-triples. A morphism between Kleisli-triples from $(M, \eta, \widehat{(-)})$ to $(M', \eta', \overline{(-)})$ is a family of morphisms $m_X : MX \longrightarrow M'X$ indexed by objects such that for all objects X and Y and morphisms $f : X \longrightarrow MY$ the diagrams

$$\begin{tikzcd}
Id \arrow[r, "\eta"] \arrow[dr, "\eta'"'] & M \arrow[d, "m"] \\
 & M'
\end{tikzcd}
\qquad
\begin{tikzcd}
MX \arrow[r, "m_X"] \arrow[d, "\widehat{f}"'] & M'X \arrow[d, "\overline{m_Y \cdot f}"] \\
MY \arrow[r, "m_Y"'] & M'Y
\end{tikzcd}$$

commute. It is easy to prove that the category of Kleisli-triples on $\mathcal{A}$ is isomorphic to $\mathsf{M}(\mathcal{A})$.

2.4.1 Free Monads and Second-Order Substitution

We have seen in Example 2.38(iii) that for a varietor H the assignment of a free algebra to an object gives a monad (F, η, μ). Michael Barr [B1] proved that this is actually a *free monad* on H with the universal arrow given by the natural transformation

$$\kappa \equiv H \xrightarrow{H\eta} HF \xrightarrow{\varphi} F \tag{2.19}$$

where φ expresses the structures of the free algebras. Conversely, a functor which has a free monad is a varietor whenever the base category is "good", see [Ma, B2] and the following theorem.

Theorem 2.42.

(i) Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be a varietor. The free algebra monad (F, η, μ) is a free monad on H, i. e., for each monad (F', η', μ') and for each natural transformation $\lambda : H \longrightarrow F'$ there exists a unique monad morphism $\overline{\lambda} : F \longrightarrow F'$ such that $\overline{\lambda} \cdot \kappa = \lambda$.

(ii) Conversely, let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be a functor on a complete and well-powered category $\mathcal{A}$. If there exists a free monad on H, then H is a varietor, and for every object X, FX carries a free H-algebra on X.

Remark 2.43. The first part of this result means that the forgetful functor $U : \mathsf{M}(\mathcal{A}) \longrightarrow [\mathcal{A}, \mathcal{A}]$ has a universal arrow at every varietor. However, U may fail to have a left-adjoint. If we restrict the codomain to finitary functors, then we get an adjoint situation

$$\begin{tikzcd}\mathsf{FM}(\mathcal{A}) \arrow[r, shift right] \arrow[r, phantom, "\perp"] & \mathsf{Fin}[\mathcal{A}, \mathcal{A}] \arrow[l, shift right]\end{tikzcd}$$

since for each finitary functor H a free monad on H is finitary.

Example 2.44. (Second-order substitution of finite Σ-trees)
As polynomial functors H_Σ are varietors, the assignment of a free Σ-algebra $F_\Sigma X$ (of all finite Σ-trees over X) to a set X yields a monad F_Σ. By Theorem 2.42, F_Σ is a free monad on H_Σ.

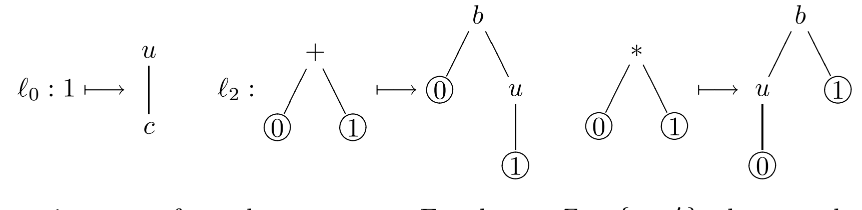
It is well-known that the freeness of the algebras $F_\Sigma X$ boils down to substitution of Σ-trees for variables. It seems to be less well-known that the universal property of the monad F_Σ specializes to *second-order substitution*, i. e., substitution of trees for operation symbols, see [C]. Before we make this precise we need to recall an important property of signatures and polynomial functors.

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 $\{ 0, \dots, n-1 \}$. 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 (2.16) in Example 2.17(v). That is, $\mathrm{Lan}_J(\Sigma)$ is the polynomial functor H_Σ. By virtue of the adjunction $\mathrm{Lan}_J(\_) \dashv (\_) \cdot J$ there is for every signature Σ and every 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.

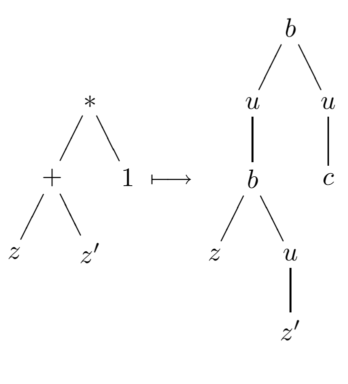
Now let Γ be another signature. Consider each $\sigma \in \Sigma_n$ as a flat tree in n distinct variables. A second-order substitution gives an "implementation" to each such σ as a finite Γ-tree in the same n variables. We model this by a natural transformation $\ell : \Sigma \longrightarrow F_\Gamma \cdot J$, i. e., a family of maps $\ell_n : \Sigma_n \longrightarrow F_\Gamma\{ 0, \dots n-1 \}$, $n \in \mathbb{N}$. As we have seen above, this gives rise to a natural transformation $\lambda : H_\Sigma \longrightarrow F_\Gamma$. Thus, from Theorem 2.42 we get a monad morphism $\overline{\lambda} : F_\Sigma \longrightarrow F_\Gamma$. For every 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 $F_\Sigma X$ by its implementation according to ℓ. More precisely, let $t = \sigma(t_1, \dots, t_n)$ with $\sigma \in \Sigma_n$ and let $t'(x_1, \dots x_n) \in F_\Gamma X$ be the implementation of σ, i. e., $\ell_n(\sigma) = t'(0, \dots, n-1)$. Then we have

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

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 $\ell : \Sigma \longrightarrow F_\Gamma \cdot J$ 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:



2.4.2 Algebras for a Monad

It is well-known that for any adjoint situation

$$\begin{tikzcd}\mathcal{B} \arrow[r, shift right, "U"'] \arrow[r, phantom, "\perp"] & \mathcal{A}, \arrow[l, shift right, "F"']\end{tikzcd} \tag{2.20}$$

with unit $\eta : Id \longrightarrow UF$ and counit $\varepsilon : FU \longrightarrow Id$ we obtain a monad

$$(UF, \eta, U\varepsilon F)$$

on $\mathcal{A}$. Conversely, given a monad $\mathbb{M} = (M, \eta, \mu)$ on $\mathcal{A}$ there are two ways to get an adjoint situation which induces the monad $\mathbb{M}$. The first one is via the *Kleisli category* $\mathcal{A}_\mathbb{M}$. Objects of $\mathcal{A}_\mathbb{M}$ are those of $\mathcal{A}$ and for morphisms we have $\mathcal{A}_\mathbb{M}(X, Y) = \mathcal{A}(X, MY)$ with identities $\eta_X : X \longrightarrow MX$ and composition defined as follows: for $f : X \multimap Y$ and $g : Y \multimap Z$ in $\mathcal{A}_\mathbb{M}$, i. e., we have $f : X \longrightarrow MY$ and $g : Y \longrightarrow MZ$ in $\mathcal{A}$, we let

$$g \circ f \equiv X \xrightarrow{f} MY \xrightarrow{Mg} MMZ \xrightarrow{\mu_Z} MZ\,.$$

There is a functor $U_\mathbb{M} : \mathcal{A}_\mathbb{M} \longrightarrow \mathcal{A}$ given by

$$U_\mathbb{M}(f : X \multimap Y) = MX \xrightarrow{Mf} MMY \xrightarrow{\mu_Y} MY$$

with a left adjoint $F_\mathbb{M} : \mathcal{A} \longrightarrow \mathcal{A}_\mathbb{M}$ which is identity on objects and is defined on morphisms by

$$F_\mathbb{M}(f : X \longrightarrow Y) = X \xrightarrow{f} Y \xrightarrow{\eta_Y} MY\,.$$

It is an easy task to check that there is indeed an adjoint situation

$$\begin{tikzcd}\mathcal{A}_\mathbb{M} \arrow[r, shift right, "U_\mathbb{M}"'] \arrow[r, phantom, "\perp"] & \mathcal{A} \arrow[l, shift right, "F_\mathbb{M}"']\end{tikzcd} \tag{2.21}$$

and that the monad induced by this adjunction is $\mathbb{M}$.

Moreover, the adjunction (2.21) is the smallest one giving $\mathbb{M}$ in the following sense: for any adjunction (2.20) defining $\mathbb{M}$ there exists a unique functor $K : \mathcal{A}_\mathbb{M} \longrightarrow \mathcal{B}$ with $U \cdot K = U_\mathbb{M}$ and $F = K \cdot F_\mathbb{M}$:

$$\begin{tikzcd} \mathcal{A}_\mathbb{M} \arrow[rr, "K"] \arrow[dr, bend right, "U_\mathbb{M}"'] & & \mathcal{B} \arrow[dl, bend left, "U"] \\ & \mathcal{A} \arrow[ul, "F_\mathbb{M}"] \arrow[ur, "F"'] & \end{tikzcd}$$

In fact, K is defined by

$$KX = FX \qquad \text{and} \qquad K(f : X \multimap Y) = \widehat{f} : FX \longrightarrow FY\,,$$

where $\widehat{f}$ is the uniquely determined morphism with $U\widehat{f} \cdot \eta_X = f : X \longrightarrow UFY$.

The second way to obtain an adjoint situation describing a given monad is via its Eilenberg-Moore category.

Definition 2.45. Let $\mathbb{M} = (M, \eta, \mu)$ be a monad on $\mathcal{A}$. An *Eilenberg-Moore algebra* for $\mathbb{M}$ is a pair (A, a) consisting of a carrier A and a structure $a : MA \longrightarrow A$ subject to the axioms

$$\begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[dr, equal] & MA \arrow[d, "a"] \\ & A \end{tikzcd} \qquad \begin{tikzcd} MMA \arrow[r, "\mu_A"] \arrow[d, "Ma"'] & MA \arrow[d, "a"] \\ MA \arrow[r, "a"'] & A \end{tikzcd}$$

which are often called *unit law* and *associativity*, respectively.

A homomorphism of Eilenberg-Moore algebras is a homomorphism of the underlying algebras for the functor M.

Remark 2.46. Eilenberg-Moore algebras and their homomorphisms form a category $\mathcal{A}^{\mathbb{M}}$ and we have an obvious forgetful functor

$$U^{\mathbb{M}} : \mathcal{A}^{\mathbb{M}} \longrightarrow \mathcal{A}\,,$$

which always has a left-adjoint $F^{\mathbb{M}} : A \longrightarrow \mathcal{A}^{\mathbb{M}}$ defined by

$$FX = (MX, \mu_X) \qquad \text{and} \qquad F(f : X \longrightarrow Y) = Mf\,.$$

Again, the adjunction

$$\begin{tikzcd} \mathcal{A}^{\mathbb{M}} \arrow[r, shift right, "U^{\mathbb{M}}"'] \arrow[r, phantom, "\perp"] & \mathcal{A} \arrow[l, shift right, "F^{\mathbb{M}}"'] \end{tikzcd} \tag{2.22}$$

induces the monad $\mathbb{M}$.

The categories $\mathcal{A}^{\mathbb{M}}$ of algebras for a monad have "algebraic flavour" as witnessed for example by the following result.

Theorem 2.47. *The forgetful functor $U^{\mathbb{M}}$ creates all limits and all colimits that are preserved by M.*

In particular, if $\mathcal{A}$ is an lfp category and $\mathbb{M}$ is finitary then $U^{\mathbb{M}}$ creates filtered colimits. Moreover, we have the following result, see [AR], 2.78.

Theorem 2.48. *Let $\mathbb{M}$ be a finitary monad on an lfp category. Then $\mathcal{A}^{\mathbb{M}}$ is lfp.*

Now suppose we have an adjoint situation (2.20) with its induced monad $\mathbb{M}$. Then the adjunction (2.22) is the largest one inducing $\mathbb{M}$ in the following sense: for any adjunction (2.20) inducing $\mathbb{M}$ there exists a unique functor $L : \mathcal{B} \longrightarrow \mathcal{A}^{\mathbb{M}}$ satisfying $U^{\mathbb{M}} \cdot L = U$ and $F^{\mathbb{M}} = L \cdot F$:

$$\begin{tikzcd} \mathcal{B} \arrow[rr, "L"] \arrow[dr, bend right, "U"'] & & \mathcal{A}^{\mathbb{M}} \arrow[dl, bend left, "U^{\mathbb{M}}"] \\ & \mathcal{A} \arrow[ul, "F"] \arrow[ur, "F^{\mathbb{M}}"'] & \end{tikzcd}$$

In fact, L is defined by

$$LB = (UB, U\varepsilon_B : MUB = UFUB \longrightarrow UB) \qquad \text{and} \qquad Lf = Uf\,.$$

One may think of L as measuring the degree of "algebraicness" of $\mathcal{B}$ over $\mathcal{A}$. In fact, if L is an isomorphism one can regard the category $\mathcal{B}$ as a category of algebras.

Definition 2.49. A functor $U : \mathcal{B} \longrightarrow \mathcal{A}$ is called *monadic* provided that

(i) U has a left adjoint $F : \mathcal{A} \longrightarrow \mathcal{B}$

(ii) the comparison functor $L : \mathcal{B} \longrightarrow \mathcal{A}^{\mathbb{M}}$, where $\mathbb{M}$ is the monad induced by the adjunction $F \dashv U$, is an isomorphism of categories.

Monadic functors $U : \mathcal{B} \longrightarrow \mathcal{A}$ can be characterized with respect to the creation of certain coequalizers by U. Recall that a *split coequalizer* of a parallel pair $f, g : A \longrightarrow B$ is a morphism $c : B \longrightarrow C$ so that we have two more morphisms s and t as in the diagram

$$\begin{tikzcd} A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \arrow[r, "c"] \arrow[l, bend left=50, "t"] & C \arrow[l, bend left=50, "s"] \end{tikzcd}$$

so that $c \cdot f = c \cdot g$, $c \cdot s = id$, $s \cdot c = f \cdot t$ and $g \cdot t = id$. In fact, it is easy to check that c is then a coequalizer of f and g. Recall further, that a colimit is called *absolute* provided that it is preserved by every functor. The proof of the following Theorem can be found in [ML].

Theorem 2.50. (Beck's Theorem)
Let $F \dashv U : \mathcal{B} \longrightarrow \mathcal{A}$ be an adjunction with an induced monad $\mathbb{M}$. Then the following are equivalent:

(i) U is monadic

(ii) U creates coequalizers which are U-absolute, i. e., coequalizers of those parallel pairs f, g in $\mathcal{B}$ for which Uf, Ug has an absolute coequalizer in $\mathcal{A}$.

(iii) U creates coequalizers of U-split pairs, i. e., pairs of parallel morphisms f, g in $\mathcal{B}$ for which the pair Uf, Ug has a split coequalizer in $\mathcal{A}$

We conclude this subsection with a number of examples.

Examples 2.51.

(i) Recall that a monoid is an algebra with a binary associative operation and a unit of this operation. The category of monoids and their homomorphisms is monadic over Set (i. e., the corresponding forgetful functor is monadic). In fact, this category is isomorphic to $\mathsf{Set}^{\mathbb{M}}$ where $\mathbb{M} = ((\_)^*, \eta, \mu)$ is the list monad of Example 2.38(ii). For every set X, the set X^* together with the operation of concatenation and the empty list is a free monoid on X.

(ii) More generally, every finitary variety is monadic over Set. In fact, recall that a finitary variety $\mathcal{V}$ is a category of algebras for a signature of operations and finitely many equations between these operations. It is not difficult to show that $\mathcal{V}$ is the category of Eilenberg-Moore algebras for the monad given by assigning to each set X (of generators) a free algebra in $\mathcal{V}$ on X.

(iii) For every varietor $H : \mathcal{A} \longrightarrow \mathcal{A}$, the category $\mathsf{Alg}\, H$ is monadic over $\mathcal{A}$. In fact, $\mathsf{Alg}\, H$ is isomorphic to the category $\mathcal{A}^{\mathbb{F}}$ of Eilenberg-Moore algebras for the free monad $\mathbb{F} = (F, \eta, \mu)$ on H. While this is trivial to prove with the help of Theorem 2.50, a direct proof is quite instructive: the isomorphism $L : \mathsf{Alg}\, H \longrightarrow \mathcal{A}^{\mathbb{F}}$ assigns to each algebra (A, a) the algebra $(A, \widehat{a})$ where $\widehat{a} : FA \longrightarrow A$ is the unique homomorphic extension of $id : A \longrightarrow A$. The inverse L^{-1} takes an algebra (A, α) to

$$HA \xrightarrow{\kappa_A} FA \xrightarrow{\alpha} A\,,$$

where $\kappa : H \longrightarrow F$ is the canonical transformation of (2.19). We leave the details to the reader. Notice that the Eilenberg-Moore algebra structure $\widehat{a} : FA \longrightarrow A$ induced by an algebra (A, a) is a generalized version of the computation of finite trees in a Σ-algebra A, see Example 2.23(i).

(iv) The algebras for the power set monad $\mathcal{P}$ on Set are the complete lattices; more precisely, partially ordered sets with all joins, together with the continuous maps, i. e., function preserving all joins.

(v) Finitary monads are algebras. In fact, suppose that $\mathcal{A}$ is an lfp category. Then the forgetful functor

$$U : \mathsf{FM}(\mathcal{A}) \longrightarrow \mathsf{Fin}[\mathcal{A}, \mathcal{A}]$$

is monadic with its left adjoint assigning to any finitary endofunctor H of $\mathcal{A}$ a free monad on H.

3 Iterative and Completely Iterative Algebras

Is this a dagger which I see before me?
William Shakespeare, *Macbeth*, Act 2, Scence 1.

It was the idea of Calvin Elgot [E] to study algebraically the semantics of recursive computations without using the common approach which is based on working with ordered or metrized structures. He introduced iterative theories, and he and his coauthors proved that the free iterative theory over a signature Σ exists [BE] and that it is the theory formed by rational trees, see [EBT]. Recall that a Σ-tree is called *rational* if it has up to isomorphism finitely many subtrees only, see [Gi_2].

Later it has been realized that a much easier approach to iterative theories is via *iterative algebras.* For a signature Σ they were studied by Evelyn Nelson [N]. Her work showed that the free iterative algebras are the algebras of rational trees over a set X, and that this yields a short proof of Elgot's free iterative theories. The first attempts to capture rational trees categorically were [GLM_1] and [AMV_1]. In the categorical approach one replaces signatures by a finitary endofunctor of Set (or more general base categories). In [AMV_1] we gave a categorical construction of the rational monad generated by such an endofunctor H, and we proved that it is characterized as the free iterative monad on H. The authors of [GLM_1] have generalized our construction but did not prove the universal property of the rational monad in their setting. Both papers do not work with iterative algebras. Moreover, our paper [AMV_1] is rather long and technical, and besides we had to put several side conditions on the category and endofunctor we worked with. In Subsection 3.2 below we will introduce and study iterative algebras for every finitary functor H of an lfp category $\mathcal{A}$. We will show that free iterative algebras exist and we give a coalgebraic construction of them. In this way we get a generalization of Nelson's results and later in Section 6 we show that free iterative algebras yield free iterative monads. Thus we obtain a new proof of Elgot's classical result in the special case of a polynomial functor H_Σ of Set. At the same time we have generalized the result and, moreover, our proof is conceptually much simpler then any of the previous ones.

But before that we introduce and study completely iterative algebras in Subsection 3.1, which allow to give an easy approach to completely iterative theories of Elgot et al. In [EBT] the authors proved that the theory of all Σ-trees forms a free completely iterative theory on a signature Σ. We will generalize this classical result in our work. It turns out that complete iterativity of infinite trees is due to the fact that they form a final coalgebra. The result that final coalgebras form a monad that is completely iterative was discovered independently and almost at the same time in work by Lawrence Moss [Mo_1] and by Peter Aczel, Jiří Adámek, Jiří Velebil [AAV]. That final coalgebras form a monad was also proved for (generalized) polynomial endofunctors of an lfp category by Neil Ghani, Christoph Lüth, Federico De Marchi and John Power [GLMP_1, GLMP_2], see also [DM]. In our paper [AAMV] we proved that final coalgebras yield a monad in general, and that this monad is characterized as the free completely iterative monad. More precisely, one works with a category $\mathcal{A}$ with binary coproducts and a functor H having "enough final coalgebras", i. e., for every object Y of $\mathcal{A}$ there exists a final coalgebra TY of $H(\_) + Y$. Then T forms a free completely iterative monad on H. In [M_1] we have added completely iterative algebras to the picture. We proved that for an object mapping T of $\mathcal{A}$ the following three statements are equivalent:

(a) for every object Y, TY is a final coalgebra for $H(\_) + Y$,

(b) for every object Y, TY is a free completely iterative H-algebra on Y, and

(c) T is a free completely iterative monad on H.

We shall establish the equivalence of (a) and (b) in Subsection 3.1 below. This is the first part of our paper [M_1]. Its second part deals with completely iterative monads, which will be studied in Section 6 where we add (c) to the above list of equivalent statements.

3.1 Completely Iterative Algebras

Before we introduce completely iterative algebras in general we would like to illustrate the leading example of completely iterative algebras for a given signature Σ, i. e., for a polynomial functor H_Σ of Set, a bit more.

A Σ-algebra A is called *completely iterative*, if every system

$$x_i \approx t_i, \quad i \in I, \tag{3.23}$$

where I is some (possibly infinite) set, $X = \{ x_i \mid i \in I \}$ is a set of variables and the t_i are terms over $X + A$, none of which is just a single variable, has a unique solution in A. By a *solution* we mean a set

$\{ x_i{}^\dagger \mid i \in I \}$ of elements of A such that the above formal equations (3.23) become actual identities in A when the variables are substituted by the solutions and the terms t_i are interpreted in A, i. e.,

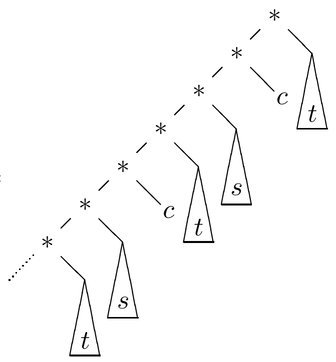
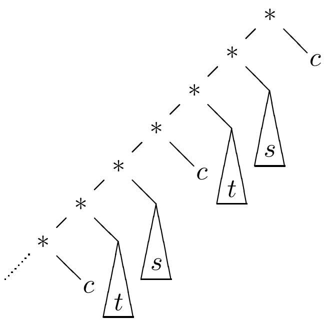
$$x_i{}^\dagger \equiv \alpha\left(t_i[x_j := x_j{}^\dagger]_{j\in J}\right), \quad i \in I,$$

where $\alpha : F_\Sigma A \longrightarrow A$ is the canonical computation of finite Σ-trees in A, see Example 2.23(i).

