České vysoké učení technické v Praze
Fakulta elektrotechnická

Czech Technical University in Prague
Faculty of Electrical Engineering

RNDr Jiří Velebil, PhD

Sémantika nekonečného chování
Semantics of Infinite Behaviour

Summary. In this talk we outline the unifying approach to semantics of infinite behaviour by means of completely iterative algebras. We show that there is a natural coherence result connecting the uininterpreted and interpreted semantics.

Souhrn. V této přednášce nastíníme jednotící přístup k sémantice nekonečného chování pomocí úplně iterativních algeber. Ukážeme existenci přirozené koherence interpretované a neinterpretované sémantiky.

Klíčová slova: algebra, koalgebra, iniciální sémantika, finální sémantika

Keywords: algebra, coalgebra, initial semantics, final semantics

Contents

1 Introduction

The present lecture is devoted to various approaches of defining the semantics of recursive concepts. Recursive definitions are a standard means in modern mathematics and computer science since they allow a neat description of a possibly very complicated behaviour. Besides this advantage, however, we are left with the question of *semantics*: what is the meaning of our recursive specification and how should we interpret it?

Historically two approaches to the above question have been developed. In order to illustrate them, let us consider the following recursive specification

$$fac(n) = ifzero\ n\ then\ 1\ else\ n * fac(n - 1) \tag{1.1}$$

specifying the factorial function.

There are two, seemingly very different, points of view of the meaning of the above specification:

1. *Interpreted Semantics*: this means that various "parts" of the above recursive specification are already interpreted in an existing mathematical structure. Here, the natural interpretation would be that n is a natural number, 1 is the number "one", $*$ denotes multiplication of natural numbers, etc.

   The interpreted semantics of a factorial function is then constructed as a lowest upper bound of a chain of partial functions on natural numbers that give us "better and better" approximations of factorial function:

   f_0 is undefined everywhere
   $f_1(0) = 1$, f_1 is undefined otherwise
   $f_2(0) = 1$, $f_2(1) = 1$, f_2 is undefined otherwise
   $f_3(0) = 1$, $f_3(1) = 1$, $f_3(2) = 2$, f_3 is undefined otherwise
   $f_4(0) = 1$, $f_4(1) = 1$, $f_4(2) = 2$, $f_4(3) = 6$, f_4 is undefined otherwise
   $\vdots$

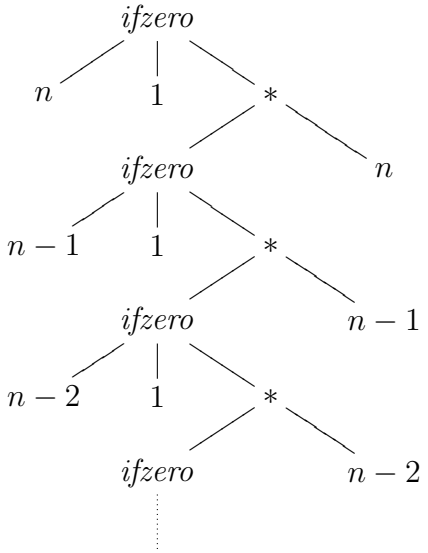
   There is a precise description how to construct the above approximations and a precise sense in which the sequence "converges" to the factorial function — this is covered by the famous fixed-point theorem for complete posets proved by Alfred Tarski in 1955.

   A general feature of classical interpreted semantics is, then, to have an *algebraic structure* (here, the algebra of natural numbers) that at the same time bears an additional *approximation structure* (here, the structure of a complete poset on partial functions) in order to be able to form a "limit" of "finite approximations" that delivers the meaning of the recursive specification in question.

   Let us remark that, in concrete examples, one can use other approximation structures than complete posets: for example, one can work with complete metric spaces and use the famous fixed-point theorem of Stefan Banach.

2. *Uninterpreted Semantics* of equation (1.1) would be purely syntactical: we "unfold"

equation (1.1) to obtain an infinite tree



(1.2)

and say that this tree is the semantics of (1.1). This uninterpreted approach has the advantage that one sees the meaning of a recursive specification immediately but we are left with the following question: where does the above tree "live"? Is there some algebraic structure again? If yes, how does one describe this structure?

We show below that in all "reasonable" situations the above two approaches are in fact two sides of one coin: the syntactic approach deals with algebras of *infinite terms* whereas in the interpreted case we deal with general algebras that allow *computations of infinite terms*. As such, the situation is very similar to the case of classical universal algebra: algebras of (finite) terms are "syntactic models" of an algebraic specification and general algebras allow "interpreted semantics".

In the sequel we will stick to very easy examples from traditional mathematics. What is crucial, however, is the fact that one can apply the same methods to examples (alas, more complicated) of "everyday" programming. This is important: we want to give an overview of methods that allow us to define semantics of constructs known from programming. Such methods necessarily need to be formal: in the ideal case in programming one should verify the semantics of recursive constructions by executing formal tools. Moreover, we want to show that there is a theory of semantics that is *independent* of sets: the carriers of algebraic structures need not be "plain" sets but rather objects of a general category. This, of course, requires to change the "set-biased" way of thinking about recursive specifications. Such a change of paradigm can be done rather harmlessly and results in a conceptually clean description, in fact, in the following text we advocate the following *algebra-coalgebra connection*:

1. Recursive specifications are certain coalgebras in an ambient category,

and

2. semantic domains are algebras in the same category that allow solutions of recursive specifications.

Before we will be able to clarify what we mean by that we give a brief account of methods that have been used to give semantics of *non-recursive specifications*. The reason is, as we will see, that the semantics of recursive specifications is in a certain sense *dual* to the semantics of non-recursive behaviour.

2 Algebraic Specification

The mathematical discipline called *universal algebra* as a study of sets endowed with operations possibly subject to equations dates back to 1890's. The main goals and techniques of this discipline however were formulated only in 1930's by Garrett Birkhoff. It took another thirty years before the relevance of universal algebra to programming has been recognized in the discipline known as *algebraic specification*.

Example 2.1. We want to study a datatype whose models consist of a set equipped with a "multiplication" (satisfying no axioms at all) and an interpretation of elements of a fixed set A (a set of "external constants").

Such a datatype can be specified as follows:

```
spec DATATYPE[A] is
     sorts: object
     operations: interpr(_): A --> object
                 (_) multipl (_): object,object --> object
endspec
```

Thus, models of our datatype are simply triples $\langle X, \star_X, f\rangle$ where X is a set, $\star_X : X \times X \longrightarrow X$ is a binary operation on X and $f : A \longrightarrow X$ is an interpretation of elements of A in X.

Given two such models, on sets X and X', a mapping $h : X \longrightarrow X'$ is called a *homomorphism* if it *respects* both the binary operations and the interpretations.

Homomorphisms therefore serve as "interpretations of one model in another".

Models and homomorphisms can be described more compactly, if we "wrap up" $\star_X$ and f together into a single mapping

$$a : (X \times X) + A \longrightarrow X$$

(having $\star_X$ and f as components).

What is the intended meaning of `DATATYPE[A]`, i.e., which model is "the best"? In algebraic specification, this is called the *initial semantics*, i.e., we choose the model that can be uniquely interpreted in every other model. Stated formally, the initial semantics is a model on set FA such that for every other model on X there exists a unique homomorphism from FA to X.

It is easy to describe the set FA in our concrete example: it is the set of all finite terms that are made out of elements of A by "multiplying freely".

The above process can be performed rather generally: the assignment $X \mapsto X \times X$ can be replaced by any assignment $X \mapsto HX$ that "extends well" to mappings, i.e., one also has an assignment $h \mapsto Hh$ for every mapping $h : X \longrightarrow X'$ satisfying natural axioms

$$H\mathrm{id}_X = \mathrm{id}_{HX} \qquad H(h' \circ h) = Hh' \circ Hh$$

for every set X and every pair h, h' of composable maps. Such an H is called a *functor* (on sets).