For example, suppose we have a signature Σ. The algebra $A = T_\Sigma$ of all finite and infinite Σ-trees is completely iterative. For example, let Σ consist of a binary operation symbol $*$ and a constant symbol c. Then the following system

$$x_1 \approx x_2 * t \qquad x_2 \approx (x_1 * s) * c \tag{3.24}$$

where s and t are some trees in T_Σ has the following solution

$x_1{}^\dagger =$  and $x_2{}^\dagger =$  (3.25)

Observe that it is sufficient to allow for the right-hand sides in (3.24) only so-called *flat terms*, i. e., terms t that are either

$$t = \sigma(x_1, \dots, x_n), \qquad \sigma \in \Sigma_n, \quad x_1, \dots, x_n \in X,$$

or

$$t \in A.$$

In fact, for every system (3.23) one can give a system with only flat terms as right-hand sides, which has the same solution. This is done by introducing (possibly infinitely many) new variables. For example for the system (3.24) we get the following flat one:

$$\begin{array}{lcllcl} x_1 & \approx & x_2 * z_1 & z_2 & \approx & x_1 * z_4 \\ x_2 & \approx & z_2 * z_3 & z_3 & \approx & c \\ z_1 & \approx & t & z_4 & \approx & s \end{array} \tag{3.26}$$

Obviously, the solutions $x_1{}^\dagger$ and $x_2{}^\dagger$ are the same trees as before.

Clearly, one can write every system with flat right-hand sides as a single map

$$e : X \longrightarrow H_\Sigma X + A\,,$$

and a solution is a map $e^\dagger : X \longrightarrow A$ such that the following square

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[d, "e"'] & A \\ H_\Sigma X + A \arrow[r, "H_\Sigma e^\dagger + A"'] & H_\Sigma A + A \arrow[u, "{[a,A]}"'] \end{tikzcd}$$

where a denotes the algebra structure of A, commutes. The following are the minimal requirements to formulate these notions.

Assumption 3.1. For the rest of this text we assume that $\mathcal{A}$ is a category with binary coproducts and that $H : \mathcal{A} \longrightarrow \mathcal{A}$ is an endofunctor of $\mathcal{A}$. We shall always denote injections of a coproduct $A + B$ by $\mathsf{inl} : A \longrightarrow A + B$ and $\mathsf{inr} : B \longrightarrow A + B$, and we denote by $\mathsf{can} : HA + HB \longrightarrow H(A + B)$ the canonical morphism $[H\mathsf{inl}, H\mathsf{inr}]$.

Definition 3.2. A morphism $e : X \longrightarrow HX + A$ of $\mathcal{A}$ is called a *flat equation morphism* in (the object of parameters) A. Suppose that A is the underlying object of an H-algebra $a : HA \longrightarrow A$. Then a *solution* of e in A is a morphism $e^\dagger : X \longrightarrow A$ such that the following diagram

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[d, "e"'] & A \\ HX + A \arrow[r, "He^\dagger + A"'] & HA + A \arrow[u, "{[a,A]}"'] \end{tikzcd} \tag{3.27}$$

commutes.

An H-algebra is called *completely iterative* (or shortly, *cia*) if every flat equation morphism in it has a unique solution.

Notation 3.3. For every flat equation morphism $e : X \longrightarrow HX + Y$ and every morphism $h : Y \longrightarrow Z$ we get a flat equation morphism $h \bullet e$ as the "renaming of parameters by h":

$$h \bullet e \equiv X \xrightarrow{e} HX + Y \xrightarrow{HX+h} HX + Z\,.$$

Proposition 3.4. ([M$_1$], Proposition 2.3)
Let (A, a) and (B, b) be completely iterative H-algebras, and let $h : A \longrightarrow B$ be a morphism. Then the following are equivalent:

(i) $h : (A, a) \longrightarrow (B, b)$ is an H-algebra homomorphism,

(ii) h is solution-preserving, i. e., for all $e : X \longrightarrow HX + A$ we have

$$(h \bullet e)^\dagger = h \cdot e^\dagger\,.$$

Notation 3.5. We denote by

$$\mathsf{CIA}\, H$$

the category of all completely iterative algebras and H-algebra homomorphisms. This choice of morphisms is explained by Proposition 3.4, and $\mathsf{CIA}\, H$ is a full subcategory of $\mathsf{Alg}\, H$, the category of all H-algebras and homomorphisms.

Examples 3.6.

(i) Classical algebras are seldom cias. For example, let $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$ be the functor expressing one binary operation, $H_\Sigma X = X \times X$. Then a group is a cia if and only if its unique element is the unit 1, since the recursive equation $x \approx x \cdot 1$ has a unique solution. A lattice is a cia if and only if it has a unique element; consider the unique solution of $x \approx x \vee x$.

(ii) In [AMV$_2$] it was proved that the algebra of addition on the augmented positive natural numbers

$$\{\,1, 2, 3, \dots\,\} \cup \{\,\infty\,\}$$

is a cia w. r. t. the functor H_Σ of (i).

(iii) Final coalgebras are completely iterative algebras. More precisely, denote by (T, α) a final coalgebra for H. Recall that by Lambek's Lemma 2.28, the structure map α is an isomorphism, whose inverse we denote by $\tau : HT \longrightarrow T$. Then this H-algebra (T, τ) is completely iterative. In fact, consider a flat equation morphism

$$e : X \longrightarrow HX + T\,,$$

and form the H-coalgebra

$$\overline{e} \equiv X + T \xrightarrow{[e,\mathsf{inr}]} HX + T \xrightarrow{HX+\alpha} HX + HT \xrightarrow{\text{can}} H(X + T)\,.$$

Then the left-hand component of the unique coalgebra homomorphism $[\![\overline{e}]\!] : X + T \longrightarrow T$ is the desired unique solution $e^\dagger$ of e.

(iv) Infinite trees form completely iterative algebras. Let Σ be a signature. Recall from Example 2.29(vi) that the algebras Σ-algebra T_Σ of all (finite and infinite) Σ-trees is a final H_Σ-coalgebra. Thus, T_Σ is a cia.

(v) The final coalgebra for $\mathcal{P}_{\mathrm{fin}} : \mathsf{Set} \longrightarrow \mathsf{Set}$ is the coalgebra T of all strongly extensional finitely branching trees, see Example 2.29(iv). It follows from (iii) that T is a cia.

(vi) Complete metric spaces have been used as a basis for semantics by several authors, see [AN, ARu, ARe, TR]. Algebras over complete metric spaces are cias. Take $\mathcal{A} = \mathsf{CMS}$, the category whose objects are complete metric spaces (i. e., such that each Cauchy sequence has a limit), where distances are measured in the interval $[0, 1]$. The morphisms of CMS are the *non-expanding maps*, i. e., functions f from a space (X, d_X) to a space (Y, d_Y) such that $d_Y(f(x), f(y)) \leq d_X(x, y)$ for all $x, y \in X$. A stronger condition is that f be an *ε-contraction*, i. e., for some $\varepsilon < 1$ we have $d_Y(f(x), f(y)) \leq \varepsilon \cdot d_X(x, y)$ for all $x, y \in X$. Recall that for given complete metric spaces (X, d_X) and (Y, d_Y) the hom-set in CMS is a complete metric space with the metric given by

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

Now suppose we have a functor $H : \mathsf{CMS} \longrightarrow \mathsf{CMS}$ which is *contracting*, i. e., there exists a constant $\varepsilon < 1$ such that all derived maps $\mathsf{CMS}(X, Y) \longrightarrow \mathsf{CMS}(HX, HY)$ are ε-contractions. Then every non-empty H-algebra (A, a) is completely iterative. In fact, given any flat equation morphism $e : X \longrightarrow HX + A$ in CMS, choose some element $a \in A$ and define a Cauchy sequence $(e_n^\dagger)_{n \in \mathbb{N}}$ in $\mathsf{CMS}(X, A)$ inductively as follows: let $e_0^\dagger = \mathsf{const}_a$, and given $e_n^\dagger$ define $e_{n+1}^\dagger$ by the commutativity of the following diagram (compare (3.27))

$$\begin{tikzcd}
X \arrow[r, "e_{n+1}^\dagger"] \arrow[d, "e"'] & A \\
HX + A \arrow[r, "He_n^\dagger + A"'] & HA + A \arrow[u, "{[a,A]}"']
\end{tikzcd} \tag{3.28}$$

In [AMV$_3$], Lemma 2.9, it is proved that this is indeed a Cauchy sequence in $\mathsf{CMS}(X, A)$ and that its limit yields a unique solution of e.

(vii) Completely metrizable algebras. Many set functors H have a lifting to contracting endofunctors H' of CMS. That is, for the forgetful functor $U : \mathsf{CMS} \longrightarrow \mathsf{Set}$ the following square

$$\begin{tikzcd}
\mathsf{CMS} \arrow[r, "H'"] \arrow[d, "U"'] & \mathsf{CMS} \arrow[d, "U"] \\
\mathsf{Set} \arrow[r, "H"'] & \mathsf{Set}
\end{tikzcd}$$

commutes. For example, for the functor $HX = X^n$ we have the lifting $H'(X, d) = (X^n, \frac{1}{2} \cdot d_{\max})$, where $d_{\max}$ is the maximum metric, which is a contracting functor with $\varepsilon = \frac{1}{2}$. Since coproducts of $\frac{1}{2}$-contracting liftings are $\frac{1}{2}$-contracting liftings of coproducts, we conclude that every polynomial endofunctor has a contracting lifting to CMS.

Let $\alpha : HA \longrightarrow A$ be an H-algebra such that there exists a complete metric, d, on A such that α is a non-expanding map from $H'(A, d)$ to (A, d). Then A is a completely iterative H-algebra. In fact, 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 ($d_0(x, x') = 1$ if and only if $x \neq x'$).

For example, let A be the interval $[\frac{3}{2}, 2]$ and let α be the unary operation $\alpha(x) = 1 + \frac{1}{x}$. Then (A, α) is an algebra for the functor $H = Id$ on Set, and this algebra is completely metrizable. In fact, α is a $\frac{1}{2}$-contraction of $[\frac{3}{2}, 2]$, thus, (A, α) is an algebra for the lifting $H'(X, d) = (X, \frac{1}{2} \cdot d)$ of H. Now consider the formal equation $x \approx \alpha(x)$ that gives a flat equation morphism $e : 1 \longrightarrow H1 + A$. Its unique solution $e^\dagger : 1 \longrightarrow A$ chooses the golden ratio φ.

(viii) 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. [Ba]. Furthermore, if $f_i : X \longrightarrow X$, $i = 1, \dots, n$, are contractions of the space X with contraction factors c_i, $i = 1, \dots, n$, then it is easy to show that the map

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

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 for the contracting endofunctor $H(X,d) = (X^n, c \cdot d_{\max})$. Thus, whenever X is non-empty, then $(C(X), \alpha_X)$ is a cia.

(ix) The Cantor "middle-third" set c is the unique non-empty compact subset of the interval $[0,1]$ which satisfies $c = \frac{1}{3}c \cup (\frac{2}{3} + \frac{1}{3}c)$. 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 $\alpha_I : C(I)^2 \longrightarrow C(I)$ with $\alpha_I(A,B) = f[A] \cup g[B]$ gives the structure of a cia on $C(I)$ for the functor $H(X,d) = (X^2, \frac{1}{3} \cdot d_{\max})$, which is a lifting of the polynomial endofunctor $H_\Sigma X = X \times X$ of Set expressing one binary operation symbol α. Now consider the formal equation

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

which gives rise to a flat equation morphism $e : 1 \longrightarrow H1 + C(I)$, where 1 denotes the trivial one point space. Its unique solution $e^\dagger : 1 \longrightarrow C(I)$ is easily seen to choose the Cantor space.

(x) Continuing with our last point, for each non-empty compact $t \in C(I)$, 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 in Section 7 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 $t \longmapsto s(t)$.

(xi) (Unary algebras over Set)
Here we have $\mathcal{A} = \mathsf{Set}$ and $H = \mathit{Id}$. A unary algebra (A, α_A) is completely iterative if and only if

(a) there exists a unique fixed point $a_0 \in A$ of all $\alpha_A^k : A \longrightarrow A$, $k \geq 1$,

(b) for every sequence $(b_i)_{i<\omega}$ in A with $b_i = \alpha_A(b_{i+1})$ we have $b_i = a_0$ for every $i < \omega$ (i.e., if a is an element of A in which an infinite *alpha*-chain of elements of A ends, then $a = a_0$).

To see that (a) and (b) are necessary, solve the equation $x \approx \alpha(x)$ to obtain the fixed point a_0. Furthermore, the system

$$x_i \approx \alpha(x_{i+1}), \qquad i < \omega,$$

has as solutions every sequence as in (b); in particular, the constant sequence at a_0 is a solution, and this must be the unique one.

For the sufficiency, suppose that (A, α_A) satisfies (a) and (b). Given any equation morphism $e : X \longrightarrow H_\Sigma X + A$ there is a unique solution $e^\dagger : X \longrightarrow A$: If $x \in X$ is such that there exist equations

$$\begin{aligned} x = x_0 &\approx \alpha(x_1) \\ x_1 &\approx \alpha(x_2) \\ &\vdots \\ x_{k-1} &\approx \alpha(x_k) \\ x_k &\approx a \end{aligned}$$

where $a \in A$, then $e^\dagger(x_k) = a$ and therefore $e^\dagger(x) = \alpha^k(a)$. Otherwise we have equations

$$x = x_0 \approx \alpha(x_1)$$

$$\begin{array}{rcl} x_1 & \approx & \alpha(x_2) \\ x_2 & \approx & \alpha(x_3) \\ & \vdots & \end{array}$$

and (a) and (b) ensure that the unique solution is given by $e^\dagger(x_i) = a_0$, for all i.

We have seen above that final coalgebras yield cias. In fact, they are precisely the initial ones.

Theorem 3.7. ([M$_1$], Theorem 2.8)
Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be any endofunctor.

(i) If (T, α) is a final H-coalgebra, then (T, τ) with $\tau = \alpha^{-1}$ is an initial completely iterative H-algebra.

(ii) Conversely, if (T, τ) is an initial completely iterative H-algebra, then τ is an isomorphism and (T, α) with $\alpha = \tau^{-1}$ is a final H-coalgebra.

Sketch of Proof. Before we prove the two statements we shall establish one useful fact about the relation between H-coalgebras and cias. Suppose that (C, c) is any H-coalgebra and (A, a) is a cia. We can form an equation morphism

$$e \equiv C \xrightarrow{c} HC \xrightarrow{\mathsf{inl}} HC + A\,.$$

Then there is a one-to-one correspondence between solutions of e and morphisms $h : C \longrightarrow A$ such that $h = a \cdot Hh \cdot c$ (the so-called coalgebra to algebra homomorphisms). Indeed, this follows easily by inspection of the following diagram:

$$\begin{tikzcd}
C \arrow[rrr, "h"] \arrow[dr, "c"] \arrow[dd, "e"'] & & & A \\
& HC \arrow[r, "Hh"] \arrow[dl, "\mathsf{inl}"] & HA \arrow[ur, "a"'] \arrow[dr, "\mathsf{inl}"'] & \\
HC + A \arrow[rrr, "Hh+A"'] & & & HA + A \arrow[uu, "{[a,A]}"']
\end{tikzcd}$$

Since there exists a unique solution $e^\dagger$ for e, there exists a unique coalgebra to algebra homomorphism h. It is now quite easy to prove the theorem.

(i) $\Rightarrow$ (ii): We have seen in Example 3.6(iii) that (T, τ) is completely iterative. It remains to prove the initiality. Given any cia (A, a) we have by the above considerations a unique coalgebra to algebra homomorphism $h : T \longrightarrow A$, i. e., unique H-algebra homomorphism $h : (T, \tau) \longrightarrow (A, a)$.

(ii) $\Rightarrow$ (i): One first shows that the structure map of an initial cia (T, τ) is an isomorphism. In fact, $(HT, H\tau)$ is a cia by Proposition 2.6 of [M$_1$]. Thus we obtain an inverse of τ as in the proof of Lambek's Lemma 2.20, and so T is a coalgebra with structure $\alpha = \tau^{-1}$. It remains to show that (T, α) is final. Given any H-coalgebra (C, c) there exists a unique coalgebra to algebra homomorphism $h : C \longrightarrow T$, i. e., a unique H-coalgebra homomorphism $h : (C, c) \longrightarrow (T, \alpha)$. □

Remark 3.8. Notice that in the proof of part "(ii) $\Rightarrow$ (i)" of Theorem 3.7 in lieu of the full universal property of (T, τ) one only uses that the structure map τ is an isomorphism. Thus, the only cia with an isomorphic structure map is the initial one.

By a *free cia* on an object Y of $\mathcal{A}$ we mean, of course, a cia (TY, τ_Y) together with a morphism $\eta_Y : Y \longrightarrow TY$ in $\mathcal{A}$ such that for every cia (A, a) and every morphism $f : Y \longrightarrow A$ in $\mathcal{A}$ there exists a unique homomorphism $\widehat{f} : (TY, \tau_Y) \longrightarrow (A, a)$ extending f, i. e., such that the following diagram

$$\begin{tikzcd}
Y \arrow[r, "\eta_Y"] \arrow[dr, "f"'] & TY \arrow[d, "\widehat{f}"] & HTY \arrow[l, "\tau_Y"'] \arrow[d, "H\widehat{f}"] \\
& A & HA \arrow[l, "a"]
\end{tikzcd}$$

commutes.

The following result shows that free cias correspond to initial cias in a similar way as is the case for ordinary H-algebras.

Theorem 3.9. ([M$_1$], Theorem 2.10)
For every object Y of $\mathcal{A}$ the following are equivalent:

(i) TY is an initial completely iterative algebra for $H(\_)+Y$.

(ii) TY is a free completely iterative H-algebra on Y.

Corollary 3.10. *Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be any endofunctor.*

(i) If (TY, α_Y) is a final coalgebra for $H(\_)+Y$ the inverse of its structure map gives an H-algebra $\tau_Y : HTY \longrightarrow TY$ and a morphism $\eta_Y : Y \longrightarrow TY$, and these form a free cia on Y.

(ii) Conversely, if (TY, τ_Y) is a free cia on Y via a universal arrow $\eta_Y : Y \longrightarrow TY$ then $[\tau_Y, \eta_Y] : HTY + Y \longrightarrow TY$ is an isomorphism and its inverse yields a final coalgebra for $H(\_)+Y$.

This corollary establishes the desired equivalence of the statements (a) and (b) from the introduction of this section.

Examples 3.11.

(i) The free cias of $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$. Recall from Example 3.6(iv) the algebra T_Σ of all (finite and infinite) Σ-trees. This algebra is a cia. For every set Y the algebra $T_\Sigma Y$ of all Σ-trees over Y, i. e., finite and infinite Σ-trees where, however, leaves can be labelled by constant symbols from Σ or variables from the set Y, is a final coalgebra for $H_\Sigma(\_)+Y$. By Corollary 3.10, this implies that $T_\Sigma Y$ is a free completely iterative Σ-algebra on Y.

(ii) The free cias of $\mathcal{P}_{\text{fin}} : \mathsf{Set} \longrightarrow \mathsf{Set}$. Recall the final coalgebra T of $\mathcal{P}_{\text{fin}}$ from Example 3.6(v). Analogously, for a set Y a final coalgebra for $\mathcal{P}_{\text{fin}}(\_)+Y$ is the algebra $T(Y)$ of all finitely branching strongly extensional trees with leaves partially labelled in the set Y. By Corollary 3.10, this implies that $T(Y)$ is a free cia on Y.

(iii) The free cias of a finitary set endofunctor. Let $H : \mathsf{Set} \longrightarrow \mathsf{Set}$ be a finitary functor. Recall from Example 2.29(vii) that H is a quotient of some polynomial functor H_Σ via $\varepsilon : H_\Sigma \longrightarrow H$ and that the final coalgebra for H is $T_\Sigma/{\sim^*}$. Analogously, for each set Y we obtain $H(\_)+Y$ as a quotient of $H_\Sigma(\_)+Y$ via $\varepsilon' = \varepsilon + id$, see [AM$_1$]. Notice that the basic equations for ε' are essentially the same as those for ε. Thus $TY = T_\Sigma Y/{\sim_Y^*}$, where $\sim_Y^*$ is defined precisely as $\sim^*$ for T_Σ in Example 2.29(vii). For example, for the functor $\mathcal{P}_2$ assigning to a set Y the set of unordered pairs of Y, TY is the coalgebra of all unordered binary trees with leaves labelled in the set Y.

Definition 3.12. We call an endofunctor H *iteratable*, if it has for every object Y of $\mathcal{A}$ a final coalgebra TY of $H(\_)+Y$.

Corollary 3.13. *For an iteratable endofunctor H the assignment $Y \longmapsto TY$ of a free cia to an object yields a monad $\mathbb{T} = (T, \eta, \mu)$.*

In fact, the unit is given by the universal arrows $\eta_Y : Y \longrightarrow TY$, and the monad multiplication $\mu_Y : TTY \longrightarrow TY$ is obtained as the unique homomorphic extension of id_{TY}. In other words, $\mathbb{T}$ is the monad arising from the adjunction

$$\begin{tikzcd}\mathsf{CIA}\ H \arrow[r, bend right, "U"'] \arrow[r, phantom, "\perp"] & \mathcal{A} \arrow[l, bend right]\end{tikzcd}$$

where the left-adjoint assigns to every object of $\mathcal{A}$ a free cia on that object, see Subsection 2.4.2.

Observe that an immediate corollary of 3.10 is the Substitution Theorem of [AAMV].

Theorem 3.14. ([AAMV], Theorem 2.17)
Let H be an iteratable endofunctor. Then for every substitution $s : X \longrightarrow TY$ there exists a unique H-algebra homomorphism $\widehat{s} : TX \longrightarrow TY$ with $\widehat{s} \cdot \eta_X = s$.

Notice that for a polynomial endofunctor H_Σ of Set, this theorem states that substitution works for all (finite and infinite) Σ-trees in precisely the same way as for finite Σ-trees (or terms), see Example 2.23(i).

Remark 3.15. For every cia (A, a) we obtain a unique homomorphism

$$\widetilde{a} : TA \longrightarrow A$$

extending id_A. It is easy to see that this yields the structure of an Eilenberg-Moore algebra for the monad $\mathbb{T}$. For a polynomial endofunctor H_Σ of Set one can think of $\widetilde{a}$ as "computations of all Σ-trees", analogously as for finite Σ-trees in Example 2.23(i).

3.2 Iterative Algebras

In the previous subsection we have seen completely iterative algebras in which every system of flat equations has a unique solution. In applications it is often desirable to be able to obtain solutions only for systems which are finitary, i. e., with only finitely many variables. In fact, Calvin Elgot introduced and studied first iterative theories, which are algebraic theories with unique solutions of certain finitary recursive equations. And Evelyn Nelson introduced in [N] iterative algebras for a signature Σ to obtain an easier approach to iterative theories. Recall from [N] that a Σ-algebra A is called *iterative* if every system (3.23) where the set X of variables is finite has a unique solution in A. So again, the algebra $A = T_\Sigma$ of all Σ-trees is iterative. But also its subalgebra R_Σ of all rational Σ-trees is iterative, where recall that a Σ-tree is rational, if it has (up to isomorphism) only finitely many subtrees. Evelyn Nelson proved that for every set X the set $R_\Sigma X$ of all rational Σ-trees over X carries a free iterative algebra on X, and that the assignment $X \longmapsto R_\Sigma X$ of a free iterative algebra yields the free iterative theory on Σ. We will now introduce iterative algebras for every finitary endofunctor of an lfp category $\mathcal{A}$. Furthermore, we will show that every object of $\mathcal{A}$ generates a free iterative algebra, and we provide a canonical construction of it. We present in this subsection the first part of the paper [AMV$_2$], which is joint work with Jiří Adámek and Jiří Velebil.

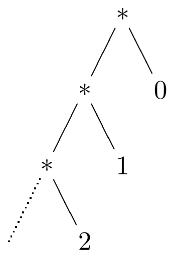
Assumption 3.16. For the rest of this subsection we assume that $\mathcal{A}$ is a locally finitely presentable category and that $H : \mathcal{A} \longrightarrow \mathcal{A}$ is a finitary endofunctor, see Section 2.1

Definition 3.17. A flat equation morphism $e : X \longrightarrow HX + A$ is called *finitary* if X is a finitely presentable object. An H-algebra $a : HA \longrightarrow A$ is called *iterative* if every finitary flat equation morphism e has a unique solution in A, i. e., there exists a unique morphism $e^\dagger : X \longrightarrow A$ such that Diagram (3.27) commutes.

Examples 3.18.

(i) Again, classical algebras are seldom iterative. In fact, notice that in Example 3.6(i) all equations are finitary.

(ii) All completely iterative algebras are obviously iterative. In particular, for a signature Σ, the algebras $T_\Sigma X$ of all Σ-trees over X are iterative H_Σ-algebras. The subalgebras $R_\Sigma X$ of all rational Σ-trees over X are iterative, too, but they usually fail to be completely iterative. For example, let Σ be a signature with a binary operation symbol $*$ and a constant c. Then for rational trees t and s the system (3.26) gives as solutions for x_1 and x_2 the trees in (3.25), which are rational. But the tree



is not rational. And so the system

$$x_i \approx x_{i+1} * i, \qquad i < \omega,$$

gives rise to a flat equation morphism which does not have a solution in $R_\Sigma \omega$.

(iii) The algebra of addition on

$$I = (0, \infty]$$

is iterative, see [AMV$_2$], but it fails to be completely iterative. The equation morphism e given by the system

$$\begin{aligned} x_0 &\approx x_1 + \frac{1}{2} \\ x_1 &\approx x_2 + \frac{1}{4} \\ x_2 &\approx x_3 + \frac{1}{8} \\ &\vdots \end{aligned}$$

has two solutions: one is $e^\dagger(x_n) = \infty$ ($n \in \mathbb{N}$), another one is $e^\dagger(x_n) = 2^{-n}$ ($n \in \mathbb{N}$).

(iv) Unary algebras. An algebra $\alpha : A \longrightarrow A$ of $H = Id$ on Set is iterative if and only if there exists a unique fixed point for each α^n, $n \geq 1$.

Notation 3.19. We denote by $\mathsf{Alg}_{it}\, H$ the full subcategory of $\mathsf{Alg}\, H$ that consists of all iterative H-algebras.

As for cias one proves that the choice of all algebra homomorphisms is appropriate for iterative algebras.

Lemma 3.20. ([AMV$_2$], Lemma 2.15)
Let (A, a) and (B, b) be iterative algebras. Then a morphism f is an H-algebra homomorphism if and only if it preserves solutions of finitary flat equation morphisms, i. e., the equation $(f \bullet e)^\dagger = f \cdot e^\dagger$ holds for all finitary $e : X \longrightarrow HX + A$.

The following result is not difficult to prove. It shows that there are enough iterative algebras to force the existence of free ones.

Proposition 3.21. ([AMV$_2$], Proposition 2.16)
Iterative algebras are closed under limits and filtered colimits.

Corollary 3.22. ([AMV$_2$], Corollaries 2.17 and 2.18)
The category $\mathsf{Alg}_{it}\, H$ is a reflective subcategory of $\mathsf{Alg}\, H$. Thus, every object of $\mathcal{A}$ generates a free iterative H-algebra.

In other words, the natural forgetful functor $U : \mathsf{Alg}_{it}\, H \longrightarrow \mathcal{A}$ has a left adjoint.

Definition 3.23. The finitary monad on $\mathcal{A}$ formed by free iterative H-algebras is called the *rational monad* of H and is denoted by $\mathbb{R} = (R, \eta, \mu)$.

Thus, $\mathbb{R}$ is the monad of the above adjunction

$$\begin{tikzcd} \mathsf{Alg}_{it}\, H \arrow[r, shift right, "U"'] \arrow[r, phantom, "\perp"] & \mathcal{A} \arrow[l, shift right] \end{tikzcd}$$

More detailed, for every object Y of $\mathcal{A}$ we denote by RY a free iterative H-algebra on Y with the universal arrow

$$\eta_Y : Y \longrightarrow RY\,,$$

and the algebra structure

$$\varrho_Y : HRY \longrightarrow RY\,.$$

Then $\mu_Y : RRY \longrightarrow RY$ is the unique homomorphism of H-algebras with $\mu_Y \cdot \eta_{RY} = id$.

Before turning to concrete examples of free iterative algebras, we will show that it is sufficient to describe the initial one:

Proposition 3.24. ([AMV$_2$], Proposition 2.20)
For every object Y of $\mathcal{A}$ the following are equivalent:

(i) RY is an initial iterative algebra for $H(\_) + Y$,

(ii) RY is a free iterative H-algebra on Y.

In fact, the proof for iterative algebras is the same as for completely iterative algebras, see Theorem 3.9.

Examples 3.25.

(i) The rational monad of $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$. Recall from Example 3.18(ii) that for every set Y the algebra $R_\Sigma Y$ of all rational Σ-trees over Y is iterative. As proved in [N], $R_\Sigma Y$ is a free iterative Σ-algebra on Y. Thus, the rational monad $\mathbb{R}_\Sigma$ of the polynomial endofunctor H_Σ of Set is given by the formation of the Σ-algebras $R_\Sigma Y$ of all rational Σ-trees over Y.

(ii) The rational monad of $\mathcal{P}_{\text{fin}} : \mathsf{Set} \longrightarrow \mathsf{Set}$, the finite power set functor was described in [AM$_1$]: it assigns to a set X the algebra of all rational strongly extensional finitely branching trees; here rational means, again, that there are (up to isomorphism) only finitely many different subtrees.

In Section 2 we have seen that initial algebras and initial cias (equivalently, final coalgebras) can be obtained via a canonical construction. From Corollary 3.22 we know that for every finitary endofunctor on $\mathcal{A}$ an initial iterative algebras exists. We shall now provide a construction of initial iterative algebras. Notice that by Proposition 3.24 this yields a construction of all free iterative algebras.

To motivate our construction recall that for a signature Σ the rational Σ-trees are precisely those Σ-trees which arise as solutions of finitary (flat) systems (3.23) where the right-hand side are terms over X that are not a variable from X.

Now for a finitary functor H on an lfp category $\mathcal{A}$ recall that by $\mathcal{A}_{\text{fp}}$ we denote a chosen set of representatives of all finitely presentable objects of $\mathcal{A}$ w. r. t. isomorphism. Consider the full subcategory EQ of $\mathsf{Coalg}\, H$ formed by all coalgebras $e : X \longrightarrow HX$ with a carrier from $\mathcal{A}_{\text{fp}}$, equivalently, all finitary flat equation morphisms in the initial object of $\mathcal{A}$. We will show that the initial iterative algebra is a colimit of the diagram

$$\mathsf{Eq} : \mathsf{EQ} \longrightarrow \mathcal{A}, \qquad (X, e) \longmapsto X\,.$$

We denote by

$$R_0 = \operatorname{colim} \mathsf{Eq}$$

a colimit of this diagram and we write $e^\sharp : X \longrightarrow R_0$, $e \in \mathsf{EQ}$, for the colimit injections. Notice that we obtain a unique morphism $i : R_0 \longrightarrow HR_0$ so that every $e^\sharp$ becomes a coalgebra homomorphism, i. e., the squares

$$\begin{tikzcd} X \arrow[r, "e"] \arrow[d, "e^\sharp"'] & HX \arrow[d, "He^\sharp"] \\ R_0 \arrow[r, "i"'] & HR_0 \end{tikzcd}$$

commute. In fact, it is easy to check that the morphisms $He^\sharp \cdot e$ form a cocone of Eq.

Theorem 3.26. ([AMV$_2$], Theorem 3.1)
An initial iterative algebra is given by the colimit

$$R_0 = \operatorname{colim} \mathsf{Eq}\,.$$

More precisely, i is an isomorphism whose inverse yields the structure $\varrho_0 : HR_0 \longrightarrow R_0$ of an initial iterative algebra.

Remark. Though easily stated, the proof of this theorem is quite non-trivial. We consider it as one of our main contributions to the subject. The construction of a free iterative algebra is essential for the proofs of most of the subsequent results on the rational monad $\mathbb{R}$ we present below.

Sketch of Proof. (1) We define a morphism

$$j : HR_0 \longrightarrow R_0\,.$$

Notice first that the diagram Eq is filtered. In fact, the category of all coalgebras is cocomplete, with colimits formed at the level of $\mathcal{A}$. Since $\mathcal{A}_{\text{fp}}$ is closed under finite colimits by Proposition 2.5, it follows that the category EQ is closed under finite colimits in the category of all H-coalgebras—thus, EQ is finitely cocomplete, whence filtered.

Consequently, H preserves the colimit of Eq: $HR_0 = \operatorname{colim} H \cdot \mathsf{Eq}$ with the colimit cocone $He^\sharp$. Since $\mathcal{A}$ is locally finitely presentable it follows that HR_0 is a colimit of the diagram D_{HR_0} of all arrows $p : P \longrightarrow HR_0$ where P is in $\mathcal{A}_{\text{fp}}$, see Proposition 2.9. Thus, in order to define j we need to define morphisms $j \cdot p : P \longrightarrow R_0$ forming a cocone of the diagram D_{HR_0}. We know that HR_0 is a filtered colimit of $H \cdot \mathsf{Eq}$ and that P is finitely presentable. Therefore, p factors through one of the colimit morphisms

$$\begin{tikzcd} P \arrow[r, "p"] \arrow[dr, "p'"'] & HR_0 \\ & HW \arrow[u, "Hg^\sharp"'] \end{tikzcd} \tag{3.29}$$

for some $g : W \longrightarrow HW$ in EQ, see Remark 2.3. We form a new object

$$e_{p'} \equiv P + W \xrightarrow{[p', g]} HW \xrightarrow{H\mathsf{inr}} H(P + W)$$

of EQ and define j to be the unique morphism such that the following square

$$\begin{tikzcd}
P \arrow[r, "\mathsf{inl}"] \arrow[d, "p"'] & P+W \arrow[d, "{e_{p'}}^\sharp"] \\
HR_0 \arrow[r, "j"'] & R_0
\end{tikzcd}$$

commutes for every p in $\mathcal{A}_{\text{fp}}/HR_0$. To prove that j is well-defined ones proves that

(i) ${e_{p'}}^\sharp \cdot \mathsf{inl}$ is independent of the choice of factorization (3.29), and

(ii) the morphisms ${e_{p'}}^\sharp \cdot \mathsf{inl}$ form a cocone of the diagram D_{HR_0}.

And then one readily shows that $j = i^{-1}$.

(2) R_0 is an iterative algebra with structure j. For every equation morphism

$$e : X \longrightarrow HX + R_0 = \operatorname{colim}(HX + \mathsf{Eq})$$

there exists, since X is finitely presentable, a factorization through the colimit morphism $HX + f^\sharp$ (for some $f : V \longrightarrow HV$ in EQ):

$$\begin{tikzcd}
X \arrow[r, "e"] \arrow[dr, "e_0"'] & HX + R_0 \\
& HX + V \arrow[u, "HX+f^\sharp"']
\end{tikzcd}$$

Recall from 3.1 that $\mathsf{can} : HX + HV \longrightarrow H(X+V)$ denotes the canonical morphism. Define a new object, $\overline{e}$, of EQ as follows:

$$\overline{e} \equiv X + V \xrightarrow{[e_0, \mathsf{inr}]} HX + V \xrightarrow{HX+f} HX + HV \xrightarrow{\mathsf{can}} H(X+V)\,.$$

We define a solution of e by

$$e^\dagger \equiv X \xrightarrow{\mathsf{inl}} X + V \xrightarrow{\overline{e}^\sharp} R_0\,.$$

A non-trivial proof now shows that $e^\dagger$ is indeed a solution of e and that it is unique, see [AMV$_2$], Lemma 3.5.

(3) Initiality of the iterative algebra (R_0, j). Let (A, α) be an iterative H-algebra. Consider the equation morphisms

$$X \xrightarrow{e} HX \xrightarrow{\mathsf{inl}} HX + A\,, \qquad e \text{ in } \mathsf{EQ}.$$

Their solutions $(\mathsf{inl} \cdot e)^\dagger : X \longrightarrow A$ form a cocone of Eq. The unique induced morphism $h : R_0 \longrightarrow A$ such that $h \cdot e^\sharp = (\mathsf{inl} \cdot e)^\dagger$ is the desired unique homomorphism of H-algebras (R_0, j) to (A, α). □

Corollary 3.27. ([AMV$_2$], Corollary 3.6)
A free iterative H-algebra RY is a colimit

$$RY = \operatorname{colim} \mathsf{Eq}_Y$$

of the diagram

$$\mathsf{Eq}_Y : \mathsf{EQ}_Y \longrightarrow \mathcal{A}\,,$$

where EQ_Y consists of all finitary equation morphisms $e : X \longrightarrow HX + Y$, and all coalgebra homomorphisms w.r.t. $H(-) + Y$, and Eq_Y sends (X, e) to X.

In fact, this is a consequence of Proposition 3.24 and Theorem 3.26.

Remark 3.28. We denote, again, the colimit morphisms of Eq_Y by

$$e^\sharp : X \longrightarrow RY$$

for all $e : X \longrightarrow HX + Y$ in EQ_Y. The appropriate isomorphism is denoted by

$$i_Y : RY \longrightarrow HRY + Y$$

It is characterized by the fact that the two coproduct injections of $HRY + Y$ are (in the notation of Definition 3.23)

$$\mathsf{inl} \equiv HRY \xrightarrow{\varrho_Y} RY \xrightarrow{i_Y} HRY + Y \qquad \text{and} \qquad \mathsf{inr} \equiv Y \xrightarrow{\eta_Y} RY \xrightarrow{i_Y} HRY + Y\,.$$

In other words, $i_Y = [\varrho_Y, \eta_Y]^{-1}$.

Example 3.29. Recall from Example 2.29(iii) that deterministic sequential automata with an input alphabet Σ are the coalgebras for the functor $HX = X^{\Sigma} \times 2$ on Set, and the final coalgebra T for H is carried by the set of all formal languages over Σ. Since H is a polynomial functor, an initial iterative algebra $R\emptyset$ is carried by a subset of T. For a finite automaton considered as a coalgebra $e : X \longrightarrow HX$ the map $e^{\sharp} : X \longrightarrow R\emptyset$ assigns to each state the language accepted by X with that state as the initial one. Thus, our construction above explains that $R\emptyset$ is the set of all formal languages accepted by automata with a finite state set, i. e., $R\emptyset$ consists of all regular languages.

4 Elgot Algebras

Unsheathe your dagger definitions.
James Joyce, *Ulysses*

In this section we present the results of our paper [AMV$_3$], which is joint work with Jiří Adámek and Jiří Velebil, again. We study Elgot algebras, a notion of algebra useful for application in the semantics of recursive computations. In fact, in Section 7 we shall use Elgot algebras to study the semantics of recursive program schemes such as (1.7) from the introduction, where two functions are recursively defined from the givens F and G. Recall that one has to distinguish between *uninterpreted* and *interpreted* semantics. In the uninterpreted semantics the givens are not functions but merely function symbols from a signature Σ. In the present section we prepare a basis for the interpreted semantics in which a program scheme comes together with a suitable Σ-algebra A, which gives an interpretation to all the given function symbols. From our discussion in the introduction we have seen that by a suitable Σ-algebra we should understand one in which all Σ-trees, or all rational ones, respectively, have a canonical computation. That means we want to describe precisely those algebras A with a canonical map $T_\Sigma A \longrightarrow A$, or $R_\Sigma A \longrightarrow A$.

More generally, we have seen in the previous section that we can abstract away from signatures and sets. For an endofunctor H of a category $\mathcal{A}$ with finite coproducts the final coalgebras TA for $H(\_) + A$ for every object A yield, if they exist, free cias. Analogously, for a finitary endofunctor H of a locally finitely presentable category $\mathcal{A}$ every object A of $\mathcal{A}$ generates a free iterative algebra RA. Our question then is: what is the largest category of H-algebras in which TA, or RA, respectively, act as free algebras on A? The answer in case of TA is: complete Elgot algebras. These are H-algebras with an additional function assigning to each flat equation morphism e a solution $e^\dagger$. Two (surprisingly simple) axioms are put on $(-)^\dagger$ which stem from the structure of TA: the free cias TA yield the monad $\mathbb{T}$, see Corollary 3.13, and complete Elgot algebras form precisely the category of Eilenberg-Moore algebras for $\mathbb{T}$, i. e., they are precisely those algebras with a "canonical" morphism $TA \longrightarrow A$. Moreover, for an object mapping T of $\mathcal{A}$ we shall add another equivalent description to (a)–(c) of the Introduction of Section 3; (a)–(c) are equivalent to

(d) for every object Y, TY is a free complete Elgot algebra on Y.

We also show that (non-complete) Elgot algebras form the Eilenberg-Moore category of the rational monad $\mathbb{R}$, see Definition 3.23. They are defined precisely as complete Elgot algebras, except that only finitary flat equation morphisms are considered. Basic examples of (complete) Elgot algebras include continuous algebras, metrizable algebras, and, of course, all (completely) iterative algebras.

The two axioms of (complete) Elgot algebras are inspired by the axioms of iteration theories of Stephen Bloom and Zoltán Ésik [BÉ]. In fact, we are able to draw a connection to those structures. For a signature Σ with a distinguished constant symbol $\perp$ the rational trees form an iteration theory whose iteration theory algebras are certain Eilenberg–Moore algebras for the rational monad R_Σ satisfying an extra extensionality property. However, not every Elgot algebra needs to satisfy this extra property, see Example 4.25 below.

4.1 Properties of (Completely) Iterative Algebras

Before we define Elgot algebras let us investigate properties of solutions in iterative algebras and cias.

Remark 4.1. We are going to prove two properties of iterative algebras and cias: the Functoriality and the Compositionality for solutions. We will use two "operations" on equation morphisms. One, $\bullet$, is just change of parameter names: given an equation morphism $e : X \longrightarrow HX + Y$ and a morphism $h : Y \longrightarrow Z$ we obtain the equation morphism $h \bullet e : X \longrightarrow HX + Z$, see Notation 3.3. The other operation $\blacksquare$ combines two flat equation morphisms

$$e : X \longrightarrow HX + Y \qquad \text{and} \qquad f : Y \longrightarrow HY + A$$

into the single flat equation morphism $f \blacksquare e : X + Y \longrightarrow H(X + Y) + A$ in a canonical way:

$$f \blacksquare e \equiv X + Y \xrightarrow{[e,\mathsf{inr}]} HX + Y \xrightarrow{HX+f} HX + HY + A \xrightarrow{\mathsf{can}+A} H(X + Y) + A, \tag{4.30}$$

4.2. Functoriality. This states that solutions are invariant under renaming of variables, provided, of course, that the right-hand sides of equations are renamed accordingly. Formally, observe that every flat equation morphism is a coalgebra for the endofunctor $H(-)+A$. Given two such coalgebras e and f, a renaming of the variables (or *morphism of equations*) is a morphism $h : X \longrightarrow Y$ which forms a coalgebra homomorphism:

$$\begin{tikzcd}
X \arrow[r, "e"] \arrow[d, "h"'] & HX+A \arrow[d, "Hh+A"] \\
Y \arrow[r, "f"'] & HY+A
\end{tikzcd} \tag{4.31}$$

The Functoriality states that

$$e^\dagger = f^\dagger \cdot h \tag{4.32}$$

holds for all equation morphisms h from e to f. In other words: $(-)^\dagger$ is a functor from the category of all flat equation morphisms in the algebra A into the comma-category of the object A.

Lemma 4.3. ([AMV$_3$], Lemma 2.16)
In every cia the assignment $(-)^\dagger$ is functorial.

Proof. For each morphism h of equations the diagram

$$\begin{tikzcd}
X \arrow[r, "h"'] \arrow[d, "e"'] \arrow[rr, bend left, "f^\dagger\cdot h"] & Y \arrow[r, "f^\dagger"'] \arrow[d, "f"'] & A \\
HX+A \arrow[r, "Hh+A"] \arrow[rr, bend right, "H(f^\dagger\cdot h)+A"'] & HY+A \arrow[r, "Hf^\dagger+A"] & HA+A \arrow[u, "{[\alpha,A]}"']
\end{tikzcd}$$

commutes. Thus, $f^\dagger \cdot h$ is a solution of e. Uniqueness of solutions now implies the desired result. □

Remark 4.4. The same holds for every iterative algebra, except that there we restrict X and Y in 4.2 to finitely presentable objects.