If we fix a set A, we study then *algebras* consisting of a set X and a mapping

$$a : HX + A \longrightarrow X \tag{2.1}$$

and their *homomorphisms*, i.e., mappings $h : X \longrightarrow X'$ making the square

$$\begin{array}{ccc} HX + A & \xrightarrow{Hh+A} & HX' + A \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle a'} \\ X & \xrightarrow[h]{} & X' \end{array} \tag{2.2}$$

commutative.

The initial semantics is then an algebra $i_A : HFA + A \longrightarrow FA$ with the property that for every algebra $a : HX + A \longrightarrow X$ there is a unique $a^\flat : FA \longrightarrow X$ making the square

$$\begin{array}{ccc} HFA + A & \xrightarrow{Ha^\flat+A} & HX + A \\ {\scriptstyle i_A}\downarrow & & \downarrow{\scriptstyle a} \\ FA & \xrightarrow[a^\flat]{} & X \end{array} \tag{2.3}$$

commutative.

The nature of FA is again that of "H-terms" on A in a precise sense of category theory.

The above process clearly has nothing to do with sets: one can start with H in an abstract category and mild side conditions then guarantee the existence of initial semantics.

3 Coalgebras

In this section we introduce the *behavioural* view of recursive specifications that leads directly to the uninterpreted semantics.

We choose the recursive equation

$$x = \frac{1}{2}\cos x$$

to explain the basic idea of the behavioural view: we read the equation from left to right with x as a "query" having $\frac{1}{2}\cos x$ as an "answer". This suggests that we observe a behaviour of a system having an infinite set of states

$$x,\ \frac{1}{2}\cos x,\ \frac{1}{2}\cos(\frac{1}{2}\cos x),\ \ldots$$

and a state s has $\frac{1}{2}\cos s$ as its next state.

Let us see a slighly more complicated example of a system.

Example 3.1. Suppose Σ is a signature consisting of a single binary operation symbol $\star$ and let A be a fixed set.

For every set X we denote by $H_\Sigma X$ the set $X \times X$ and we think of $H_\Sigma X$ as of the set of Σ-terms on X of depth ≤ 1, i.e., a pair (x, x'), $x, x' \in X$, is identified with $(x \star x')$.

A mapping

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

then encodes very naturally a recursive specification as follows: we are given a *system of equations*

$$x = e(x),\ x \in X$$

and the right-hand side $e(x)$ of the x-th equation is either an element $a \in A$ (i.e., such an equation specifies no recursive behaviour), or an element $(x', x'') \in H_\Sigma X$ (in this case, recursive behaviour is present).

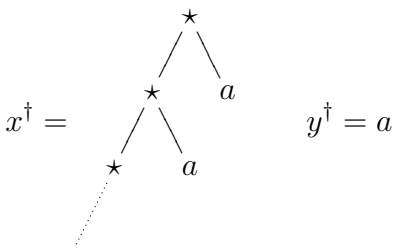
To see an example of such an e, let $X = \{x, y\}$ and $A = \{a\}$ and let $e : X \longrightarrow H_\Sigma X + A$ be given as

$$e(x) = (x, y),\ e(y) = a \tag{3.1}$$

or, written more suggestively, as

$$e(x) = x \star y,\ e(y) = a$$

"Unfolding" the above system of recursive equations results in the following two trees



that are the semantics of the infinite behaviour specified recursively in (3.1).