4.5. Compositionality. This tells us how to perform simultaneous recursion: given an equation morphism f in A with a variable object Y, we can combine it with any equation morphism e in Y with a variable object X to obtain the equation morphism $f \blacksquare e$ in A of Remark 4.1. The Compositionality decrees that the left-hand component of $(f \blacksquare e)^\dagger$ is just the solution of $f^\dagger \bullet e$, i. e., in lieu of solving f and e simultaneously we first solve f, plug in the solution in e and solve the resulting equation morphism.

In symbols: given $f : Y \longrightarrow HY + A$ and $e : X \longrightarrow HX + Y$ the Compositionality states that

$$\left(f^\dagger \bullet e\right)^\dagger = (f \blacksquare e)^\dagger \cdot \mathsf{inl} \tag{4.33}$$

Remark 4.6. Notice that the coproduct injection $\mathsf{inr} : Y \longrightarrow X + Y$ is a morphism of equations from f to $f \blacksquare e$. The Functoriality then implies that $f^\dagger = (f \blacksquare e)^\dagger \cdot \mathsf{inr}$. Thus, in the presence of Functoriality, the Compositionality is equivalent to

$$(f \blacksquare e)^\dagger = [(f^\dagger \bullet e)^\dagger, f^\dagger]\,. \tag{4.34}$$

Lemma 4.7. ([AMV$_3$], Lemma 2.20)
In every cia the assignment $(-)^\dagger$ satisfies the Compositionality.

Remark 4.8. The same holds for every iterative algebra, except that there we restrict X and Y in 4.5 to finitely presentable objects.

Remark 4.9. As mentioned in the introduction of this section, our two axioms, Functoriality and Compositionality, are not new as ideas of axiomatizing recursion—we believe however, that their concrete form is new, and their motivation strengthened by the results below.

The Functoriality resembles the "functorial dagger implication" of Stephen Bloom and Zoltán Ésik [BÉ], 5.3.3. And the Compositionality resembles the "left pairing identity" of [BÉ], 5.3.1. This identity corresponds also to the Bekić-Scott identity, see e. g. [Mo$_1$], 2.1.

4.2 The Category of Elgot Algebras

Definition 4.10. Let H be an endofunctor of a category with finite coproducts. An *Elgot algebra* is an H-algebra $\alpha : HA \longrightarrow A$ together with a function $(-)^\dagger$ which to every finitary flat equation morphism

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

assigns a solution $e^\dagger : X \longrightarrow A$ in such a way that the Functoriality (4.32) and the Compositionality (4.33) are satisfied.

By a *complete Elgot algebra* we analogously understand an H-algebra together with a function $(-)^\dagger$ assigning to every flat equation e a solution $e^\dagger$ so that Functoriality and Compositionality are satisfied.

Examples 4.11.

(i) Every join semilattice A is an Elgot algebra. More precisely: consider the polynomial endofunctor $HX = X \times X$ of Set (expressing one binary operation). Then for every join semilattice A there is a "canonical" structure of an Elgot algebra on A obtained as follows: the algebra RA of all rational binary trees on A has an interpretation on A given by the function $\alpha : RA \longrightarrow A$ forming, for every rational binary tree t the join of all the (finitely many) labels of leaves of t in A. Now given a finitary flat equation morphism $e : X \longrightarrow X \times X + A$, it has a unique solution $e^\dagger : X \longrightarrow RA$ in the free iterative algebra RA, and composed with α this yields a structure $e \longmapsto \alpha \cdot e^\dagger$ of an Elgot algebra on A. See Example 4.24 for a proof. Recall that in contrast, no nontrivial join semilattice is iterative: it has too many idempotents, i. e., solutions of $x \approx x \vee x$, see Example 3.6(i).

(ii) Continuous algebras on cpos are complete Elgot algebras. Let us work here in the category

$$\mathsf{CPO}$$

of all *ω-complete posets*, i.e., posets (not necessarily with a least element) having joins of increasing ω-chains; morphisms are the *continuous functions*, i.e., functions preserving joins of ω-chains. Observe that the category CPO has coproducts: they are the disjoint unions with elements of different summands incompatible.

A functor $H : \mathsf{CPO} \longrightarrow \mathsf{CPO}$ is called *locally continuous* provided that for arbitrary cpos, X and Y, the derived function from $\mathsf{CPO}(X,Y)$ to $\mathsf{CPO}(HX,HY)$ is continuous (i.e., $H(\bigsqcup f_n) = \bigsqcup Hf_n$ holds for all increasing ω-chains $f_n : X \longrightarrow Y$). For example, every polynomial endofunctor $X \longmapsto \coprod_n \Sigma_n \times X^n$ of CPO (where Σ_n are cpos) is locally continuous.

Let $H : \mathsf{CPO} \longrightarrow \mathsf{CPO}$ be a locally continuous functor. Every H-algebra $\alpha : HA \longrightarrow A$ with a least element $\perp \in A$ is a complete Elgot algebra provided that to every equation morphism e the least solution $e^\dagger$ is assigned, see [AMV$_3$], Proposition 3.5. Notice that the least solution of $e : X \longrightarrow HX + A$ refers to the element-wise order of the hom-set $\mathsf{CPO}(X, A)$. We can actually prove a concrete formula for $e^\dagger$ as a join of the ω-chain

$$e^\dagger = \bigsqcup_{n\in\omega} e_n^\dagger$$

of "approximations": $e_0^\dagger$ is the constant function to $\perp$, the least element of A, and given $e_n^\dagger$, then $e_{n+1}^\dagger$ is defined by the commutativity of Diagram (3.28).

(iii) Many set functors H have a lifting to locally continuous endofunctors H' of CPO. That is, for the forgetful functor $U : \mathsf{CPO} \longrightarrow \mathsf{Set}$ the following square

$$\begin{tikzcd}
\mathsf{CPO} \arrow[r, "H'"] \arrow[d, "U"'] & \mathsf{CPO} \arrow[d, "U"] \\
\mathsf{Set} \arrow[r, "H"'] & \mathsf{Set}
\end{tikzcd}$$

commutes. For example, every polynomial functor H_Σ has such a lifting. Let $\alpha : HA \longrightarrow A$ be an H-algebra such that there exists a CPO-ordering $\sqsubseteq$ with a least element on the set A such that α is a continuous function from $H'(A, \sqsubseteq)$ to $(A, \sqsubseteq)$. Then A is a complete Elgot algebra for H. In fact, to every equation morphism $e : X \longrightarrow HX+A$ assign the least solution of $e : (X, \leq) \longrightarrow H'(X, \leq)+(A, \sqsubseteq)$ where $\leq$ is the discrete ordering of X ($x \leq y$ if and only if $x = y$). We shall see in Example 4.20 below that not every complete Elgot algebra needs to arise as a CPO-enrichable one.

(iv) Unary algebras. Let $H = Id$ as an endofunctor of Set. Given an H-algebra $\alpha : A \longrightarrow A$, if α has no fixed point, then A carries no structure of an Elgot algebra: consider the formal equation $x \approx \alpha(x)$.

Conversely, every fixed point a_0 of α yields a flat cpo structure with a least element a_0 on A, i. e., $x \leq y$ if and only if $x = y$ or $x = a_0$. Thus, A is a complete Elgot algebra since it is CPO-enrichable.

(v) Every complete lattice A is a complete Elgot algebra for $HX = X \times X$. Analogously to Example 4.11(i) we have a function $\alpha : TA \longrightarrow A$ assigning to every binary tree t in TA the join of all labels of leaves of t in A. Now for every flat equation morphism e in A we have its unique solution $e^\dagger$ in TA and this yields a structure $e \longmapsto \alpha \cdot e^\dagger$ of a complete Elgot algebra. See Example 4.19 for a proof.

4.3 The Eilenberg-Moore Algebras for the Monad $\mathbb{T}$

Recall our standing assumptions that H is an endofunctor of a category $\mathcal{A}$ with finite coproducts, and recall the operation $\bullet$ and $\blacksquare$ of Remark 4.1.

Definition 4.12. A *homomorphism* h from a complete Elgot algebra $(A, a, (-)^\dagger)$ to a complete Elgot algebra $(B, b, (-)^\ddagger)$ is a morphism $h : A \longrightarrow B$ preserving solutions: for each $e : X \longrightarrow HX + A$ we have the following equation

$$X \xrightarrow{e^\dagger} A \xrightarrow{h} B \equiv X \xrightarrow{(h \bullet e)^\ddagger} B\,. \tag{4.35}$$

Lemma 4.13. ([AMV$_3$], Lemma 5.2)
Every homomorphism $h : (A, a, (-)^\dagger) \longrightarrow (B, b, (-)^\ddagger)$ of complete Elgot algebras is a homomorphism of H-algebras.

Example 4.14. The converse of Lemma 4.13 is true for completely iterative algebras, as proved in Proposition 3.4, but for complete Elgot algebras in general it is false. In fact, consider the unary algebra $id : A \longrightarrow A$, where $A = \{0, 1\}$. This is a complete Elgot algebra with the solution structure $(-)^\dagger$ given by the fixed point $0 \in A$, see Example 4.11(v).

Then $\mathrm{const}_1 : A \longrightarrow A$ is a homomorphism of unary algebras that does not preserve solutions. Indeed, consider the following equation morphism

$$e : \{x\} \longrightarrow \{x\} + A, \quad x \longmapsto x.$$

We have $e^\dagger(x) = 0$, and thus $1 = \mathrm{const}_1 \cdot e^\dagger(x) \neq (\mathrm{const}_1 \bullet e)^\dagger(x) = e^\dagger(x) = 0$.

Notation 4.15. We denote by

$$\mathsf{Alg}_c^\dagger\, H$$

the the (non-full) subcategory of $\mathsf{Alg}\, H$ formed by all complete Elgot algebras and their homomorphism.

The following result establishes that statement (d) from the introduction of this section is equivalent to (a)–(c) in Section 3, see also Corollary 3.10.

Theorem 4.16. ([AMV$_3$], Theorem 5.3)
Let Y be an object of $\mathcal{A}$. Then the following are equivalent:

(1) TY is a final coalgebra for $H(\_) + Y$, and

(2) TY is a free complete Elgot algebra on 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, see Corollary 3.10, and therefore an Elgot algebra.

Furthermore, $(TY, \tau_Y, (-)^\dagger)$ is a free Elgot algebra on Y. For any given Elgot algebra $(A, a, (-)^\ddagger)$ and any given 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 5.3 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 [AMV$_3$]. Denote by α_Y its inverse. Then (TY, τ_Y) is a cia; i. e., for every 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 the 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 for $H(\_) + Y$. 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. □

The proof of this Theorem 4.16 is once again non-trivial, see [AMV$_3$] for the details. It is another important contribution and it is the key ingredient needed to prove the following result. Recall that a functor is called iteratable, if the final coalgebra TX for $H(\_) + X$ exists for every object X of $\mathcal{A}$.

Theorem 4.17. ([AMV$_3$], Theorem 5.7)
If H is an iteratable functor, then the category $\mathsf{Alg}^\dagger_c\, H$ of complete Elgot algebras is isomorphic to the Eilenberg–Moore category $\mathcal{A}^{\mathbb{T}}$ of monadic $\mathbb{T}$-algebras (for the free cia monad $\mathbb{T}$ of H).

Remark 4.18. By Theorem 4.16, the natural forgetful functor $U : \mathsf{Alg}^\dagger_c\, H \longrightarrow \mathcal{A}$ has a left adjoint $Y \longmapsto TY$. Thus, the monad obtained from this adjunction is $\mathbb{T}$, see Corollary 3.13. To prove that the comparison functor $\mathsf{Alg}^\dagger_c\, H \longrightarrow A^{\mathbb{T}}$ is an isomorphism the easiest proof we know uses Beck's Theorem, see Theorem 2.50. But it is not very intuitive. A slightly more technical (and much more illuminating) proof has the following sketch: Denote for any object Y by $(TY, \tau_Y, (-)^\ddagger)$ a free Elgot algebra on Y with a universal arrow $\eta_Y : Y \longrightarrow TY$.

(i) For every $\mathbb{T}$-algebra $a_0 : TA \longrightarrow A$ we have an "underlying" H-algebra

$$\alpha \equiv HA \xrightarrow{H\eta_A} HTA \xrightarrow{\tau_A} TA \xrightarrow{a_0} A,$$

and the following formula for solving equations: given a flat equation morphism $e : X \longrightarrow HX + A$ put

$$e^\dagger \equiv X \xrightarrow{(\eta_A \bullet e)^\ddagger} TA \xrightarrow{a_0} A\,.$$

It is not difficult to see that this formula indeed yields a choice of solutions satisfying the Functoriality and the Compositionality.

(ii) Conversely, given a complete Elgot algebra $(A, a, (-)^\dagger)$, define $a_0 : TA \longrightarrow A$ as the solution $\alpha^\dagger_A$ of the structure of the final coalgebra $\alpha_A : TA \longrightarrow HTA + A$ considered as a flat equation morphism in A. One readily proves that a_0 satisfies the two axioms of an Eilenberg-Moore algebra.

(iii) It is necessary to prove that the above passages extend to the level of morphisms and they form functors which are mutually inverse.

Example 4.19. Let A be a complete lattice. Recall from Example 4.11(vi) the function $\alpha : TA \longrightarrow A$ assigning to every binary tree t in TA the join of all labels of leaves of t in A. Since joins commute with joins it follows that $\alpha : TA \longrightarrow A$ is the structure of an Eilenberg-Moore algebra on A. Thus, A is a complete Elgot algebra as described in Example 4.11(vi).

Example 4.20. Not every complete Elgot algebra for an endofunctor of Set needs to be CPO-enrichable. To see this consider the polynomial endofunctor $HX = X + X + 1$ expressing two unary operations s and t and a constant $\perp$. Its free cia monad T assigns to every set X the set $TX = \{\, s, t\,\}^* \times X + \{\, s, t\,\}^\infty$, where S^∞ denotes the set of all (finite and infinite) sequences of elements of S and S^* denotes the set of all finite such, respectively.

Now consider the set $C = \{\, 0, 1\,\}$ equipped with the following algebra structure

$$\begin{array}{rcl} \gamma : TC & \longrightarrow & C \\ (w, i) & \longmapsto & i, \qquad i = 0,1,\ w \in \{\, s,t\,\}^* \\ v & \longmapsto & \begin{cases} 1 & \text{if } v = up,\ u \in \{\, s,t\,\}^*,\ p = ttt\ldots, \\ 0 & \text{else} \end{cases} \end{array}$$

It is not difficult to check that this is a structure of an Eilenberg-Moore algebra for the monad T. Thus, C is a complete Elgot algebra by Theorem 4.17.

The two formal equations

$$x \approx s(x) \qquad \text{and} \qquad x \approx t(x) \tag{4.36}$$

give rise to two flat equation morphism $e, f : 1 \longrightarrow H1 + C$; e and f map the unique element of 1 to the left-hand and middle component of $H1 = 1 + 1 + 1$, respectively. From the definition of $(-)^\dagger$ on C as in Remark 4.18 we see that $e^\dagger : 1 \longrightarrow C$ and $f^\dagger : 1 \longrightarrow C$ pick 0 and 1 in C, respectively. However, if C were CPO-enrichable both solutions would have to choose the smallest element in the order on C. Thus, there does not exist any order of C such that C is a continuous algebra and the Elgot algebra structure associated to γ above assigns to each flat equation morphism its least solution.

4.4 The Eilenberg-Moore Algebras for the Monad $\mathbb{R}$

Throughout this subsection H denotes a finitary endofunctor of a locally finitely presentable category $\mathcal{A}$.

In the following result the concept of *homomorphism of Elgot algebras* is defined analogously to Definition 4.12: the equation (4.35) is required to hold for *finitary* flat equation morphisms e only. We denote by

$$\mathsf{Alg}^\dagger H$$

the category of all Elgot algebras and their homomorphisms.

Lemma 4.21. ([AMV$_3$], Lemma 4.2)
Every homomorphism $h : (A, a, (-)^\dagger) \longrightarrow (B, b, (-)^\ddagger)$ of Elgot algebras is a homomorphism of H-algebras.

Notice that although the statement of this result is similar to Lemma 4.13, its proof uses slightly different techniques, see [AMV$_3$]. The converse of Lemma 4.21 is again false, see Example 4.14.

Proposition 4.22. ([AMV$_3$], Proposition 4.6)
A free iterative algebra on Y is a free Elgot algebra on Y.

Sketch of Proof. Since every iterative algebra is an Elgot algebra one only has to prove that a free iterative algebra RY has the universal property. Suppose that $(A, \alpha, (-)^\dagger)$ is an Elgot algebra and let $m : Y \longrightarrow A$ be a morphism. Recall the construction of RY from Theorem 3.26. Define a morphism $h : RY \longrightarrow A$ by commutativity of the following triangles

$$\begin{tikzcd}
RY \arrow[r, "h"] & A \\
X \arrow[u, "e^\sharp"] \arrow[ur, "(m \bullet e)^\dagger"'] &
\end{tikzcd}$$

for all $e : X \longrightarrow HX + Y$ in EQ_Y. In fact, the morphisms $(m \bullet e)^\dagger$ form a cocone of Eq_Y, whence h is well-defined. Now one has to show that h preserves solutions, satisfies the equation $h \cdot \eta_Y = f$, and is uniquely determined, see [AMV$_3$]. $\square$

Theorem 4.23. ([AMV$_3$], Theorem 4.7)
The category $\mathsf{Alg}^\dagger H$ of Elgot algebras is isomorphic to the Eilenberg-Moore category $\mathcal{A}^\mathbb{R}$ of $\mathbb{R}$-algebras for the rational monad $\mathbb{R}$ of H.

Sketch of Proof. The proof of this theorem is analogous to the proof of Theorem 4.17. By Proposition 4.22 the natural forgetful functor $V : \mathsf{Alg}^\dagger H \longrightarrow \mathcal{A}$ has a left adjoint $Y \longmapsto RY$. Thus, the monad obtained by this adjunction is the rational monad $\mathbb{R}$. Ones proves that the comparison functor $\mathsf{Alg}^\dagger H \longrightarrow \mathcal{A}^\mathbb{R}$ is an isomorphism, using Beck's theorem, see Theorem 2.50. $\square$

Example 4.24. Let A be a join semilattice. Recall from Example 4.11(i) the function $\alpha : RA \longrightarrow A$ assigning to a rational binary tree t in RA the join of the labels of all leaves of t in A. Since joins commute with joins it follows that this is the structure of an Eilenberg-Moore algebra on A. Thus, A is an Elgot algebra as described in Example 4.11(i).

Example 4.25. Iteration algebras of [BÉ] are Elgot algebras. More precisely, let Σ be a signature with a special chosen constant symbol $\perp$. Then the algebraic theory R_Σ formed by all rational Σ-trees is an iteration theory, see [BÉ], Example 6.4.8. Recall that any morphism $n \longrightarrow m$ in this algebraic theory is essentially just a morphism $n \longrightarrow R_\Sigma m$ in the Kleisli category of the monad (associated to) R_Σ, where n and m are considered as sets in the usual way. Furthermore, every interpretation $i : n \longrightarrow A$ of variables in an Eilenberg-Moore algebra A of the monad R_Σ gives rise to a unique homomorphism $\widehat{i} : R_\Sigma n \longrightarrow A$ extending i. Now an R_Σ-iteration algebra in the sense of [BÉ], Definition 7.1.1, is an Eilenberg-Moore algebra (A, α) of the monad R_Σ that satisfies the following extensionality property: Suppose that $f, g : n \longrightarrow n + m$ are two morphisms in the algebraic theory R_Σ, i. e., morphisms $f, g : n \longrightarrow R_\Sigma(n + m)$ in the Kleisli category of the monad R_Σ. If for all interpretations $i : n + m \longrightarrow A$ we have a commuting square

$$\begin{tikzcd}
n + m \arrow[r, "f"] \arrow[d, "g"'] & R_\Sigma(n+m) \arrow[d, "\widehat{i}"] \\
R_\Sigma(n+m) \arrow[r, "\widehat{i}"'] & A
\end{tikzcd}$$

then also for all $j : m \longrightarrow A$ the square

$$\begin{tikzcd}
n \arrow[r, "f^\dagger"] \arrow[d, "g^\dagger"'] & R_\Sigma m \arrow[d, "\widehat{j}"] \\
R_\Sigma m \arrow[r, "\widehat{j}"'] & A
\end{tikzcd}$$

commutes, where $f^\dagger$ and $g^\dagger$ are the solutions of f and g, respectively, in the iteration theory R_Σ.

Obviously, any R_Σ-iteration algebra is an Elgot algebra for the polynomial functor associated to the signature Σ. But not conversely. In fact, for the signature Σ with two unary operation symbols s and t and the constant symbol $\perp$, the algebra (C, γ) of Example 4.20 yields by precomposition with the inclusion $R_\Sigma C \hookrightarrow T_\Sigma C$ an Eilenberg-Moore algebra for R_Σ, whence an Elgot algebra for H_Σ. However, the two formal equations (4.36) give rise to morphisms $1 \longrightarrow R_\Sigma(1 + 0)$ that violate the desired extensionality property. In fact, we have $\widehat{i} \cdot f = \widehat{i} \cdot g = i$ for all $i : 1 + 0 \longrightarrow C$. But $\widehat{j} \cdot f^\dagger$ picks 0 while $\widehat{j} \cdot g^\dagger$ picks 1 for the unique $j : 0 \longrightarrow C$.

5 Solution Theorems

If you are not part of the solution,
you are part of the problem.
Eldridge Cleaver, *speech in San Francisco*, 1968

In Section 3.1 and 3.2 we started by considering non-flat systems (3.23) of formal recursive equations for Σ-algebras. We then argued that due to the possibility of flattening such a system it suffices to consider flat equation morphisms $X \longrightarrow HX + A$ in general. We shall make this more precise in this section by showing that for all (completely) iterative algebras much more general systems of recursive equations have unique solutions. For polynomial endofunctors of Set, this implies that our notion of iterative algebra coincides with that presented by Evelyn Nelson [N]. As we shall see in the Section 6, this also implies that the rational monad $\mathbb{R}$ is iterative and that the free cia monad $\mathbb{T}$ is completely iterative.

Notice first that the requirement stated in (3.23) that no right-hand side of a system is a variable is important, the equation $x \approx x$ has a unique solution only in the trivial one-point algebra. Systems satisfying the above condition are called *guarded.*

In the current section we present the result of Sections 3 and 4 of [M$_1$] and [AMV$_2$], respectively. We start with completely iterative algebras.