This suggests that the semantics of recursive specifications of the form $e : X \longrightarrow H_\Sigma X + A$ are systems of (finite or infinite) Σ-trees. Before we will be able to state it more precisely, let us observe the natural candidates for *homomorphisms* of recursive specifications. Given $e : X \longrightarrow H_\Sigma X + A$ and $e' : X' \longrightarrow H_\Sigma X' + A$, a mapping $h : X \longrightarrow X'$ is a homomorphism (from e to e'), if the following square

$$\begin{array}{ccc} X & \xrightarrow{\quad h \quad} & X' \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle e'} \\ H_\Sigma X + A & \xrightarrow[H_\Sigma h + A]{} & H_\Sigma X' + A \end{array} \tag{3.2}$$

commutes, where $H_\Sigma h$ sends $(x \star x')$ to $(h(x) \star h(x'))$. Chasing the elements around the above square we obtain the following behavioural meaning of a homomorphism: e' can, by means of h, simulate the recursive behaviour specified by e.

The above example suggests that recursive behaviour can be studied by gadgets of the form

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

where H is a functor. Such gadgets are called *coalgebras* since they are formally dual to algebras (see (2.1)) and in our context they we will also call them *flat recursive specifications*. *Homomorphisms* from $e : X \longrightarrow HX + A$ to $e' : X' \longrightarrow HX' + A$ are those mappings $h : X \longrightarrow X'$ making the following square

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

commutative (compare with (2.2)).

Now comes the crucial part: what if we make a dual of (2.3), i.e., what if we require the existence of a coalgebra $\alpha_A : TA \longrightarrow HTA + A$ such that for every coalgebra $e : X \longrightarrow HX + A$ there is a unique $e^\sharp : X \longrightarrow TA$ making the square

$$\begin{array}{ccc} X & \xrightarrow{e^\sharp} & TA \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle \alpha_A} \\ HX + A & \xrightarrow[He^\sharp + A]{} & HTA + A \end{array} \tag{3.3}$$

commutative?

This is the so called *final semantics* of recursive specifications: one can prove that (if it exists) each TA can be viewed as a "set of finite and infinite terms on A" in a certain precise sense.

In the case of one binary operation the set TA has the following description (see Example 3.1):

Example 3.2. For $H_\Sigma X = X \times X$, it can be proved that the above $\alpha_A : TA \longrightarrow H_\Sigma TA + A$ exists for every A and it has the following description: elements of TA are (finite and infinite) $\star$-terms visualized as finite and infinite binary trees with leaves labelled in the set A, $\alpha_A : A \longrightarrow TA$ interprets every element of $a \in A$ as a one-node tree (labelled with a) and $\tau_A : H_\Sigma TA \longrightarrow TA$ sends a pair (t_1, t_2) of trees to $(t_1 \star t_2)$. The square (3.3) is then a *Solution Theorem* — the unique mapping $e^\sharp : X \longrightarrow TA$ picks up, for every $x \in X$, a unique tree $e^\sharp(x) \in TA$ that solves the x-th equation of the flat recursive specification $e : X \longrightarrow HX + A$.

As an example, consider the recursive specification (3.1) again. Then the diagram

$$\begin{array}{ccc} X & \xrightarrow{e^\sharp} & TA \\ {\scriptstyle e}\downarrow & & {\scriptstyle \alpha_A}\downarrow \uparrow{\scriptstyle [\tau_A, \eta_A]} \\ H_\Sigma X + A & \xrightarrow[H_\Sigma e^\sharp + A]{} & H_\Sigma TA + A \end{array}$$

expresses precisely the solution of e. To see it, consider elements of $X = \{x, y\}$: we obtain equations

$$e^\sharp(x) = e^\sharp(x) \star e^\sharp(y), \;\; e^\sharp(y) = a$$

that state precisely that $e^\sharp(x)$ and $e^\sharp(y)$ solve the system (3.1).

Therefore, the square (3.3) provides us with the *uninterpreted semantics* of recursive behaviour.