5.1 Solution Theorem for Completely Iterative Algebras

We assume in this subsection that $\mathcal{A}$ is a category with finite coproducts. Recall from Corollary 3.13 that for an iteratable endofunctor H, free cias exist on every object Y of $\mathcal{A}$. As before we denote by

$$\tau_Y : HTY \longrightarrow TY \qquad \text{and} \qquad \eta_Y : Y \longrightarrow TY$$

the structure and universal arrow of the free cia TY. Moreover, recall from Remark 3.15 that for every cia (A, a) we denote by

$$\widetilde{a} : TA \longrightarrow A$$

the unique H-algebra homomorphism with $\widetilde{a} \cdot \eta_A = id$.

Definition 5.1. By an *equation morphism* is meant a morphism

$$e : X \longrightarrow T(X + A)$$

in $\mathcal{A}$ where X is any object (of "variables") and A is any object (of "parameters").

The equation morphism e is called *guarded* if it factors through the summand $HT(X + A) + A$ of $HT(X + A) + X + A = T(X + A)$ (see Corollary 3.10(ii)):

$$\begin{tikzcd} X \arrow[r, "e"] \arrow[dr, dotted, "e_0"'] & T(X+A) \\ & HT(X+A)+A \arrow[u, "{[\tau_{X+A}, \eta_{X+A} \cdot \mathsf{inr}]}"'] \end{tikzcd}$$

Given a cia (A, a) and an equation morphism e we call a morphism $e^\dagger : X \longrightarrow A$ a solution of e in A if the square

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[d, "e"'] & A \\ T(X+A) \arrow[r, "{T[e^\dagger, A]}"'] & TA \arrow[u, "\widetilde{a}"'] \end{tikzcd} \tag{5.37}$$

commutes.

Remark 5.2. For a polynomial endofunctor H_Σ of Set an equation morphism e is the same as a system (3.23) where all right-hand sides t_i are Σ-trees over $X + A$, and guardedness of e corresponds precisely to the requirement that no t_i be a variable from X. The commutativity of (5.37) means that the assignment $e^\dagger$ of variables of X to elements of A has the following property: form first the "substitution" mapping $[e^\dagger, A] : X + A \longrightarrow A$ (which replaces variables by their solution according to $e^\dagger$ and leaves parameters unchanged). Apply this substitution to the right-hand sides of the given system e of formal equations, and compute the resulting infinite trees in A. This yields the same assignment of variables to elements of A as $e^\dagger$. That means that the formal equations $x \approx e(x)$ become actual identities in A after the substitution $x \longmapsto e^\dagger(x)$ is performed on both sides of the equations and the right-hand side is evaluated in A.

Formally, one extends $[e^\dagger, A]$ to the unique homomorphism

$$\widetilde{a} \cdot T[e^\dagger, A] : T(X+A) \longrightarrow A$$

from the free cia on $X + A$ to A. Precomposed with e it yields the morphism $e^\dagger$.

Theorem 5.3. *In every cia, for every guarded equation morphism there exists a unique solution.*

Sketch of Proof. Let (A, a) be a cia. Given a guarded equation morphism e with a factor e_0 as in Definition 5.1, form the flat equation morphism

$$\overline{e} \equiv T(X+A) \xrightarrow{[\tau,\eta]^{-1}} HT(X+A) + X + A \xrightarrow{[\mathsf{inl}, e_0, \mathsf{inr}]} HT(X+A) + A$$

and consider its unique solution $s : T(X+A) \longrightarrow A$ in the cia A. One readily proves that the morphism

$$e^\dagger \equiv X \xrightarrow{\mathsf{inl}} X + A \xrightarrow{\eta} T(X+A) \xrightarrow{s} A$$

is the desired unique solution of e in A. □

Remark 5.4. We have chosen to present in this summary a slightly different version of the solution theorem from [M$_1$]. However, the above Theorem 5.3 follows immediately from Theorem 3.9 of [M$_1$].

5.2 Solution Theorem for Iterative Algebras

It is well-known that for an iterative algebra for a polynomial functor of Set one can uniquely solve guarded system (3.23) where the set of variables is finite and where all right-hand sides t_i are rational trees. We shall now present the corresponding result in our more general setting. We assume in this subsection that $\mathcal{A}$ is an lfp category and that H is a finitary endofunctor of $\mathcal{A}$.

Definition 5.5. By a *rational equation morphism* in an object A we mean a morphism

$$e : X \longrightarrow R(X+A), \qquad X \text{ finitely presentable},$$

and e is called *guarded* if it factors through the summand $HR(X+A) + A$ of $R(X+A) = HR(X+A)+X+A$ (see Remark 3.28):

$$\begin{tikzcd} X \arrow[r, "e"] \arrow[dr, dotted, "e_0"'] & R(X+A) \\ & HR(X+A)+A \arrow[u, "{[\varrho, \eta\cdot\mathsf{inr}]}"'] \end{tikzcd}$$

Suppose that A is an underlying object of an iterative H-algebra $a : HA \longrightarrow A$. We denote (analogously to $\widetilde{a}$ in the previous subsection) by

$$\widehat{a} : RA \longrightarrow A$$

the unique homomorphism of H-algebras with $\widehat{a} \cdot \eta_A = id$. Then by a *solution* of e in the iterative algebra A is meant a morphism $e^\dagger : X \longrightarrow A$ in $\mathcal{A}$ such that the square

$$\begin{tikzcd} X \arrow[r, "e^\dagger"] \arrow[d, "e"'] & A \\ R(X+A) \arrow[r, "{R[e^\dagger, A]}"'] & RA \arrow[u, "\widehat{a}"'] \end{tikzcd}$$

commutes.

Theorem 5.6. ([AMV$_2$], Theorem 4.6)
If A is an iterative H-algebra, then every guarded rational equation morphism e in A has a unique solution.

Sketch of Proof. Let $a : HA \longrightarrow A$ be an iterative algebra and let e be a guarded rational equation morphism with a factor e_0 as in Definition 5.5.

Recall from Corollary 3.27 that $R(X + A) = \operatorname{colim} \mathsf{Eq}_{X+A}$ with colimit cocone $g^\sharp : W \longrightarrow R(X + A)$ for all $g : W \longrightarrow HW + X + A$ in EQ_{X+A}. Since this colimit is filtered and H is finitary, we have a filtered colimit $HR(X + A) + A = \operatorname{colim} H\mathsf{Eq}_{X+A} + A$ with the colimit cocone formed by all $Hg^\sharp + A$. Since X is a finitely presentable object, the morphism $e_0 : X \longrightarrow \operatorname{colim} H\mathsf{Eq}_{X+A} + A$ factors through the colimit cocone:

$$\begin{tikzcd}
X \arrow[r, "e_0"] \arrow[dr, "w"'] & HR(X + A) + A \\
& HW + A \arrow[u, "Hg^\sharp + A"']
\end{tikzcd}$$

for some object $g : W \longrightarrow HW + X + A$ of EQ_{X+A} and some morphism w.

We define a finitary flat equation morphism as follows:

$$\overline{e} \equiv W + X \xrightarrow{[g,\mathsf{inm}]} HW + X + A \xrightarrow{[\mathsf{inl},w,\mathsf{inr}]} HW + A \xrightarrow{H\mathsf{inl}+A} H(W + X) + A$$

where $\mathsf{inm} : X \longrightarrow HW + X + A$ is the middle coproduct injection. Now consider the unique solution $\overline{e}^\dagger : W + X \longrightarrow A$ in the iterative algebra A. A non-trivial proof shows that the morphism

$$e^\dagger \equiv X \xrightarrow{\mathsf{inr}} W + X \xrightarrow{\overline{e}^\dagger} A$$

is the desired unique solution of e. □

Remark 5.7. It follows from Theorem 5.6 that for a polynomial endofunctor H_Σ of Set our notion of iterative algebra coincides with that of Evelyn Nelson [N]. In fact, she required that iterative algebras have unique solutions of guarded systems (3.23) with finitely many variables where all right-hand sides t_i are terms over $X + A$, i. e. elements of a free H_Σ-algebra on $X + A$. It is not difficult to formulate this for an arbitrary finitary functor on our lfp category $\mathcal{A}$. Indeed, recall that every object X of $\mathcal{A}$ generates a free algebra FX, see Theorem 2.25. Now one formulates the notions of *finitary* equation morphism, solution and guardedness by replacing R by F in Definition 5.5. Then Theorem 5.6 implies that an H-algebra is iterative if and only if it admits unique solutions of all guarded finitary equation morphisms. For details, see [AMV$_2$], Definition 4.2 and Theorem 4.4.

6 Iterative and Completely Iterative Monads

Und diese Monaden sind die wahrhaften Atomi der Natur
und mit einem Worte / die Elemente derer Dinge.
Gottfried Wilhelm Leibniz, *Monadologie*, 1724

For a polynomial endofunctor of Set Evelyn Nelson [N] has proved that the algebraic theory induced by free iterative algebras is a free iterative theory in the sense of Calvin Elgot [E]. More generally, in this section we will show that the rational monad $\mathbb{R}$ of a finitary endofunctor H, introduced in Definition 3.23, is iterative and it can be characterized as the free iterative monad on H. Similarly, for an iteratable endofunctor H, i. e., all free cias TY of H exist, the free cia monad $\mathbb{T}$, see Corollary 3.13, is a free completely iterative monad on H.

As we shall see the universal property of the monads $\mathbb{R}$ and $\mathbb{T}$ yields an abstract version of the notion of second-order substitution, see Example 2.44. In fact, for a polynomial endofunctor H_Σ of Set the freeness of $\mathbb{T}$ yields second-order substitution of all Σ-trees, and the freeness of $\mathbb{R}$ means that rational trees are closed under this second-order substitution. The (generalized) second-order substitution is a key ingredient of our treatment of the semantics of recursive program schemes in Section 7.

The current section presents the main results of the two papers [AMV$_2$] and [M$_1$].

6.1 Ideal Monads

For a polynomial endofunctor of Set induced by a signature Σ it follows from the universal property of the free completely iterative algebras $T_\Sigma X$ that substitution of trees for variables works for all Σ-trees in precisely the same way as for terms (i. e., all finite Σ-trees), and it follows from the universal property of free iterative algebras $R_\Sigma X$ that rational Σ-trees are closed under substitution. For example, let X be a set of variables and let $s : X \longrightarrow T_\Sigma Y$ be a mapping that assigns to each variable its substitute, a Σ-tree over Y. The unique induced homomorphism $\widehat{s} : T_\Sigma X \longrightarrow T_\Sigma Y$ from the free cia $T_\Sigma X$ on X to $T_\Sigma Y$ with $\widehat{s} \cdot \eta_Y = s$ performs on any given tree of $T_\Sigma X$ the substitution of variables according to s to obtain a tree of $T_\Sigma Y$. Analogously, every substitution $s : X \longrightarrow R_\Sigma Y$ yields by the freeness of the iterative algebra $R_\Sigma X$ a unique H-algebra homomorphism $R_\Sigma X \longrightarrow R_\Sigma Y$ extending s and performing substitution of rational trees.

Notice that the fact that each $\widehat{s} : T_\Sigma X \longrightarrow T_\Sigma Y$ is an H-algebra homomorphism results in the following property of substitution of all Σ-trees: for each tree t which is not just a leaf labelled by a variable, i. e., for all elements of the left-hand coproduct component $H_\Sigma T_\Sigma X$ of $T_\Sigma X$, the result of any substitution will never be just a leaf labelled in Y, i. e., $\widehat{s}(t)$ lies in $H_\Sigma T_\Sigma Y$. Or, more concisely, non-variables are preserved by substitution.

Whereas the concept of variables and substitution is appropriately captured categorically by the concept of a monad, the idea of "non-variable" and its preservation by substitution is not. However, we will need such a concept when we speak of guarded systems of equations for an arbitrary monad below. In fact, in the setting of algebraic theories (i. e., finitary monads on Set) Calvin Elgot [E] introduced the concept of an ideal theory. In [AAMV] we proved that Elgot's ideal theories are equivalent to the following concept.

For a monad $\mathbb{S} = (S, \eta, \mu)$ over Set we can form the complements of the image $\eta_X[X]$ of X under η_X in SX, say,

$$\sigma_X : S'X \longrightarrow SX$$

for all objects X.

The monad is called *ideal* provided $\sigma : S' \longrightarrow S$ is a subfunctor of S, and the monad multiplication has a domain-codomain restriction $\mu' : S'S \longrightarrow S'$. In this subsection we present the corresponding concept for general base categories.

Assumption 6.1. For the rest of Section 6 we assume that $\mathcal{A}$ is a category with finite coproducts such that coproduct injections are monomorphic.

The last requirement that coproduct injections are monomorphic is a mere technicality and could even be totally avoided, see Section 5 of [M$_1$] or Remarks 5.8 and 5.12(1) of [AMV$_2$]. However, we obtain this slightly increased generality at the expense of having to make the following notions and results more technically involved. Hence, we decided to keep the additional requirement for the convenience of the reader and as this property is very common indeed.

Definition 6.2. By an *ideal monad* is understood a six-tuple

$$\mathbb{S} = (S, \eta, \mu, S', \sigma, \mu')$$

consisting of a monad (S, η, μ), a subfunctor $\sigma : S' \hookrightarrow S$ and a natural transformation $\mu' : S'S \longrightarrow S'$ such that

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

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

$$\begin{tikzcd} S'S \arrow[r, "\mu'"] \arrow[d, "\sigma S"'] & S' \arrow[d, "\sigma"] \\ SS \arrow[r, "\mu"'] & S \end{tikzcd}$$

commutes.

Examples 6.3.

(i) Free monads are ideal. If $\mathbb{F} = (F, \eta, \mu)$ is a free monad on the endofunctor H with the universal arrow $\kappa : H \longrightarrow F$, then the functor $HF + Id$ is a monad with unit $\widetilde{\eta} = \mathsf{inr} : Id \longrightarrow HF + Id$ and multiplication

$$\begin{tikzcd} \widetilde{\mu} \equiv\ (HF+Id)^2 = HF(HF+Id)+HF+Id \arrow[d, "HF(\kappa F+Id)+HF+Id"'] \\ HF(FF+Id)+HF+Id \arrow[d, "HF[\mu,\eta]+HF+Id"'] \\ HFF+HF+Id \arrow[d, "{[H\mu,\mathsf{inl}]+Id}"'] \\ HF+Id \end{tikzcd}$$

see Lemma 3.4 of [M$_2$]. Thus, the natural transformation $\mathsf{inl} \cdot H\eta : H \longrightarrow HF + Id$ induces a unique monad morphism $\alpha : F \longrightarrow HF + Id$ with $\alpha \cdot \kappa = \mathsf{inl} \cdot H\eta$. It is not difficult to show that the natural transformation

$$\beta \equiv\ HF + Id \xrightarrow{[\kappa F, Id]} FF + Id \xrightarrow{[\mu,\eta]} F$$

is a monad morphism and furthermore an inverse of α. In fact, use the ideas from the proof of Theorem 3.1 of [M$_2$]. Thus F is a coproduct of HF and Id via the injections $\varphi = \beta \cdot \mathsf{inl}$ and $\eta = \beta \cdot \mathsf{inr}$. Since coproduct injections are monomorphic we obtain a subfunctor $\varphi : HF \hookrightarrow F$ making F an ideal monad. In fact, it is easy to show using the associativity of μ and naturality of κ that the desired restriction of μ is $\mu' = H\mu : HFF \longrightarrow HF$. We leave the details to the interested reader.

(ii) For an iteratable endofunctor H, the free cia monad $\mathbb{T} = (T, \eta, \mu)$ together with the endofunctor HT and the natural transformation

$$\tau : HT \hookrightarrow T$$

expressing the H-algebra structure $\tau_Y : HTY \longrightarrow TY$ of each TY is ideal. The restriction of μ here is

$$\mu' = H\mu : HTT \longrightarrow HT\,.$$

In fact, we know that $TY = HTY + Y$ with the coproduct injections τ_Y and η_Y: this follows from Corollary 3.10(ii). And the following square

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

commutes because each μ_Y is a homomorphism of H-algebras.

(iii) Similarly, if H is a finitary endofunctor of an lfp category $\mathcal{A}$, then the rational monad $\mathbb{R}$ is ideal. In fact, recall from Remark 3.28 that $R = HR + Id$. We consider the subfunctor $HR \hookrightarrow R$ expressing the H-algebra structure $\varrho_Z : HRZ \longrightarrow RZ$ of each RZ. The "restriction" of μ is $\mu' = H\mu$ again.

(iv) The monad on Set given by the free algebras with a binary commutative operation is ideal. In fact, this is the free monad on the endofunctor $\mathcal{P}_2$ that assigns to every set X the set of unordered pairs from X, see Example 2.23(iii) and Theorem 2.42(i).

(v) The free semigroup monad $X \longmapsto X^+$ on Set is ideal. Here X^+ denotes the set of all non-empty words on X and $S'X \hookrightarrow X^+$ is the subset of words of length at least 2, and μ' is the obvious restriction of the concatenation of words to that subset.

(vi) The free monoid monad $X \longmapsto X^*$ on Set is not ideal. In fact, recall that the unit η_X maps elements of X to words of length 1. Now consider the word xx' in $\{ x, x' \}^*$ and the substitution s that substitutes x by itself and x' by the empty word. Then $\widehat{s}(xx') = x$ whence μ cannot have the necessary restriction.

(vii) Classical algebraic theories (groups, lattices, etc.) are usually not ideal.

(viii) Coproducts of ideal monads exist and are ideal. Assume that $\mathcal{A}$ has colimits of ω-chains and let $S = S' + Id$ and $M = M' + Id$ be ideal monads so that S' and M' preserve colimits of ω-chains. Then a coproduct of S and M in the category of monads of $\mathcal{A}$ exists and is an ideal monad, see [GUs].

Definition 6.4.

(i) An *ideal monad morphism* from an ideal monad $(S, \eta^S, \mu^S, S', \sigma, \mu'^S)$ 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 means that there exists a natural transformation $m' : S' \longrightarrow U'$ such that the square

$$\begin{tikzcd}
S' \arrow[r, "m'"] \arrow[d, "\sigma"'] & U' \arrow[d, "\omega"] \\
S \arrow[r, "m"'] & U
\end{tikzcd}$$

commutes.

(ii) Given a functor H, a natural transformation $\lambda : H \longrightarrow S$ is called *ideal* provided that it factors through $\sigma : S' \hookrightarrow S$, i. e., there exists a natural transformation $\lambda' : H \longrightarrow S'$ with $\sigma \cdot \lambda' = \lambda$.

Example 6.5. For every endofunctor H, a free monad $\mathbb{F}$, a rational monad $\mathbb{R}$ and a free cia monad $\mathbb{T}$ (if they exist) come with canonical ideal natural transformations

$$H \xrightarrow{H\eta} HF \xrightarrow{\varphi} F\,, \qquad HR \xrightarrow{H\eta} HR \xrightarrow{\varrho} R\,, \qquad HT \xrightarrow{H\eta} HT \xrightarrow{\tau} T\,. \tag{6.38}$$

For a polynomial endofunctor H_Σ of Set these transformations express the operation symbols of the signature Σ (regarded as flat Σ-trees over the set X) as terms in $F_\Sigma X$ or (rational) Σ-trees in $T_\Sigma X$ and $R_\Sigma X$, respectively.

Theorem 6.6. *A free monad $\mathbb{F}$ on H (if it exists) is a free ideal monad on $\mathcal{A}$. More precisely, for any monad $\mathbb{S}$ and any natural transformation $\lambda : H \longrightarrow S$ there exists a unique monad morphism $\overline{\lambda} : \mathbb{F} \longrightarrow \mathbb{S}$ such that $\overline{\lambda} \cdot \kappa = \lambda$. Furthermore, if $\mathbb{S}$ is an ideal monad and λ an ideal natural transformation, then $\overline{\lambda}$ is an ideal monad morphism.*

Proof. If $\mathbb{F} = (F, \eta, \mu)$ is a free monad on H we know from Example 6.3(i) that $\mathbb{F}$ is an ideal monad. We only need to show that the last part of the statement of our Theorem. So consider any ideal monad $\mathbb{S}$ and any ideal natural transformation λ. Then for the unique induced monad morphism $\overline{\lambda} : \mathbb{F} \longrightarrow \mathbb{S}$ consider the diagram

$$\begin{tikzcd}
HF \arrow[r, "\kappa F"] \arrow[d, "H\overline{\lambda}"'] \arrow[rr, bend left, "\varphi"] & FF \arrow[r, "\mu"] \arrow[dd, "\overline{\lambda}*\overline{\lambda}"'] & F \arrow[ddd, "\overline{\lambda}"] \\
HS \arrow[d, "\lambda' S"'] \arrow[dr, "\lambda S"] & & \\
S'S \arrow[r, "\sigma S"'] \arrow[d, "\mu'"'] & SS \arrow[dr, "\mu^S"] & \\
S' \arrow[rr, "\sigma"'] & & S
\end{tikzcd}$$

All its inner parts commute: the upper part is the definition of φ, see Example 6.3, the two parts below it commute since $\overline{\lambda} \cdot \kappa = \lambda$ and since $\overline{\lambda}$ is a monad morphism, the inner triangle commutes since λ is an ideal natural transformation and the lowest part commutes since $\mathbb{S}$ is an ideal monad. Thus, the left-hand edge $\mu' \cdot \lambda' S \cdot H\overline{\lambda} : HF \longrightarrow S'$ is the desired restriction of $\overline{\lambda}$. $\square$

6.2 The Universal Properties

Definition 6.7. Let $\mathbb{S} = (S, \eta, \mu, S', \sigma, \mu')$ be an ideal monad on $\mathcal{A}$.

(i) By an *equation morphism* is meant a morphism

$$e : X \longrightarrow S(X + A)$$

in $\mathcal{A}$ where X is an object ("of variables") and A is an object ("of parameters"). We call e *finitary* if X is a finitely presentable object of $\mathcal{A}$.

(ii) By a *solution* of e is meant a morphism $e^\dagger : X \longrightarrow SA$ for which the square

$$\begin{tikzcd}
X \arrow[r, "e^\dagger"] \arrow[d, "e"'] & SY \\
S(X+Y) \arrow[r, "{S[e^\dagger, \eta_Y]}"'] & SSY \arrow[u, "\mu_Y"']
\end{tikzcd}$$

commutes.