4 Interpreted Semantics

In this section we show that a slight modification of the square (3.3) for final semantics allows us to treat *interpreted semantics* in the same manner as the uininterpreted one. The modification we have in mind is the following: redraw the square

$$\begin{array}{ccc} X & \xrightarrow{e^\sharp} & TA \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle \alpha_A} \\ HX + A & \xrightarrow[He^\sharp + A]{} & HTA + A \end{array}$$

using the mappings $\eta_A : A \longrightarrow TA$ and $\tau_A : HTA \longrightarrow TA$ as follows:

$$\begin{array}{ccc} X & \xrightarrow{\quad e^\sharp \quad} & TA \\ {\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [\tau_A, A]} \\ HX + A & & \\ {\scriptstyle HX+\eta_A}\downarrow & & \\ HX + TA & \xrightarrow[He^\sharp + TA]{} & HTA + TA \end{array} \tag{4.1}$$

The crux of this juggling lies in the fact that we now may try to swap the mapping $\tau_A : HTA \longrightarrow TA$ for a *general* mapping $b : HB \longrightarrow B$ and require the following

> For every $e : X \longrightarrow HX + B$ there exists a unique $e^\dagger : X \longrightarrow B$ making the following diagram
>
> $$\begin{array}{ccc} X & \xrightarrow{\quad e^\dagger \quad} & B \\ {\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [b, B]} \\ HX + B & \xrightarrow[He^\dagger + B]{} & HB + B \end{array} \tag{4.2}$$
>
> commutative.

The general mapping $b : HB \longrightarrow B$ then states that terms of depth ≤ 1 can be interpreted in B and the property (4.2) demands that every flat recursive specification e has a *unique solution* $e^\dagger$ in B.

Mappings $b : HB \longrightarrow B$ with the above property (4.2) are called *completely iterative algebras* (*CIAs*, for short). Every CIA $b : HB \longrightarrow B$ then comes equipped with a canonical map

$$\widehat{b} : TB \longrightarrow B$$

that allows to compute *all terms* in B. This map is defined by $\widehat{b} = {\alpha_B}^\dagger$ from the instance of (4.2):

$$\begin{array}{ccc} TB & \xrightarrow{\quad {\alpha_B}^\dagger \quad} & B \\ {\scriptstyle \alpha_B}\downarrow & & \uparrow{\scriptstyle [b, B]} \\ HTB + B & \xrightarrow[H{\alpha_B}^\dagger + B]{} & HB + B \end{array}$$

Moreover, we obtain immediately the *Coherence Theorem* stating that, given a CIA $b : HB \longrightarrow B$ and a recursive specification $e : X \longrightarrow HX + B$, the *interpreted semantics* $e^\dagger : X \longrightarrow B$ coincides with the *canonical interpretation of the uninterpreted semantics*, i.e., that the diagram

$$\begin{array}{ccccc} & & e^\dagger & & \\ & \overset{\frown}{} & & \overset{\frown}{} & \\ X & \xrightarrow{\quad e^\sharp \quad} & TB & \xrightarrow{\quad {\alpha_B}^\dagger \quad} & B \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle \alpha_B} & & \uparrow{\scriptstyle [b, B]} \\ HX + B & \xrightarrow[He^\sharp + B]{} & HTB + B & \xrightarrow[H{\alpha_B}^\dagger + B]{} & HB + B \\ & \underset{\smile}{} & & \underset{\smile}{} & \\ & & \underset{He^\dagger + B}{} & & \end{array}$$

commutes.

The conclusion is then the following: every CIA $b : HB \longrightarrow B$ is then a suitable semantics domain for recursive specifications of the form $e : X \longrightarrow HX + B$. The interpreted semantics $e^\dagger : X \longrightarrow B$ is then the uninterpreted semantics $e^\sharp : X \longrightarrow TB$ followed by the canonical computation function $\widehat{b} : TB \longrightarrow B$.

5 Conclusion

We indicated that the coalgebraic-algebraic approach to recursive specifications offers a unifying frame that allows one to treat recursive specifications and their semantics independently of sets. This approach had already allowed to give semantics of *recursive program schemes*, see [MoMi] by Larry Moss and Stefan Milius.

We emphasize now the main features and possible future developments of the approach presented in this lecture:

1. Independence of sets: besides from the study of recursive program schemes mentioned above, the technique allows one to define and study the algebra of *algebraic trees* that appear naturally in programming applications, see the paper [C] of Bruno Courcelle.

2. The same technique can be used to study *finitary recursive specifications* in a more general setting than "plain" sets, see [AMV$_1$] by Jiří Adámek, Stefan Milius and the current author. Finitary recursive specifications have their importance in practice. In the cited paper main properties of the resulting algebra of *rational trees* have been proved in a rather general setting.

3. Beside "full" recursion treated in this lecture, one can *a priori* restrict certain variables that may enter the recursion and studied *parametrized recursive specifications*. The first steps in this direction have been made in [AMV$_2$].

4. Algebras that allow solutions of *all* recursive specifications have been studied by Stepen Bloom and Zoltán Ésik in [BÉ]. Recently, the existence of algebras that allow the so called *strict solutions* of recursive specifications have been studied by John Mersch [Me] and by Jiří Adámek, Stefan Milius and the current author in [AMV$_3$].

6 Historical Comments

The fixed-point theorems that allow "classical" interpreted semantics are due to Stefan Banach (the case of complete metric spaces, [Ba]) and Alfred Tarski (the case of complete posets, see [Ta]).

The main goals of universal algebra were formulated by Garret Birkhoff in [Bi] and their categorical couterparts by William Lawvere in [L]. The relevance of universal algebra (and most of all of the initial semantics) to computer science was emphasised by various authors in the late 1960's, for an overview see, e.g., the books [AT] and [W].

The pioneering work of Calvin Elgot and his collaborators, [E], [BE], [EBT], in the field of recursive equations cannot be overestimated. In the abovementioned three papers *iterative theories* were introduced in order to avoid additional approximation structure as, e.g., metric spaces or partial orders traditionally used in models of infinite behaviour,

see, e.g., [AR] or [SP]. The language Elgot *et al.* used is the language of universal algebra and the techniques are rather refined.

After the work of Peter Aczel on *antifoundation axiom* in set theory, see [A], it gradually became clear that recursively defined objects are best seen as coalgebras. The summary of the state-of-art on "self-referential" objects in set theory and elsewhere in the mid-1990's is presented in the monograph [BM], see also Jan Rutten's paper [R] for coalgebraic description of systems arising naturally in computer science.

When Peter Aczel gave talks during the summer school *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction* in Oxford in April 2000, the time was ripe to formulate the fascinating properties of the algebra of infinite terms generally. Indeed, the papers [Mo], [GLMP] and [AAMV] were written independently but almost parallel in time.

The crucial point of recognizing the algebra-coalgebra connection and of introducing *completely iterative algebras* (see [Mi]) is due to the inspiring work of Evelyn Nelson [N] and Jerzy Tiurin [T].

References

[A] P. Aczel, *Non-well-founded Sets*, CSLI Lecture Notes No. 14, Stanford University, Stanford, CA, 1988

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

[AMV_1] J . Adámek, S. Milius and J. Velebil: From Iterative Algebras to Iterative Theories (Extended Abstract), Electron. Notes Theor. Comput. Sci., 106 (2004), 3–24

[AMV_2] J. Adámek, S. Milius and J. Velebil, Algebras With Parametrized Iterativity, (2005), *submitted*

[AMV_3] J. Adámek, S. Milius and J. Velebil, How Iterative are Iterative Algebras? *Electron. Notes Theor. Comput. Sci.* 164 (2006), 157–175

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

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

[Ba] S. Banach, Sur les opérations dans les ensembles abstraits et leurs applications aux équations intégrales, *Fund. Math.* 3 (1922), 133–181