(iii) The equation morphism e is called *guarded* if it factors through the summand $S'(X + Y) + Y$ of $S(X + Y) = S'(X + Y) + X + Y$:

$$\begin{tikzcd}
X \arrow[r, "e"] \arrow[dr, dotted] & S(X+Y) \\
& S'(X+Y)+Y \arrow[u, "{[\sigma_{X+Y}, \eta_{X+Y}\mathsf{inr}]}"']
\end{tikzcd}$$

(iv) The ideal monad $\mathbb{S}$ is called *completely iterative* provided that every guarded equation morphism has a unique solution. And $\mathbb{S}$ is called *iterative* if every guarded finitary equation morphism has a unique solution.

Clearly, any completely iterative monad is also iterative. Furthermore, we have the following result:

Theorem 6.8.

(i) If H is an iteratable endofunctor, then its free cia monad $\mathbb{T}$ is completely iterative.

(ii) If H is a finitary endofunctor of the lfp category $\mathcal{A}$, then the rational monad $\mathbb{R}$ of H is iterative.

In fact, this follows easily from Theorems 5.3 and 5.6. For the full proof of (ii) see Corollary 4.7 of [AMV$_2$]. The proof of (i) is completely analogous.

Theorem 6.9. ([M$_1$], Theorem 4.3)
Let H be an iteratable endofunctor of $\mathcal{A}$. Then the monad $\mathbb{T}$ is a free completely iterative monad on H. That is, the canonical transformation $\kappa = \tau \cdot H\eta : H \longrightarrow T$, see Example 6.5, has the following universal property: for every completely iterative monad $\mathbb{S}$ and every ideal natural transformation $\lambda : H \longrightarrow S$ there exists a unique monad morphism $\overline{\lambda} : \mathbb{T} \longrightarrow \mathbb{S}$ such that the triangle

$$\begin{tikzcd} H \arrow[r, "\kappa"] \arrow[dr, "\lambda"'] & T \arrow[d, "\overline{\lambda}"] \\ & S \end{tikzcd}$$

commutes. Furthermore, this $\overline{\lambda}$ is an ideal monad morphism.

Sketch of Proof. For every object A of $\mathcal{A}$ consider SA as an H-algebra with the structure

$$HSA \xrightarrow{\lambda_{SA}} SSA \xrightarrow{\mu_A} SA\,.$$

This is a completely iterative algebra. In fact, every flat equation morphism $e : X \longrightarrow HX + SA$ yields the following equation morphism w. r. t. $\mathbb{S}$:

$$\overline{e} \equiv X \xrightarrow{\quad e \quad} HX + SA \xrightarrow{\lambda_X + SA} SX + SA \xrightarrow{\quad \mathsf{can} \quad} S(X + A)\,.$$

It is easy to verify that $\overline{e}$ is guarded, and that solutions of $\overline{e}$ w. r. t. the completely iterative monad $\mathbb{S}$ are in 1-1-correspondence with solutions of e. Thus, since $\overline{e}$ has a unique solution so does e.

Now use that (TA, τ_A) is a free cia on A to obtain a unique H-algebra homomorphism $\overline{\lambda}_A : TA \longrightarrow SA$ extending η^S_A, i. e., such that $\overline{\lambda}_A \cdot \eta_A = \eta^S_A$. One readily proves that $\overline{\lambda}$ is natural in A, that it is a monad morphism from $\mathbb{T}$ to $\mathbb{S}$, that it is uniquely determined, and that it is an ideal monad morphism. In fact, see Theorem 4.4 of [M$_1$] for the details. □

Theorem 6.9 shows that any iteratable endofunctor admits a free completely iterative monad. The converse is the main result of [M$_2$], see also [M$_1$] for an improved proof. This establishes that a free completely iterative monad on an endofunctor is precisely the monad of free cias. Notice that this result does not need any side conditions on $\mathcal{A}$ as is the case in Theorem 2.42 relating free monads and free algebras.

Theorem 6.10. ([M$_1$], Theorem 6.1)
Every endofunctor generating a free completely iterative monad is iteratable. More precisely, if H has a free completely iterative monad $\mathbb{T} = (T, \eta, \mu, T', t, \mu')$, then H is iteratable, and moreover, for any object A, TA is the final coalgebra for $H(\_) + A$.

Sketch of Proof. Given a free completely iterative monad T on H with a canonical transformation $\kappa : H \longrightarrow T$. Then we obtain a completely iterative monad $HT + Id$, see [M$_2$], Lemmas 3.3 and 3.5, with an ideal natural transformation $\mathsf{inl} \cdot H\eta : H \longrightarrow HT + Id$. Thus, we obtain a unique ideal monad morphism $\alpha : T \longrightarrow HT + Id$ such that $\alpha \cdot \kappa = \mathsf{inl} \cdot H\eta$. One readily shows that the natural transformation

$$\beta \equiv HT + Id \xrightarrow{\kappa + Id} TT + Id \xrightarrow{[\mu, \eta]} T$$

is an inverse of α. Then it is not difficult to show that TA is a final coalgebra for $H(\_) + A$. In fact, every coalgebra $c : C \longrightarrow HC + A$ yields a guarded equation morphism

$$e \equiv C \xrightarrow{\quad c \quad} HC + A \xrightarrow{\quad [\kappa_C, \eta_A] \quad} TC + TA \xrightarrow{\quad \mathsf{can} \quad} T(C + A)\,.$$

The solutions of e are in one to one correspondence with coalgebra homomorphisms from (C, c) to (TA, α_A). Since there exists a unique solution $e^\dagger$ it follows that (TA, α_A) is the desired final coalgebra. □

Theorem 6.11. ([AMV$_2$], Theorem 5.13)
Let H be a finitary endofunctor of an lfp category $\mathcal{A}$. Then the rational monad $\mathbb{R}$ is the free iterative monad on H. That is, the canonical transformation $\kappa = \varrho \cdot H\eta : H \longrightarrow R$, see Example 6.5, has the following universal property: for every iterative monad $\mathbb{S}$ and every ideal natural transformation $\lambda : H \longrightarrow S$ there exists a unique ideal monad morphism $\overline{\lambda} : \mathbb{R} \longrightarrow \mathbb{S}$ such that the triangle

$$\begin{tikzcd}
H \arrow[r, "\kappa"] \arrow[dr, "\lambda"'] & R \arrow[d, "\overline{\lambda}"] \\
& S
\end{tikzcd}$$

commutes. Furthermore, this $\overline{\lambda}$ is an ideal monad morphism.

Remark 6.12. The proof of Theorem 6.11 is analogous to the proof of Theorem 6.9. Furthermore, Theorem 6.11 has the following equivalent, and more categorical, formulation. Let $\mathcal{A}$ be an lfp category and recall that $\mathsf{Fin}[\mathcal{A}, \mathcal{A}]$ is the category of all finitary endofunctors of $\mathcal{A}$ and natural transformations between them. For the category

$$\mathsf{FIM}(\mathcal{A})$$

of all finitary iterative monads (i.e., iterative monads $(S, \eta, \mu, S', \sigma, \mu')$ with S and S' finitary) and ideal monad morphisms we have a forgetful functor

$$U : \mathsf{FIM}(\mathcal{A}) \longrightarrow \mathsf{Fin}[\mathcal{A}, \mathcal{A}], \qquad \mathbb{S} \longmapsto S'.$$

Theorem 6.11 states that U has a left adjoint, viz, the functor $H \longmapsto \mathbb{R}$.

A similar formulation of Theorem 6.9 is possible when $\mathcal{A}$ is a locally presentable category, see Remark 2.13, and only accessible functors are considered, see [M$_1$], Remark 4.4, for the precise formulation.

Remark 6.13. Observe that in case of a polynomial endofunctor H_Σ of Set the results of Theorems 6.9 and 6.11 specialize to second-order substitution of all (rational) Σ-trees. In fact, recall Example 2.44, and notice that for signatures Σ and Γ a natural transformation $\lambda : H_\Sigma \longrightarrow T_\Gamma$ essentially gives an "implementation" to each operation symbol of Σ as a Γ-tree. 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. The action of the induced monad morphism $\overline{\lambda} : T_\Sigma \longrightarrow T_\Gamma$ is to perform for every set X the second-order substitution of trees of $T_\Sigma X$ according to λ_X. The result of Theorem 6.11 now implies that if for all sets X, the image of λ_X consists of rational trees of $R_\Gamma X$, then the above monad morphism has a domain-codomain restriction to $\overline{\lambda} : R_\Sigma \longrightarrow R_\Gamma$, i. e., the result of second-order substitution according to λ applied to rational Σ-tree is a rational Γ-tree. Shortly, rational trees are closed under second-order substitution.

7 Recursive Program Schemes

Describe, using diagrams where appropriate,
the exact circumstances leading to your death.
Grant Naylor, Red Dwarf: Infinity Welcomes Careful Drivers.

In this section we present the results of our paper [MM], which is joint work with Larry Moss. Here we shall be interested in the semantics of recursive program schemes such as the one given in (1.7). Their classical theory is compactly presented in [G]. Recall that we distinguish between uninterpreted and interpreted semantics of recursive program schemes. For example, let Σ be a signature of givens consisting of a binary symbol F and a unary one G. Then (1.7) is an uninterpreted recursive program scheme and its semantics is given by the infinite trees in (1.8), which are the result of unfolding the given scheme. An interpreted scheme comes with an algebra having operations for all the givens. A basic example is the recursive program scheme (1.9) defining the factorial function. The standard way to obtain this interpreted solution is to turn the natural numbers into a continuous algebra, and to compute the least fixed point of the continuous function induced by (1.9).

In this section we shall present a generalization of the classical theory based on the results of the previous sections. The point in a nutshell is that knowing that Σ-trees (over a set X) are the final coalgebra for a polynomial functor on Set allows us to provide an uninterpreted semantics to recursive program schemes, i. e., we show how to study and solve (suitably generalized) recursive program schemes in final coalgebras. A part of the proof of our Solution Theorem 7.9 below is inspired by the work of Neil Ghani, Christoph Lüth and Federico De Marchi, see [GLM$_2$]. They have obtained a general solution theorem with the aim of providing a categorical treatment of uninterpreted recursive program scheme solutions. However, the connection we provide to (generalized) second-order substitution in Theorem 6.9 is new in our work.

Furthermore, we present an interpreted semantics of recursive program schemes. We show how to give an interpreted solution to recursive program schemes in arbitrary complete Elgot algebras as introduced in Section 4. Recall that in the classical theory a fundamental result states that uninterpreted and interpreted solutions are consistent. We explained this in the introduction, see (1.12). In our categorical treatment of recursive program schemes we can formulate this precisely, and we shall see that it follows immediately from the proofs of our solution theorems for (un)interpreted program schemes.

Finally, we present several applications of our theory. Our method for obtaining interpreted solutions easily specializes to the usual denotational semantics using complete partial orders. We also show how to solve recursive program schemes in complete metric spaces, see Examples 1.13 and 1.14 in the introduction. Furthermore, we obtain application which go beyond the possibilities of the classical theory. For example, it is possible to recursively define operations satisfying certain equations like commutativity with a recursive program scheme directly in out setting.

7.1 Iteratable Endofunctors

Assumption 7.1. For the rest of Section 7 we assume that $\mathcal{A}$ is a category with finite coproducts (having monomorphic injections). In addition all endofunctors on $\mathcal{A}$ we consider are assumed to be iteratable, see Definition 3.12.

There are fairly mild conditions. In fact, monomorphic coproduct injections could even be avoided, see our discussion after Assumption 6.1. However, we 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, see [AAMV], Example 2.12. In the concrete categories we consider here there are stronger yet much nicer conditions that ensure iteratability:

Examples 7.2.

(i) Recall from Remark 2.13 that in the category Set an endofunctor is called accessible if it preserves λ-filtered colimits for some regular cardinal λ. Every accessible endofunctor is iteratable, see [B$_3$]. In particular, functors derived from signatures on Set are iteratable. We discussed the final coalgebra $T_\Sigma X$ of $H_\Sigma(\_) + X$ in Example 2.29(vi).

(ii) Recall from Example 4.11(i) the category CPO of ω-complete partial orders and continuous functions. Unfortunately, not all locally continuous functors of CPO need be iteratable. For a counterexample consider the endofunctor assigning to a cpo X the power set of the set of order components of X. This is a locally continuous endofunctor but it does not have a final coalgebra. However, every accessible endofunctor H of CPO has a final coalgebra, see [B$_3$], and moreover, H is iteratable.

(iii) Recall the category CMS of complete metric spaces from Example 3.6(vi). Every contracting endofunctor on CMS is iteratable, see [ARe].

Notation 7.3. As we shall frequently have to deal with final coalgebras coming from two different functors we denote from now on for an endofunctor H of $\mathcal{A}$ by

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

the final coalgebra for $H(\_) + X$ (or, equivalently, a free cia on X). Whenever confusion is unlikely we will drop the parenthetical (H) and simply write TX as before. Recall from Corollary 3.13 that $\mathcal{T}(H)$ carries the structure of a monad whose unit and multiplication we denote by

$$\eta^H : Id \longrightarrow T \qquad \text{and} \qquad \mu^H : TT \longrightarrow T\,,$$

respectively. Recall furthermore, that the structures of the free cias TX yield a natural transformation

$$\tau^H : HT \longrightarrow T\,,$$

and from Example 6.5 we have the canonical natural transformation

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

We shall frequently drop the superscripts if confusion is unlikely.

Examples 7.4. We mention additional examples of iteratable endofunctors of interest.

(i) Recall from Example 3.11(iii) that every finitary endofunctor H of Set comes as a quotient $\varepsilon : H_\Sigma \longrightarrow H$ of some polynomial functor H_Σ and that its free cias TX are given as a quotient of Σ-trees over X modulo finite and infinite application of the basic equations describing ε.

(ii) The free cias of the finite power set functor $\mathcal{P}_{\text{fin}}$ are the algebras TY of strongly extensional finitely branching trees over Y, see Example 3.11(ii). There is also a different but very elegant description of TY using non-well founded sets, see [BM]. In fact, assuming the Anti-Foundation Axiom, a final $\mathcal{P}_{\text{fin}}$-coalgebra is carried by the set HF_1 of hereditarily finite sets. Similarly, TY is the set $HF_1(Y)$ of hereditarily finite sets generated from the set Y.

(iii) In our applications, the key point is that certain Set functors lift to (iteratable) endofunctors of 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, see Example 4.11(iii). Then H' is iteratable, and moreover, the final coalgebra functor $\mathcal{T}(H')$ is a lifting of $\mathcal{T}(H)$:

$$\begin{tikzcd}
\mathsf{CPO} \arrow[r, "\mathcal{T}(H')"] \arrow[d, "U"'] & \mathsf{CPO} \arrow[d, "U"] \\
\mathsf{Set} \arrow[r, "\mathcal{T}(H)"'] & \mathsf{Set}
\end{tikzcd} \tag{7.39}$$

To see this first recall that for every set X the final coalgebra $\mathcal{T}(H)X$ is obtained from the final coalgebra chain T_i of $H(\_) + X$, see Theorem 2.32. 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 bijective. 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 $j + k$, $k \leq \mathsf{card}(T_j \times T_j)$. 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 T_{j+k}, whence the square (7.39) commutes as desired.

For example, every polynomial functor H_Σ has a locally continuous and iteratable lifting H'. The 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 discretely ordered set, $+$ is the coproduct of CPO (a lift of the coproduct on Set) and $\times$ the usual product. It should be noted that even if X has a least element $\perp$, $H'X$ 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 more detailed later in Example 7.24(i).

(iv) For a set endofunctor H with a contracting lifting H' on CMS, see Example 3.6(vii), we have that H is iteratable and $U \cdot \mathcal{T}(H') = \mathcal{T}(H) \cdot U$, for the forgetful functor $U : \mathsf{CMS} \longrightarrow \mathsf{Set}$. In fact, this follows from the results of [ARe] since U preserves limits. Recall that every polynomial functor H_Σ of Set has a contracting lifting to CMS. The final coalgebra $\mathcal{T}(H')X$ is the Σ-tree algebra $T_\Sigma X$ equipped with a suitable complete metric. We will discuss this metric later in Remark 7.26.

7.2 Uninterpreted Recursive Program Schemes

The classical treatment of recursive program schemes fits into our work in the following way: Suppose we have a signature Σ of *given* operation symbols. Let Φ be a (finite) 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

$$f(x_1, \dots, x_n) \approx t^f(x_1, \dots, x_n), \qquad f \in \Phi_n, \qquad n \in \mathbb{N}\,. \tag{7.40}$$

Now observe that the names of the variables in (7.40) do not matter. More precisely, regarding Φ as a functor from $\mathbb{N}$ to Set as in Example 2.44, every RPS as in (7.40) gives rise to a natural transformation

$$\Phi \longrightarrow T_{\Sigma+\Phi} \cdot J\,, \tag{7.41}$$

where recall that $J : \mathbb{N} \longrightarrow \mathsf{Set}$ is the inclusion functor mapping a number n to the set $\{\,0, \dots, n-1\,\}$. The formulation in (7.41) insures that in each equation of an RPS such as (7.40), if the symbol on the left side is n-ary, then the variables that can appear on the right are the n elements of $\{\,0, \dots, n-1\,\}$. 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 (7.41) corresponds to a unique natural transformation

$$H_\Phi \longrightarrow T_{\Sigma+\Phi} \tag{7.42}$$

as explained in Example 2.44. The point is that the formulation in (7.42) is more useful to us than the one in (7.41) because (7.42) 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 polynomial functors for Σ and Φ, respectively. With this in mind, we can rewrite (7.42), and we see that recursive program schemes correspond to natural transformations of the following form:

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

Example 7.5. 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 (1.7). The polynomial 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 (1.7) gives a natural transformation $H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$.

Similarly, the RPS (1.9) defining the factorial function with the signature Σ of givens and the signature Φ containing the unary symbol f gives rise to a natural transformation $H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$.

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

RPS. But much has been omitted so far. We have given no general account of any solution method, or even, why the trees in (1.8) solve the scheme (1.7).

We will now 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 $V \longrightarrow \mathfrak{T}(H+V)$ where H, V, and $H+V$ are iteratable endofunctors on the category $\mathcal{A}$. To say what a solution of a recursive program scheme is we use the universal property of the monads $\mathfrak{T}(H)$ as presented in Theorem 6.9. 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 7.6. Let V and H be endofunctors on $\mathcal{A}$. A *recursive program scheme* (or *RPS*, for short) is a natural transformation

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

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

The RPS e is called *guarded* if there exists a natural transformation $f : V \longrightarrow H\mathfrak{T}(H+V)$ such that the following diagram commutes:

$$\begin{tikzcd} V \arrow[r, "e"] \arrow[dd, "f"'] & \mathfrak{T}(H+V) \\ & (H+V)\mathfrak{T}(H+V) \arrow[u, "\tau^{H+V}"'] \\ \arrow[r] & H\mathfrak{T}(H+V) \arrow[u, "\mathsf{inl}*\mathfrak{T}(H+V)"'] \end{tikzcd} \tag{7.43}$$

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

$$\begin{tikzcd} V \arrow[r, "e^\dagger"] \arrow[d, "e"'] & \mathfrak{T}(H) \\ \mathfrak{T}(H+V) \arrow[ur, "\overline{[\kappa^H, e^\dagger]}"'] & \end{tikzcd} \tag{7.44}$$

Remark 7.7. Recall that $\overline{[\kappa^H, e^\dagger]}$ is the unique ideal monad morphism extending $\sigma = [\kappa^H, e^\dagger] : H+V \longrightarrow \mathfrak{T}(H)$, see Theorem 6.9. Observe that therefore it is important to require that $e^\dagger$ be an ideal natural transformation since otherwise $\overline{\sigma}$ is not defined.

Remark 7.8.

(i) From the discussion at the beginning of this subsection it follows that 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.

Our notion of guardedness captures precisely the requirement that all right-hand sides of (7.40) 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].

Notice that the two recursive program schemes of Example 7.5 are in Greibach normal form, whence they give rise to guarded RPS in the sense of Definition 7.6.

(ii) Suppose that $H = H_\Sigma$ and $V = H_\Phi$ are polynomial endofunctors of Set, and consider the recursive program scheme $e : H_\Phi \longrightarrow \mathfrak{T}(H_\Sigma + H_\Phi)$ as a set of formal equations as in (7.40). Then for every set X of syntactic variables the X-component $e^\dagger_X : H_\Phi X \longrightarrow T_\Sigma X$ of a solution assigns to a 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 (7.44) 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 $e^\dagger_X(f, \vec{x})$.

Theorem 7.9. ([MM], Theorem 6.15)
Every guarded recursive program scheme has a unique solution.

Sketch of Proof. Let $H : \mathcal{A} \longrightarrow \mathcal{A}$ be any iteratable functor. Then $T = \mathcal{T}(H)$ is a final coalgebra for the functor $H \cdot \_ + Id$ on the endofunctor category $[\mathcal{A}, \mathcal{A}]$. This result extends to the comma-category $H/\mathsf{M}(\mathcal{A})$ whose objects are pairs $(S, \sigma : H \longrightarrow S)$ where S is a monad on $\mathcal{A}$ and σ is a natural transformation, and whose morphisms $h : (S_1, \sigma_1) \longrightarrow (S_2, \sigma_2)$ are monad morphisms $h : S_1 \longrightarrow S_2$ with $h \cdot \sigma_1 = \sigma_2$. In fact, we obtain a functor $\mathcal{H}$ on $H/\mathsf{M}(\mathcal{A})$ with $\mathcal{H}(S, \sigma) = (HS + Id, \mathsf{inl} \cdot H\eta)$, where $\eta : Id \longrightarrow S$ is the unit of the monad S. This functor restricts to the subcategory of $H/\mathsf{M}(\mathcal{A})$ formed by all pairs (S, σ), where S is a completely iterative monad and σ an ideal natural transformation, and by all ideal monad morphisms, see 6.4. Furthermore, T together with $\kappa^H : H \longrightarrow T$, see Notation 7.3, is the final $\mathcal{H}$-coalgebra.

Now suppose we are given a guarded RPS $e : V \longrightarrow \mathcal{T}(H + V)$. Then we have a natural transformation $\sigma = \mathsf{inl} \cdot [H\eta^{H+V}, f] : H + V \longrightarrow H\mathcal{T}(H+V) + Id$ which is ideal in the sense that it factors through $H\mathcal{T}(H+V)$. Notice that $H\mathcal{T}(H + V) + Id$ is obtained by applying the functor $\mathcal{H}$ to $(\mathcal{T}(H + V), \kappa^{H+V} \cdot \mathsf{inl})$. Thus it is a monad; in fact, it is a completely iterative monad. Use the universal property of the free completely iterative monad $\mathcal{T}(H + V)$, see Theorem 6.9, to obtain a monad morphism $\overline{e} : \mathcal{T}(H + V) \longrightarrow H\mathcal{T}(H + V) + Id$. It is easy to see that this gives rise to a $\mathcal{H}$-coalgebra, and so there exists a unique coalgebra homomorphism h from this coalgebra to the final one carried by $(\mathcal{T}(H), \kappa^H)$, which gives a monad morphism. Now define

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

A non-trivial proof shows that this is indeed the desired unique solution of e, see [MM] for details. □

Remark 7.10. The first part of the above proof showing the finality of $\mathcal{T}(H)$ and defining the monad morphism h uses similar ideas than the proof of the main result of [GLM$_2$]. However, the second part in which it is proved that $e^\dagger$ is a solution of e and that this solution is unique is new in our work. It connects solutions to the (generalized) second-order substitution as presented in Theorem 6.9.