[Bi] G. Birkhoff, On the Structure of Abstract Algebras, *Proc. Camb. Phil. Soc.* 31 (1935), 433–454

[BM] J. Barwise and L. Moss, *Vicious Circles*, CSLI Lecture Notes No. 60, Stanford University, Stanford, CA, 1996

[BE] S. L. Bloom and C. C. Elgot, The Existence and Construction of Free Iterative Theories, *J. Comput. System Sci.* 12 (1976), 305–318

[BÉ] S. L. Bloom and Z. Ésik, *Iteration Theories: The Equational Logic of Iterative Processes*, EATCS Monograph Series on Theoretical Computer Science, Springer-Verlag, 1993

[C] B. Courcelle, Fundamental Properties of Infinite Trees, *Theoret. Comput. Sci.* 25 (1983), 95–169

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

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

[GLMP] N. Ghani, C. Lüth, F. de Marchi and J. Power, Algebras, Coalgebras, Monads and Comonads, *Electron. Notes Theor. Comput. Sci.* 44.1 (2001)

[L] F. W. Lawvere, *Functorial Semantics of Algebraic Theories*, PhD Thesis, Columbia University, 1963

[M] E. G. Manes, *Algebraic Theories*, Graduate Texts in Mathematics, Springer-Verlag, New York, 1976

[Me] J. G. Mersch, Equational Logic of Recursive Program Schemes, LNCS 3629, Springer-Verlag 2005, 278–292

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

[Mo] L. Moss, Parametric Corecursion, *Theoret. Comput. Sci.* 260 (2001), 139–163

[MoMi] L. Moss and S. Milius, The Category Theoretic Solution of Recursive Program Schemes, to appear in Springer Lecture Notes in Computer Science

[N] E. Nelson, Iterative Algebras, *Theoret. Comput. Sci.* 25 (1983), 67–94

[R] J. J. M. M. Rutten, Universal Coalgebra: A Theory of Systems, *Theoret. Comput. Sci.* 249 (2000), 3–80

[SP] M. B. Smyth and G. D. Plotkin, The Category-Theoretic Solution of Recursive Domain Equations, *SIAM J. Comput.* 11 (1982), 761–783

[Ta] A. Tarski, A Lattice Theoretical Fixpoint Theorem and its Applications, *Pacific J. Math.* 5 (1955), 285–309

[T] J. Tiurin, Unique Fixed Points vs. Least Fixed Points, *Theoret. Comput. Sci.* 12 (1980), 229–254

[W] W. Wechler, *Universal Algebra for Computer Scientists*, Springer-Verlag, Berlin, 1992

7 RNDr Jiří Velebil, PhD

Address: Department of Mathematics,
FEL ČVUT,
Technická 2, 166 27 Praha 6,
Czech Republic

e-mail: velebil@math.feld.cvut.cz

Education:

1998: PhD in mathematics, FEL ČVUT, Praha, Czech Republic
supervisor: Prof. Jiří Adámek
thesis: *Categorical Domain Theory*

1993–1998: PhD student, FEL ČVUT, Praha, Czech Republic

1988: RNDr in mathematics, Charles University, Praha, Czech Republic

1987: Diploma in mathematics, Charles University, Praha, Czech Republic
supervisor: Prof. Jan Čerych
thesis: *Normal and Approximatively Normal Function Algebras* (in Czech)

1982–1987: undergraduate student of mathematics, Charles University, Praha, Czech Republic

Employment:

1991– : assistant professor, Department of Mathematics, FEL ČVUT, Praha, Czech Republic

1990: study stay, Department of Applied Mathematics, Charles University, Praha, Czech Republic

1989–1990: military service

1988–1989: high school teacher of mathematics, Praha, Czech Republic

1987–1988: study stay, Department of Mathematical Analysis, Charles University, Praha, Czech Republic

Publications:

1. J. Adámek, S. Milius, J. Velebil: Iterative Algebras at Work, *Math. Structures Comput. Sci.* 16.6 (2006), 1085–1131

2. J. Adámek, S. Milius, J. Velebil: Elgot Algebras, *Logical Methods in Computer Science* Vol. 2(5:4) (2006), 1–31

3. J. Adámek, S. Milius, J. Velebil: How Iterative are Iterative Algebras?, *Electron. Notes Theor. Comput. Sci.* 164 (2006), 157–175

4. J. Adámek, S. Milius, J. Velebil: Elgot Algebras (Extended Abstract), *Electron. Notes Theor. Comput. Sci.* 155 (2006), 87–109

5. J. Adámek, S. Milius, J. Velebil: A General Final Coalgebra Theorem, *Math. Structures Comput. Sci.*, 15.3 (2005), 409–432

6. J. Adámek, S. Milius, J. Velebil: Iterative Algebras for a Base, *Electron. Notes Theor. Comput. Sci.*, 122 (2005), 147–170

7. P. Karazeris, J. Rosický, J. Velebil: Completeness of Cocompletions, *J. Pure Appl. Algebra*, 196 (2005), 229–250

8. J. Adámek, S. Milius, J. Velebil: From Iterative Algebras to Iterative Theories (Extended Abstract), *Electron. Notes Theor. Comput. Sci.*, 106 (2004), 3–24

9. J. Adámek, S. Milius, J. Velebil: On Coalgebra Based on Classes, *Theoret. Comput. Sci.*, 316 (2004), 3–23

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

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

12. J. Adámek, S. Milius, J. Velebil: Free Iterative Theories — A Coalgebraic View, *Math. Structures Comput. Sci.*, 13 (2003), 259–320

13. J. Adámek, S. Milius, J. Velebil: On Rational Monads and Free Iterative Theories, *Electron. Notes Theor. Comput. Sci.*, 69 (2003)

14. R. El Bashir, J. Velebil: Simultaneously Reflective and Coreflective Subcategories of Presheaves, *Theory Appl. Categ.*, Vol.10 (2002), 410–423

15. J. Adámek, S. Milius, J. Velebil: Final Coalgebras and a Solution Theorem for Arbitrary Endofunctors, *Electron. Notes Theor. Comput. Sci.*, 65.1 (2002)

16. J. Velebil, J. Adámek: A Remark on Conservative Cocompletions of Categories, *J. Pure Appl. Algebra*, 168 (2002), 107–124

17. J. Adámek, R. El Bashir, M. Sobral, J. Velebil: On Functors Which Are Lax Epimorphisms, *Theory Appl. Categ.*, Vol.8 (2001), 509–521

18. P. Aczel, J. Adámek, J. Velebil: A Coalgebraic View of Infinite Trees and Iteration, *Electron. Notes Theor. Comput. Sci.*, 44.1 (2001)

19. J. Adámek, V. Koubek, J. Velebil: A Duality Between Infinitary Varieties and Algebraic Theories, *Comment. Math. Univ. Carolin.*, 41,3 (2000), 529–541

20. V. Trnková, J. Velebil: On Categories Generalizing Universal Domains, *Math. Structures Comput. Sci.*, (1999), vol. 9, 159–175

21. J. Velebil: Categorical Generalization of a Universal Domain, *Appl. Categ. Structures*, 7 (1999), 209–226

22. K. Richta, J. Velebil: *Semantics of Programming Languages*, lecture notes (in Czech), 169pp., Karolinum Publishing House, Praha 1997

Stays Abroad:

Feb 2002–Aug 2003: Wissenschaftlicher Mitarbeiter,
Institut für Theoretische Informatik,
TU Braunschweig, Germany

Oct 1999–Jan 2000: Visiting Scholar,
School of Mathematics and Statistics,
University of Sydney, Australia

Jan 1994–Apr 1994: PhD student,
Laboratory for Foundations of Computer Science
University of Edinburgh, United Kingdom