Remark 7.11. 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 (finite and infinite) tree there is a recursion variable. More precisely, suppose we have signatures Σ and Φ, and an RPS as in (7.40) which is in Greibach normal form. The component of $\overline{e} : \mathcal{T}(H_\Sigma + H_\Phi) \longrightarrow H\mathcal{T}(H_\Sigma + H_\Phi)$ at some set X of syntactic variables is a flat equation morphism $\overline{e}_X : T_{\Sigma+\Phi}X \longrightarrow HT_{\Sigma+\Phi}X + X$. Therefore it can be seen as a set of formal equations, which we will now describe. For every tree $t \in T_{\Sigma+\Phi}X$ we have a variable $\underline{t}$. If t is a single node tree labelled by the syntactic variable $x \in X$ then we have the formal equation

$$\underline{t} \approx x,$$

and otherwise we have, for some $n \in \mathbb{N}$ and some $\sigma \in \Sigma_n$,

$$\underline{t} \approx \sigma(\underline{t_1}, \dots, \underline{t_n}),$$

where the tree $s = \sigma(t_1, \dots, t_n)$ 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].

Now the component at X of the induced natural transformation $h : \mathcal{T}(H_\Sigma + H_\Phi) \longrightarrow \mathcal{T}(H_\Sigma)$ assigns to every $\underline{t} \in T_{\Sigma+\Phi}X$ of the flat system given by $\overline{e}_X$ its solution in the cia $T_\Sigma X$, i.e., the Σ-tree obtained by unfolding the recursive definition of $\underline{t}$ in the flat system. Thus, to every element $(f, \vec{x})$ of $H_\Phi X$ the component of the uninterpreted solution $e^\dagger$ at X assigns the tree unfolding of $\underline{(f, \vec{x})}$ according to the system given by $\overline{e}_X$.

Examples 7.12.

(i) For the guarded RPS of (1.7) the flat system obtained from $\overline{e}_X$ for $X = \{ x \}$ includes the equations of the system

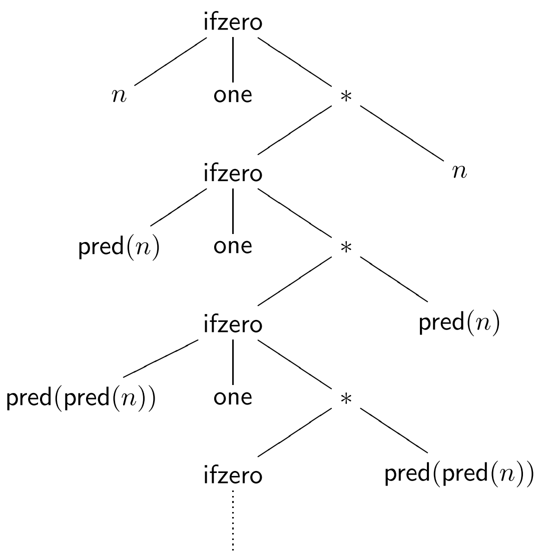
$$\begin{array}{rcl@{\qquad}rcl}
\underline{x} & \approx & x & \underline{\varphi(Gx)} & \approx & F(\underline{Gx}, \underline{\varphi(GGx)}) \\
\underline{\varphi(x)} & \approx & F(\underline{x}, \underline{\varphi(Gx)}) & \underline{F(x, \varphi(Gx))} & \approx & F(\underline{x}, \underline{F(Gx,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

It is clear that the solution of this system in the cia $T_\Sigma X$ will assign to $\underline{\varphi(x)}$ and $\underline{\psi(x)}$ the trees of (1.8).

(ii) The recursive program scheme of (1.9) yields for $X = \{ n \}$ the extension $\overline{e}$ which at X gives a system including the following formal equations:

$$\begin{aligned}
\underline{f(n)} &\approx \mathsf{ifzero}(\underline{n}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(n)) * n}) \\
\underline{n} &\approx n \\
\underline{\mathsf{one}} &\approx \mathsf{one} \\
\underline{f(\mathsf{pred}(n)) * n} &\approx \underline{\mathsf{ifzero}(\mathsf{pred}(n), \mathsf{one}, f(\mathsf{pred}(\mathsf{pred}(n))))} * \underline{n} \\
\underline{f(\mathsf{pred}(n))} &\approx \mathsf{ifzero}(\underline{\mathsf{pred}(n)}, \underline{\mathsf{one}}, \underline{f(\mathsf{pred}(\mathsf{pred}(n)) * \mathsf{pred}(n))}) \\
&\vdots
\end{aligned} \tag{7.45}$$

The map h_X assigns to $\underline{f(n)}$ the following infinite tree



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

Example 7.13. Let us now present an example of RPSs which are *not* captured in the classical setting. 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 defining recursively 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. Then Σ 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 we want to define a commutative binary operation φ by the formal equation

$$\varphi(x, y) \approx F(x, y, \varphi(Gx, Gy)) \,. \tag{7.46}$$

To do it we express φ by the endofunctor V assigning to a set X the set of unordered pairs of X. It is not difficult to see that the formal equation (7.46) 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 Example 3.11(iii). Notice that in the classical setting we are unable to demand that (the solution of) φ is a commutative operation at this stage: this fact would be proved separately once a solution has been obtained. Here we have encoded this additional requirement into our RPS—every solution will be commutative. In fact, the components of the uninterpreted solution $e^\dagger_X : VX \longrightarrow \mathcal{T}(H)X$ assign to an unordered pair $\{x, y\}$ in VX the tree

$$\begin{tikzcd}[column sep=small, row sep=small]
 & F \arrow[dl, dash] \arrow[dr, dash] & & & \\
\{x,y\} & & F \arrow[dl, dash] \arrow[dr, dash] & & \\
 & \{Gx,Gy\} & & F \arrow[dl, dash] \arrow[dr, dash, dotted] & \\
 & & \{GGx,GGy\} & & {}
\end{tikzcd}$$

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.

7.3 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 RPS 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 equations in A.

Of course, in general an algebra A will not admit interpreted solutions. We shall show in this section that any complete Elgot algebra $(A, a, (\_)^*)$ admits an interpreted solution of every guarded recursive program scheme. Moreover, if A is a cia, there is a unique interpreted solution. We also show that uninterpreted solutions and interpreted ones are consistent as explained informally in (1.12). This is a fundamental result for algebraic semantics.

After having presented our main results we turn to applications. In Subsection 7.3.1 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 7.3.2 for the category 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.

Notation 7.14. Recall from Theorem 4.17 that for every iteratable functor H the category of complete Elgot algebras for H and their homomorphisms is isomorphic to the category of Eilenberg-Moore algebras for $\mathcal{T}(H)$. Analogously as in Remark 3.15, we denote for any complete Elgot algebra $(A, a, (-)^*)$ by

$$\widetilde{a} : TA \longrightarrow A$$

its associated structure of an Eilenberg-Moore algebra for $T = \mathcal{T}(H)$, and we shall refer to $\widetilde{a}$ as *evaluation morphism* of the complete Elgot algebra $(A, a, (-)^*)$.

Definition 7.15. Let $(A, a, (-)^*)$ be a complete Elgot algebra w.r.t. 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 the $(H + V)$-algebra $[a, e^\ddagger_A] : (H + V)A \longrightarrow A$ is a complete Elgot algebra and such that the triangle

$$\begin{tikzcd}
VA \arrow[r, "e^\ddagger_A"] \arrow[d, "e_A"'] & A \\
\mathcal{T}(H+V)A \arrow[ur, "\widetilde{[a,e^\ddagger_A]}"'] &
\end{tikzcd} \tag{7.47}$$

commutes, where the diagonal arrow denotes the evaluation morphism of the complete Elgot algebra $[a, e^\ddagger_A]$.

Remark 7.16.

(i) In our leading example where $H = H_\Sigma$ and $V = H_\Phi$ are polynomial endofunctors of Set the commutativity of (7.47) 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 (7.40). The interpreted solution $e_A^\ddagger$ gives for each n-ary operation symbol f of the signature Φ an operation $f_A : A^n \longrightarrow A$, and the commutativity of (7.47) 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 $e_A^\ddagger$ for Φ on A—this is the action of $\widetilde{[a, e_A^\ddagger]}$. The resulting element of A is the same as $f_A(\vec{a})$.

(ii) The requirement that $[a, e_A^\ddagger]$ be the structure morphism of a complete Elgot algebra may seem odd at first. However, we need to assume this in order to be able to use $\widetilde{[a, e_A^\ddagger]}$ in (7.47). 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 polynomial functors of Set as in (i) above the Elgot algebra with structure map $[a, e_A^\ddagger]$ 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 7.17. ([MM], Theorem 7.3)
Let $(A, a, (-)^*)$ be a complete 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 $e_A^\ddagger$ of e in A,

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

Sketch of Proof. Recall the $\mathcal{H}$-coalgebra structure $\overline{e}$ from the proof of Theorem 7.9. The component at A of $\overline{e}$ yields a flat equation morphism

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

w.r.t. the given complete Elgot algebra. Denote its solution $(\overline{e}_A)^*$ by β, and define

$$e_A^\ddagger \equiv VA \xrightarrow{\mathsf{inr}} (H + V)A \xrightarrow{\kappa_A^{H+V}} \mathcal{T}(H + V)A \xrightarrow{\beta} A\,. \tag{7.48}$$

A non-trivial proof shows that this morphism is an interpreted solution of e, and that it is the unique interpreted solution if A is a cia. □

Definition 7.18. For any guarded RPS e and any complete Elgot algebra $(A, a, (-)^*)$, let $e_A^\ddagger$ be the interpreted solution obtained from (7.48) above. We call this the *standard interpreted solution* of e in A.

Finally, we state the "Fundamental Theorem of Algebraic Semantics", which establishes that uninterpreted and interpreted solutions are consistent, see (1.12).

Theorem 7.19. ([MM], Theorem 7.7)
Let $(A, a, (\_)^*)$ be a complete Elgot algebra and consider its evaluation morphism $\widetilde{a} : \mathcal{T}(H)A \longrightarrow A$. Let e be any guarded recursive program scheme, let $e_A^\ddagger : VA \longrightarrow A$ be the standard interpreted solution of e in A of Theorem 7.17, and let $e^\dagger : V \longrightarrow \mathcal{T}(H)$ be the (uninterpreted) solution of Theorem 7.9. Then the following triangle commutes:

$$\begin{tikzcd} VA \arrow[r, "e_A^\dagger"] \arrow[dr, "e_A^\ddagger"'] & \mathcal{T}(H)A \arrow[d, "\widetilde{a}"] \\ & A \end{tikzcd} \tag{7.49}$$

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

Remark. Notice that $(e^\dagger)_A$ is the component at A of the natural transformation $e^\dagger$ whereas $e^\ddagger_A$ is not a component of some natural transformation but merely a morphism from VA to A. Also observe that the commuting triangle (7.49) above gives a precise meaning to the informal triangle of (1.12).

Sketch of Proof. Recall the morphisms $h : \mathcal{T}(H+V) \longrightarrow \mathcal{T}(H)$ and $\beta : \mathcal{T}(H+V)A \longrightarrow A$ from the proofs of Theorems 7.9 and 7.17, respectively. It is not difficult to show that the equation $\beta = \widetilde{a}\cdot h_A$ holds. Precompose both sides with $\kappa^{H+V}_A \cdot \mathsf{inr} : VA \longrightarrow \mathcal{T}(H+V)A$ to obtain the desired result.

The uniqueness of $e^\ddagger_A$ follows since neither $e^\dagger_A$ nor $\widetilde{a}$ depend on $e^\ddagger_A$. □

7.3.1 Interpreted Solutions in CPO

We shall show in this subsection that if we have $\mathcal{A} = \mathsf{CPO}$, the category of complete partial orders from Example 4.11(ii), as our base category, then interpreted solutions of guarded RPSs e in a continuous algebra are given as least fixed points of a continuous function on a function space. In this way we recover the usual denotational semantics from our categorical interpreted semantics of recursive program schemes.

Example 7.20. Recall the RPS (1.9), see also Example 7.5. Clearly, the intended interpreted solution is the factorial function on the natural numbers $\mathbb{N}$.

To obtain a suitable complete Elgot algebra in which we can find an interpreted solution of (1.9), we turn the natural numbers into an object of CPO. Let $\mathbb{N}_\perp$ be the flat CPO obtained from the discretely ordered $\mathbb{N}$ by adding a bottom element $\perp$, i. e., $x \le y$ if and only if $x = \perp$ or $x = y$. We equip $\mathbb{N}_\perp$ with the obvious strict operations $\mathsf{one}_{\mathbb{N}_\perp}$, $\mathsf{pred}_{\mathbb{N}_\perp}$ and $*_{\mathbb{N}_\perp}$, which are all continuous. In addition, we use the continuous operation

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

Hence we have a continuous Σ-algebra with $\perp$, and therefore $\mathbb{N}_\perp$ is an Elgot algebra w.r.t. $H_\Sigma : \mathsf{Set} \longrightarrow \mathsf{Set}$, see Example 4.11(iii).

The 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 (1.9) itself as a continuous function R on $\mathsf{CPO}(\mathbb{N}_\perp, \mathbb{N}_\perp)$ acting as

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

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 (1.9) 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 interpreted solution $e^\ddagger_{\mathbb{N}_\perp}$ obtained from Theorem 7.17. We shall do this shortly in greater generality.

In general, any recursive program scheme can be turned into a continuous function R on the function space $\mathsf{CPO}(VA, A)$. Theorem 7.21 below shows that the least fixed point of R is the same as the interpreted solution obtained from Theorem 7.17.

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 $\perp$. By Example 4.11(ii), we know that this carries the structure of a complete Elgot algebra $(A, a, (-)^*)$, where $(-)^*$ assigns to every flat equation morphism its least solution. As before we will use the notation $\widetilde{a} : \mathcal{T}(H)A \longrightarrow A$ for the evaluation morphism. Furthermore, observe that for every continuous map $f : VA \longrightarrow A$ we have a complete Elgot algebra on A with structure $[a, f] : (H+V)A \longrightarrow A$. Due to Remark 4.18(i), its evaluation morphism satisfies

$$\widetilde{[a,f]} \cdot \kappa^{H+V}_A = [a, f]\,. \tag{7.50}$$

Theorem 7.21. ([MM], Theorem 7.10)
The following function R on $\mathsf{CPO}(VA, A)$

$$f \longmapsto VA \xrightarrow{e_A} \mathcal{T}(H+V)A \xrightarrow{\widetilde{[a,f]}} A \tag{7.51}$$

is continuous. Its least fixed point is the standard interpreted solution $e^\ddagger_A : VA \longrightarrow A$ of Theorem 7.17.

Sketch of Proof. To see the continuity of R is suffices to prove that $\widetilde{(\_)} : \mathsf{CPO}(HA, A) \longrightarrow \mathsf{CPO}(\mathcal{T}(H)A, A)$ is continuous. Let us write T for $\mathcal{T}(H)$. Recall from Remark 4.18 that for every 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 $F(a, -) : \mathsf{CPO}(TA, A) \longrightarrow \mathsf{CPO}(TA, A)$ with $F(a, f) = [a, A] \cdot (Hf + A) \cdot \alpha_A$. 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 $\widetilde{(\_)}$.

We prove that $e^\ddagger_A$ is the least fixed point of R. Notice that the least fixed point of R is the join t of the increasing chain in $\mathsf{CPO}(VA, A)$ given by $t_0 = \mathsf{const}_\perp$ and $t_{i+1} = \widetilde{[a, t_i]} \cdot e_A$, for $i \in \mathbb{N}$.

Furthermore, recall that the interpreted solution $e^\ddagger_A$ is defined by $\beta \cdot \kappa^{H+V}_A \cdot \mathsf{inr}$, where $\beta = (\overline{e}_A)^*$ is the least solution of the flat equation morphism which is obtained from the component at A of the $\mathcal{H}$-coalgebra $\overline{e}$, see Theorem 7.17. By Example 4.11(ii), the solution β of $\overline{e}_A$ is the join of the chain given by $\beta_0 = \mathsf{const}_\perp$ and $\beta_{i+1} = [a, A] \cdot H(\beta_i + A) \cdot \overline{e}_A$, for $i \in \mathbb{N}$.

Observe that $e^\ddagger_A$ is a fixed point of R, see (7.47). Thus, we have $t \sqsubseteq e^\ddagger_A$. To show the reverse inequality one proves by induction on i the inequalities $\beta_i \sqsubseteq \widetilde{[a, t]}$, for $i \in \mathbb{N}$, see [MM]. This implies that $\beta \sqsubseteq \widetilde{[a, t]}$ and therefore $e^\ddagger_A = \beta \cdot \kappa^{H+V}_A \cdot \mathsf{inr} \sqsubseteq \widetilde{[a, t]} \cdot \kappa^{H+V}_A \cdot \mathsf{inr} = t$, by (7.50) above. □

Remark 7.22. The result of Theorem 7.21 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

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

where $\perp = \mathsf{const}_\perp$ is the least element of $\mathsf{CPO}(VA, A)$. Thus, with $e^\ddagger_0 = \mathsf{const}_\perp$ 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 7.23. 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

$$\begin{tikzcd}
\mathsf{CPO} \arrow[r, "\mathcal{T}(H'+V')"] \arrow[d, "U"'] & \mathsf{CPO} \arrow[d, "U"] \\
\mathsf{Set} \arrow[r, "\mathcal{T}(H+V)"'] & \mathsf{Set}
\end{tikzcd}$$

by Example 7.4(iii). 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 a complete Elgot algebra, see Example 4.11(iii). Then we can apply Theorem 7.21 to obtain an interpreted solution $e^\ddagger_A$ of e in the algebra A as a least fixed point of the above function R of (7.51).

Example 7.24.

(i) Suppose we have signatures Σ and Φ. Then the polynomial 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 a cpo X the algebra $T_{\Sigma+\Phi}X$ with the cpo structure induced by X, see Example 7.4(iii). 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 $t \sqsubseteq s$ holds in $T_{\Sigma+\Phi}X$ if and only if 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 (7.40) 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 every 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.

Let (A, a) be a CPO-enrichable H_Σ-algebra; i. e., a continuous Σ-algebra with a least element $\perp$. We wish to consider the continuous function R on $\mathsf{CPO}(H_\Phi A, A)$ which assigns to a continuous algebra structure $\varphi : H_\Phi A \longrightarrow A$ the algebra structure $R(\varphi) = \widetilde{[a, \varphi]} \cdot e'_A$. The structure $R(\varphi)$ 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 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 7.21 states that an 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 interpreted solution $e^\ddagger_A$ gives the usual denotational semantics.

(ii) Apply the previous example to the RPS of (1.9) considered as a natural transformation, see Example 7.5. Then Theorem 7.21 states that the interpreted solution of the RPS (1.9) in the complete Elgot algebra $\mathbb{N}_\perp$ is obtained as the least fixed point of the function R of Example 7.20. That is, the interpreted solution gives the desired factorial function.

(iii) Recall the guarded RPS e from Example 7.13. Consider again the algebra $\mathbb{N}_\perp$ together with the following two operations:

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

Since the first operation obviously satisfies $F_{\mathbb{N}_\perp}(x, y, z) = F_{\mathbb{N}_\perp}(y, x, z)$ we have defined an H-algebra. It is not difficult to check that the set functor H has a locally continuous lifting H' on CPO and that $\mathbb{N}_\perp$ 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: $\{x, y\} \sqsubseteq \{x', y'\}$ if and only if either $x \leq x'$ and $y \leq y'$, or $x \leq y'$ and $y \leq x$. Thus, we have defined a complete Elgot algebra for $H : \mathsf{Set} \longrightarrow \mathsf{Set}$, see Example 4.11(iii). The interpreted solution $e^\ddagger_{\mathbb{N}_\perp} : V\mathbb{N}_\perp \longrightarrow \mathbb{N}_\perp$ is given by one commutative binary operation $\varphi_{\mathbb{N}_\perp}$ on $\mathbb{N}_\perp$. We leave it to the reader to verify that for natural numbers x and y, $\varphi_{\mathbb{N}_\perp}(x, y)$ is the natural number represented by the greatest common prefix in the binary representation of x and y, e. g., $\varphi_{\mathbb{N}_\perp}(12, 13) = 6$. Notice that we do not have to prove separately that $\varphi_{\mathbb{N}_\perp}$ is commutative. The way we have formed the RPS e in Example 7.13 ensures that the interpreted solution must be given by a commutative operation.

(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 polynomial set 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 a complete Elgot algebra. Turn the formal equations (1.7) into a recursive program scheme $e : H_\Phi \longrightarrow \mathcal{T}(H_\Sigma + H_\Phi)$ as demonstrated in Example 7.5. The RPS e has a lifting $e' : V' \longrightarrow \mathcal{T}(H' + V')$, where V' denotes the lifting of H_Φ. The 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 $\varphi_A(\perp)$ is the least fixed point of f.

7.3.2 Interpreted Solutions in CMS

Recall the category CMS of complete metric spaces from Example 3.6(vi), and let $H, V : \mathsf{CMS} \longrightarrow \mathsf{CMS}$ be contracting endofunctors. We shall show in this subsection that for every guarded RPS $e : V \longrightarrow \mathcal{T}(H + V)$ we can find a unique interpreted solution in every 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 every non-expanding map $f : VA \longrightarrow A$ we obtain an algebra structure $[a, f] : (H + V)A \longrightarrow A$, thus we have the induced evaluation morphism

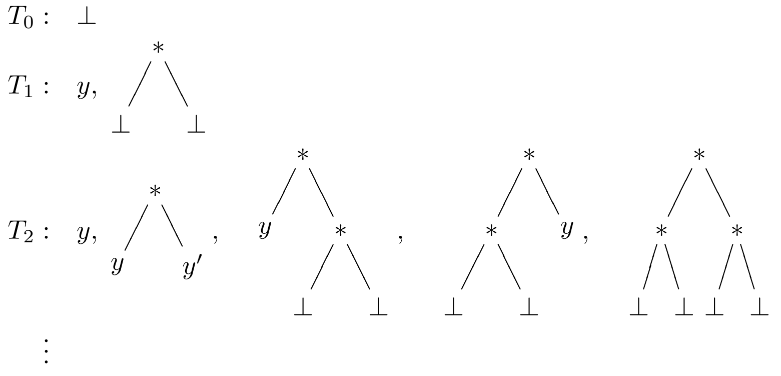
$$\widetilde{[a, f]} : \mathcal{T}(H + V)A \longrightarrow A\,.$$

As in CPO, the RPS e induces a function R on $\mathsf{CMS}(VA, A)$, see (7.51). 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 7.17. 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 7.25. ([MM], Corollary 7.14)
The interpreted solution $e^\ddagger_A : VA \longrightarrow A$ of e in A as obtained in Theorem 7.17 is the unique fixed point of the function R on $\mathsf{CMS}(VA, A)$ defined by (7.51).

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 7.17. □

Remark 7.26. Let H_Σ be a polynomial functor on Set and denote by H' a contracting lifting to CMS as described in Example 3.6(vii). 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 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 Section 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 $\perp$ 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 $*$:


T_0 : ⊥
T_1 : y, * ⊥ ⊥
T_2 : y, * y y' , * y * ⊥ ⊥ , * * y ⊥ ⊥ , * * * ⊥ ⊥ ⊥ ⊥
⋮

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 $t = \perp * \perp$, 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.

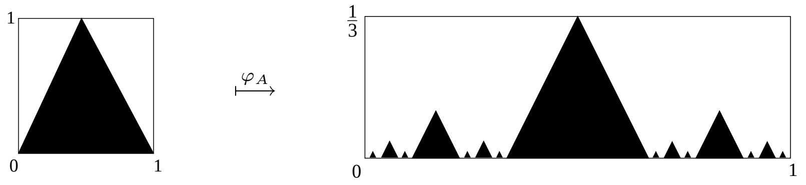
Example 7.27.

(i) Consider the endofunctor $H' : \mathsf{CMS} \longrightarrow \mathsf{CMS}$ obtained by lifting the polynomial set functor $H_\Sigma X = X \times X + X$ expressing a binary operation F and a unary one G as described in Example 3.6(vii). The Euclidean interval $I = [0, 1]$ together with the operations $F(x, y) = \frac{x+y}{4}$ and $G(x) = \frac{1}{2}\sin(x)$ is an H'-algebra, whence a cia. Use only the first equation in (1.7) to obtain a guarded RPS $e : \mathit{Id} \longrightarrow \mathcal{T}(H_\Sigma + \mathit{Id})$ 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 (1.13).

(ii) Self similar sets are solutions of interpreted program schemes. Recall from Example 3.6(viii) that for every complete metric space (X, d) we obtain the complete metric space $(C(X), h)$ of all non-empty compact subspaces of X. 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 polynomial 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 an $\frac{1}{3}$-contracting map, whence a structure of an H'-algebra. The following formal equation

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

gives rise to a guarded RPS $e : \mathit{Id} \longrightarrow \mathcal{T}(H_\Sigma + \mathit{Id})$, 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 3.6(x) let us consider $(C(I), \alpha_I)$, where $I = [0, 1]$ is the Euclidean interval 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 3.6(viii). The formal equation

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

gives similarly as in (i) above a guarded RPS $e : \mathit{Id} \longrightarrow \mathcal{T}(H_\Sigma + \mathit{Id})$, 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, cf. (1.14).

8 Conclusions and Further Research

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened.
Douglas Adams, *The Restaurant at the end of the Universe.*

In this summary we have presented the material of our papers [AMV$_1$, AMV$_2$, AMV$_3$, AAMV, M$_1$, M$_2$, MM]. We have started with the study of iterative and completely iterative algebras for an endofunctor, and we have shown that the free (completely) iterative algebras yield an easy description of free (completely) iterative monads. In this way we have generalized and extended the classical results of Calvin Elgot and his coauthors [E, BE, EBT] following the footsteps of Evelyn Nelson [N]. We have seen that free completely iterative algebras or free completely iterative monads are equivalently given by final coalgebras. We believe that this characterization of final coalgebras as free algebras of some sort is a new and interesting result. In particular, for our subsequent work on semantics of recursive program schemes that connection between coalgebra and algebra is a corner stone—it opens up the possibility to apply coalgebraic methods in algebraic semantics.

In the second part of our work we have introduced and studied (complete) Elgot algebras and we characterized them as the Eilenberg-Moore algebras for free (completely) iterative monads. We then argued that the structure of Elgot algebras captures the most important structural properties of algebras that are suitable for algebraic semantics.

In the last part we have shown how to apply our results to provide semantics of recursive program schemes in a category theoretic way. We have seen that the universal property of free completely iterative monads serves as a generalized second-order substitution and it therefore allows to formulate in a category theoretic way the notion of a solution of a recursive program scheme. We then proved that every guarded recursive program scheme has a unique uninterpreted solution. Next we provided a canonical interpreted semantics of guarded recursive program schemes in complete Elgot algebras.

As the semantics of recursive program schemes is a topic at the heart of theoretical computer science it is an important problem to see whether it can be handled using coalgebraic methods. We are pleased that we can report a success in this matter. Our applications show that we have developed a unified view of solution principles for a large class of recursive definitions including the usual algebraic semantics of recursive program schemes. Of course, our more abstract theory takes somewhat more effort to build. But we feel that the gain in conceptual clarity more than outweighs this small disadvantage.

In addition to the material we have presented here we have obtained a number of results which are closely related but cannot be treated in detail here. We will discuss them briefly now.

Our whole work rests on the assumption that enough final coalgebras for an endofunctor exist. While this is a weak assumption, there are important examples of endofunctors not satisfying this requirement, e. g., the power set functor on Set. In our paper [AMV$_4$] we have shown that this can be circumvented by working in the category Class of classes in lieu of Set. We proved that every endofunctor of Class has an initial algebra and a final coalgebra so that all endofunctors generate free completely iterative monads. In particular, we gave a description of the final coalgebra for the power set functor. The fact that every endofunctor of Class has a final coalgebra was also independently discovered by Daniela Cancila [Ca]. In [AMV$_5$] we have extended our work from Set to other categories $\mathcal{K}$. More precisely, we start with any locally small, cocomplete and cowell-powered category $\mathcal{K}$, and we consider its free cocompletion $\mathcal{K}^\infty$ under transfinite colimits, e. g., for $\mathcal{K} = \mathsf{Set}$ we have $\mathcal{K}^\infty = \mathsf{Class}$. We proved that every endofunctor H of $\mathcal{K}$ extends essentially uniquely to an endofunctor H^∞ of $\mathcal{K}^\infty$, and that H^∞ has a final coalgebra. Moreover, if $\mathcal{K}$ is a locally finitely presentable category then H^∞ generates a completely iterative monad in such a way that every guarded equation morphism which lies in the base category $\mathcal{K}$ has its solution in $\mathcal{K}$, too. For example, for the power set functor that means that every guarded equation morphism can be uniquely solved in Set—one does not need to worry about classes at all.

Another line of our current research has been inspired by the work of Tarmo Uustalu [U]. He proposed to study complete iterativity with respect to a so-called base in lieu of an endofunctor. A base is a functor from the category $\mathcal{A}$ to the category of (finitary) monads on $\mathcal{A}$. Bases allow an interesting extension of our theory which captures algebras satisfying certain equations and where the iterativity can be restricted. In fact, we have taken up the task to extend the results on iterative algebras and iterative monads of [AMV$_1$, AMV$_2$] presented here to bases with the series of papers [AMV$_6$, AMV$_7$, AMV$_8$, AMV$_9$, AMV$_{10}$]. In [AMV$_{11}$] we present one application of this extended theory; a categorical description of the monad of algebraic trees,

i. e., those trees that arise as uninterpreted solutions of (classical) recursive program schemes, see [C].

Very recently, we have turned our attention to recursive coalgebras, inspired by the work of Venanzio Capretta, Tarmo Uustalu and Varmo Vene [CUV]. A coalgebra is recursive if it admits a unique coalgebra-to-algebra homomorphism into any given algebra. In the paper [ALM] which arose from the second author's diploma thesis, we have proved that for a finitary endofunctor of Set preserving inverse images a coalgebra is recursive if and only if it admits a coalgebra homomorphism into the initial algebra, or if and only if it satisfies the dual concept of complete iterativity. Applications of recursive coalgebras lie, for example, in the realm of the semantics of functional programs using divide-and-conquer strategies such as Quicksort etc. In the subsequent work [AM$_2$] we will extend the above equivalences beyond finitary functors: the same results hold for all endofunctors of Set preserving inverse images, and moreover, the category Set can be generalized substantially.

Now let us mention a few points which seem worthwhile to be addressed in future research. Our work on recursive program scheme semantics is only in its beginning phase. We suspect that much more can be said about the relation of our work to operational semantics. One should investigate higher-order recursive program schemes using our tools. The paper [MU] addresses variable binding and infinite λ-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. 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.

One of our applications of recursive program schemes showed how to define operations satisfying basic equations, e. g., commutativity. One would like tools to consider more general equations, e. g., associativity. This can however not be achieved by using endofunctors and natural transformations to express recursive program schemes as we have done. One should start with monads expressing givens and variables, e. g., a binary associative operation is expressed by the list monad on Set. But at the moment we are even lacking the most basic tools to study recursive program schemes this way. For example, we need an analogue of a free completely iterative monad on an endofunctor, i. e., we need the completely iterative reflection of a given monad. We have taken first steps in this direction with [M$_3$].

Another topic for future research is the investigation of the connection of our work to other approaches to semantics, e. g., the traced monoidal categories of [JSV, Ha] or the iteration theories of Stephen Bloom and Zoltán Ésik [BÉ]. For example, the assignment of a free iterative monad to a finitary set endofunctor gives rise to a monad on $\mathsf{Fin}[\mathsf{Set},\mathsf{Set}]$. We have characterized the Eilenberg-Moore algebras of this monad in [M$_4$]. We call those Elgot monads, analogously to Elgot algebras. They are monads providing for any guarded equation morphism a canonical solution, i. e., such that certain axioms of the solution operation are fulfilled. These axioms are very similar to the iteration theory axioms. But the precise relationship of Elgot monads and iteration theories needs still to be investigated.

Finally, to conclude this thesis let us sum up the gist of our work in one sentence: We believe to have contributed to the part of the theory of coalgebras which pertains to the semantics of recursion by exploring and using the structure of monads that arises from final coalgebras—in other words, we studied the connection of coalgebras, monads and semantics. Of course, much remains to be done, or as Albert Einstein put it:

The important thing is not to stop questioning.

References

[Ac] Peter Aczel, *Non-Well-Founded Sets*, CSLI Lecture Notes 14, Stanford University, 1988.

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

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

[A_1] Jiří Adámek, Free Algebras and Automata Realization in the Language of Categories, *Comment. Math. Univ. Carolin.* 15 (1974), 589–602.

[A_2] Jiří Adámek, Introduction the Coalgebra, *Theory Appl. Categ.* 14 (2005), 157–199.

[AK_1] Jiří Adámek and Václav Koubek, Functorial Algebras and Automata, *Kybernetika* 13 (1977), no. 4, 245–260.

[AK_2] Jiří Adámek and Václav Koubek, Are Colimits of Algebras Simple to Construct?, *J. Algebra* 66 (1980), no. 1, 226–250.

[AK_3] Jiří Adámek and Václav Koubek, On the greatest fixed point of a set functor, *Theoret. Comput. Sci.* 150 (1995), 57–75.

[ALM] Jiří Adámek, Dominik Lücke and Stefan Milius, Recursive Coalgebras of Finitary Functors, *submitted*, available via `http://www.iti.cs.tu-bs.de/~milius`.

[AM_1] Jiří Adámek and Stefan Milius, Terminal Coalgebras and Free Iterative Theories, accepted for publication in *Inform. and Comput.*, available via `http://www.iti.cs.tu-bs.de/~milius`.

[AM_2] Jiří Adámek and Stefan Milius, Recursive Coalgebras, *preprint* (2006).

[AMV_1] Jiří Adámek, Stefan Milius and Jiří Velebil, Free Iterative Theories: a coalgebraic view, *Math. Stuctures Comput. Sci.* 13 (2003), no. 2, 259–320.

[AMV_2] Jiří Adámek, Stefan Milius and Jiří Velebil, From Iterative Algebras to Iterative Theories, *submitted*, available via `http://www.iti.cs.tu-bs.de/~milius`, extended abstract appeared in *Electron. Notes Theor. Comput. Sci.* 106 (2004), 3–24.

[AMV_3] Jiří Adámek, Stefan Milius and Jiří Velebil, Elgot Algebras, *preprint* (2005), available via `http://www.iti.cs.tu-bs.de/~milius`, extended abstract to appear in *Electron. Notes Theor. Comput. Sci.*

[AMV_4] Jiří Adámek, Stefan Milius and Jiří Velebil, On Coalgebras based on Classes, *Theoret. Comput. Sci.* 316 (2004), 3–23.

[AMV_5] Jiří Adámek, Stefan Milius and Jiří Velebil, A General Final Coalgebra Theorem, *Math. Structures Comput. Sci.* 15 (2005), no. 3, 409–432.

[AMV_6] Jiří Adámek, Stefan Milius and Jiří Velebil, Algebras with Parameterized Iterativity, *submitted*, available via `http://www.iti.cs.tu-bs.de/~milius`.

[AMV_7] Jiří Adámek, Stefan Milius and Jiří Velebil, Bases for Parameterized Iterativity, *submitted*, available via `http://www.iti.cs.tu-bs.de/~milius`.

[AMV_8] Jiří Adámek, Stefan Milius and Jiří Velebil, Iterative Algebras for a Base, *Electron. Notes Theor. Comput. Sci.* 122 (2005), 147–170.

[AMV_9] Jiří Adámek, Stefan Milius and Jiří Velebil, Base Modules for Parameterized Iterativity, *preprint* (2005).

[AMV_{10}] Jiří Adámek, Stefan Milius and Jiří Velebil, Description of Bases for Parameterized Iterativity, *preprint* (2005).

[AMV_{11}] Jiří Adámek, Stefan Milius and Jiří Velebil, Algebraic Trees Coalgebraically, *manuscript* (2005).

[AP] Jiří Adámek and Hans-Eberhardt Porst, On tree coalgebras and coalgebra presentations, *Theoret. Comput. Sci.* 311 (2004), 257–283.

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

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

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

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

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

[Ban] Stefan Banach, Sur les opèrations dans les ensembles abstraits et leurs applications aux èquations intègrales, *Fund. Math.* 3 (1922), 133–181.

[Ba] Michael F. Barnsley, *Fractals everywhere*, Academic Press 1988.

[B_1] Michael Barr, Coequalizers and free triples, *Math. Z.* 116, 307–322.

[B_2] Michael Barr, Relational algebras, *Lecture Notes in Math.* 137, Springer Verlag, 39–55.

[B_3] Michael Barr, Terminal coalgebras in well-founded set theory, *Theoret. Comput. Sci.* 114 (1993), 299–315.

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

[Bl] Stephen 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] Stephen L. Bloom and Calvin C. Elgot, The Existence and Construction of Free Iterative Theories, *J. Comput. System Sci.* 12 (1974), 305–318.

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

[CUV] Venanzio Capretta, Tarmo Uustalu, Varmo Vene, Recursive Coalgebras from Comonads, *Electron. Notes Theor. Comput. Sci.* 106 (2004), 43–61.

[Ca] Daniela Cancila, Investigations in the Categorical Foundations and Applications of Coalgebras and Hypersets, PhD thesis, University of Udine, 2003.

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

[Ei] Samuel Eilenberg, The category $\mathcal{C}$. Unpublished manuscript.

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

[GU] Peter Gabriel and Friedrich Ulmer, *Lokal präsentierbare Kategorien*, Lecture N. Math. 221, Springer-Verlag, Berlin 1971.

[GLM_1] Neil Ghani, Christoph Lüth and Federico De Marchi, Coalgebraic Monads, *Electron. Notes Theor. Comput. Sci.* 65 (2002), no. 1, 21 pp.

[GLM_2] Neil Ghani, Christoph Lüth and Federico De Marchi, Solving Algebraic Equations using Coalgebra, *Theor. Inform. Appl.* 37 (2003), 301–314.

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

[GLMP$_2$] Neil Ghani, Christoph Lüth, Federico De Marchi and A. John Power, Dualising initial algebras, *Math. Structures Comput. Sci.* 13 (2003), no. 2, 349–370.

[GUs] Neil Ghani and Tarmo Uustalu, Coproducts of ideal monads, *Theor. Inform. Appl.* 38 (2004), no. 4, 321–342.

[Gi$_1$] Susanna Ginali, *Iterative Algebraic Theories, Infinite Trees, and Program Schemata*, PhD thesis, University of Chicago, 1976.

[Gi$_2$] Susanna Ginali, Regular trees and the free iterative theory, *J. Comput. System Sci.* 18 (1979), 228–242.

[GTWW] Joseph Goguen, James W. Thatcher, Eric G. Wagner and Jesse B. Wright, Initial algebra semantics and continuous algebras, *J. ACM* 24 (1977), 68–95.

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

[Gu] Heinz-Peter Gumm, Elements of the General Theory of Coalgebras, LUATCS'99, Rand Africaans University, Johannesburg, South Africa, 1999.

[Ha] Masahito Hasegawa, Recursion from Cyclic Sharing: Traced Monoidal Categories and Models of Cyclic Lambda Calculi, *Proc. 3rd International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Comput. Sci. 1210, 196–213 Springer-Verlag, Berlin, 1997.

[JR] Bart Jacobs and Jan J. M. M. Rutten, A Tutorial on (Co)Algebras and (Co)Induction, *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 62 (1997), 222–259.

[JSV] André Joyal, Ross Street and Dominic Verity, Traced Monoidal Categories, *Math. Proc. Cambridge Philos. Soc.* 119 (1996), no. 3, 447–468.

[L] Joachim Lambek, A fixpoint theorem for complete categories, *Math. Z.* 103 (1968), 151–161.

[La] F. William Lawvere, *Functorial Semantics of Algebraic Theories*, PhD thesis, Columbia University, 1963.

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

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

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

[Li$_1$] Fred E. J. Linton, Some aspects of equational categories, *Proceedings of the Conference on Categorical Algebra*, Springer-Verlag, 1966, 84–94.

[Li$_2$] Fred E. J. Linton, Coequalizers in Categories of Algebras, *Lecture Notes in Math.* 80, Springer-Verlag, New York, 1976.

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

[Ma] Ernest G. Manes, *Algebraic Theories*, Springer Verlag, 1976.

[DM] Federico De Marchi, *Monads in Coalgebra*, PhD thesis, University of Leicester, 2003.

[MU] Ralph Matthes and Tarmo Uustalu, Substitution in Non-Wellfounded Syntax with Variable Binding, *Electron. Notes Theor. Comput. Sci.* 82 (2003), no. 1, 15 pp.

[M$_1$] Stefan Milius, Completely Iterative Algebras and Completely Iterative Monads, *Inform. and Comput.* 196 (2005), 1–41.

[M$_2$] Stefan Milius, On Iteratable Endofunctors, *Electron. Notes Theor. Comput. Sci.* 69 (2002), 18 pp.

[M$_3$] Stefan Milius, The Iterative Reflection of an Ideal Monad, *manuscript* (2005).

[M_4] Stefan Milius, Elgot monads, *manuscript* (2005).

[MM] Stefan Milius and Lawrence S. Moss, The Category Theoretic Solution of Recursive Program Schemes, full version, *submitted*, available via `http://www.iti.cs.tu-bs.de/~milius`, extended abstract appeared in the Proceedings of the first conference on Algebra and Coalgebra in Computer Science (CALCO'05), *Lecture Notes in Comput. Sci.* 3629 (2005), 293–312.

[MS] Gregory Moore and Nathan Seiberg, Classical and quantum conformal field theory, *Comm. Math. Phys.* 123 (1989), no. 2, 177–254.

[Mo_1] Lawrence S. Moss, Parametric Corecursion, *Theoret. Comput. Sci.*, 260 (2001), no. 1–2, 139–163.

[Mo_2] Lawrence S. Moss, The Coalgebraic Treatment of Second-Order Substitution and Uninterpreted Recursive Program Schemes, *preprint* (2002). 2002.

[N] Evelyn Nelson, Iterative Algebras, *Theoret. Comput. Sci.* 25 (1983), 67–94.

[P] Gordon D. Plotkin, *Domains*, The "Pisa" notes, available from `http://www.dcs.ed.ac.uk/home/gdp/`, 1983.

[R_1] J. J. M. M. Rutten, *Automata and coinduction (an exercise in coalgebra)*, Technical Report SEN-R9803, CWI, Amsterdam, 1998.

[R_2] Jan J. M. M. Rutten, Universal coalgebra, a theory of systems, *Theoret. Comput. Sci.* 249 (2000), no. 1, 3–80.

[Ta] Alfred Tarski, A lattice Theoretical Fixed Point Theorem and its Applications, *Pacific J. Math.* 5 (1955), 285–309.

[T] Jerzy Tiuryn, Unique Fixed Points vs. Least Fixed Points, *Theoret. Comput. Sci.* 12 (1980), 229–254.

[TR] Danielle Turi and Jan J. M. M. Rutten, On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces, *Math. Structures Comput. Sci.* 8 (1998), no. 5, 481–540.

[U] Tarmo Uustalu, Generalizing substitution, *Theor. Inform. Appl.* 37 (2003), no. 4, 315–336.

[W_1] James Worrell, *On Coalgebras and Final Semantics*, PhD thesis, Oxford Unversity Computing Laboratory, 2000.

[W_2] James Worrell, On the final sequence of a finitary set functor, *Theoret. Comput. Sci.* 338 (2005), 184–199.

Appendix

In this appendix we include the following of our papers:

[AMV_1]	Free Iterative Theories: a coalgebraic view
[AMV_2]	From Iterative Algebras to Iterative Theories
[AMV_3]	Elgot Algebras
[AAMV]	Infinite Trees and Completely Iterative Theories: A Coalgebraic View
[M_1]	Completely Iterative Algebras and Completely Iterative Monads
[M_2]	On Iteratable Endofunctors
[MM]	The Category Theoretic Solution of Recursive Program Schemes

These papers contain the results with full proofs that we have presented in the preceeding sections. Each of the papers may, of course, be read on its own. Together they can serve as a reference for all the details we had to omit.