UDC 004.423.45
Original Scientific Paper

Some Useful Structures for Categorical Approach for Program Behavior

Viliam Slodičák *viliam.slodicak@tuke.sk*
Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical university of Košice
Letná 9, 042 00 Košice
Slovak Republic

Abstract

Using of category theory in computer science has extremely grown in the last decade. Categories allow us to express mathematical structures in unified way. Algebras are used for constructing basic structures used in computer programs. A program can be considered as an element of the initial algebra arising from the used programming language. In our contribution we formulate two ways of expressing algebras in categories. We also construct the codomain functor from the arrow category of algebras into the base category of sets which objects are also the carrier-sets of the algebras. This functor expresses the relation between algebras and carrier-sets.

Keywords: Algebra, arrow category, monad, Kleisli category, codomain functor

1. Introduction

Knowing and proving of the expected behavior of complex program systems is very important and actual *rôle*. It carries the time and cost savings: in mathematics [8] or in practical applications of economical character. The aim of programming is to construct such correct programs and program systems that during their execution provide expected behavior [7]. A program can be considered as an element of the initial algebra arising from the used programming language [14]. Algebraic structures and number systems are widely used in computer science. They allow to abstract from concrete objects which lead to the mathematical branches of abstract algebra and universal algebra. On the other hand, category theory provides possibilities to model many important features of computer science [2, 6]. It affords suitable structures for the describing program construction using algebras $T(C) \to C$ and for modelling observable behavior using coalgebras $C \to T(C)$, where C is a category object and T is a polynomial endofunctor induced by a signature. Algebra and coalgebra are from category's point of view dual constructions [17]. In this paper we present two ways of expressing the relation between T-algebras and their carrier-sets. We construct arrow category of algebras and Kleisli arrow category of algebras where objects are morphisms from a base category. The relation we will formulate with the codomain functor which takes objects of arrow category into morphism codomains from base category.

2. Basic Concepts

Algebraic and coalgebraic concepts are based on category theory [4, 11]. A category $\mathscr{C}$ is a mathematical structure consisting of objects, e.g. $A, B, C, \ldots$ and morphisms of the form $f : A \to B$ between objects. Every object has the identity morphism $id_A : A \to A$ and morphisms are composable. Because the objects of category can be arbitrary structures, categories are useful

in computer science, where we often use more complex structures not expressible by sets. Morphisms between categories are called functors. A functor $F : \mathscr{C} \to \mathscr{D}$ from a category $\mathscr{C}$ into a category $\mathscr{D}$ preserves the structure, i.e. it sends the objects A, B from $\mathscr{C}$ into the objects FA, FB from $\mathscr{D}$ and the morphism $f : A \to B$ from $\mathscr{C}$ into the morphism $Ff : FA \to FB$ from $\mathscr{D}$. In this contribution we consider only the category $\mathscr{S}et$ with sets as objects and functions between them as morphisms, but this approach can be extended to categories of arbitrary complex objects.

3. Algebras in category

In our research we are interested in formal description of program construction by algebras and observation of program behavior by coalgebras. Construction of algebras over the signatures were introduced in [11, 15]. We construct a polynomial endofunctor over the category $\mathscr{S}et$ of sets for substantiation of the signature operations for a given program. Let T be an endofunctor

$$T : \mathscr{S}et \to \mathscr{S}et. \tag{1}$$

Algebras over signatures we construct in category. Operations in signature determine polynomial endofunctor that can be constructed inductively from T using constants, identities, products, co-products and powersets. One of the most used categorical forms of algebra is as follows: T-algebra, the model of signature, is a pair

$$(A, a)$$

where A is a carrier-set, a representation of a type. The algebraic structure (or structuring map)

$$a = [cons_1, \ldots, cons_n] : TA \to A$$

is defined as cotuple function of constructors $cons_1, \ldots, cons_n$.

The relations between algebras are defined by algebra homomorphisms. Let (A, a) and (B, b) be T-algebras. A homomorphism $f : (A, a) \to (B, b)$ of T-algebras is the function $f : A \to B$ between carrier-sets which commutes with the operations as it is illustrated on the following diagram at the Fig. 1

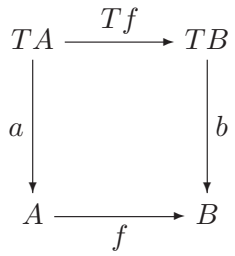


Figure 1: The relation between algebras

so it holds the equality $f \circ a = b \circ Tf$.

Homomorphisms of T-algebras can be composed, and every T-algebra (A, a) has the identity homomorphism $id_{(A,a)} : (A, a) \to (A, a)$. Therefore we can construct the category $\mathscr{A}lg$ of T-algebras consisting of T-algebras as objects and homomorphisms between them as category morphisms. The most important concept in algebraic approach is the initial T-algebra [1, 11]. A T-algebra is initial if for arbitrary T-algebra there exists unique homomorphism from initial to arbitrary T-algebra. Initial T-algebras, if they exists have some important properties:

- they are unique up to isomorphism, therefore we write the initial T-algebra as $u : TU \cong U$, and
- the initial algebra has an inverse $u^{-1} : U \to TU$.

In the other words, from the first property it follows that there exists at most one initial T-algebra. Because from the initial T-algebra exists unique homomorphism to every T-algebra, the initial T-algebra is the initial object in the category $\mathscr{A}lg$. The second property was proved in [12] and it says that the initial T-algebra is the least fixed point of the functor T. Initial algebras are generalizations of the least fixed points of monotone functions, since they have unique maps into arbitrary T-algebra.

Such formulated category of T-algebras allows us to work with algebras as with category objects. If we want to formulate the relations between algebras and carrier-sets, we need to define the couple of two functors

$$U : \mathscr{A}lg \to \mathscr{S}et \qquad F : \mathscr{S}et \to \mathscr{A}lg$$

which we call *the adjoint functors* [5]

$$F \dashv U.$$

The functor U is forgetful functor which assigns to any algebra from category of sets an appropriate carrier-set from the category of sets. Vice versa, the functor F is defined as functor assigning to each carrier-set an appropriate algebraic structure. But there is also another form of representation of algebras in category. By availing of the algebra properties and using some special categories we enclose algebras in the arrow category.

4. Arrow category for algebras

We define algebras for simpler working and expressing in category of another form - we interpret algebraic structure given by the pair (A, a) as a map

$$TA \xrightarrow{a} A.$$

For such expressed algebras we define category of morphisms - the arrow category. For formulation of the relation between algebras and carrier-sets we construct the codomain functor from arrow category into category of carrier-sets. This category of algebras we denote $\mathscr{T}Alg$. The objects are the algebras of the form $TA \xrightarrow{a} A$, $TB \xrightarrow{b} B$, ... as objects and morphisms between objects. Morphisms are algebra homomorphisms of the form (f, a, b), where map f is the function between codomains of the appropriate algebras - the carrier-sets A and B (Fig. 2)

$$f : cod(a) \to cod(b)$$

where $cod(a) = A$ and $cod(b) = B$ are the codomains of the algebraic structures a and b respectively.

We must prove the following properties, that have to be valid for $\mathscr{T}Alg$ be the category. We define for each algebra $TA \xrightarrow{a} A$ the identity morphism of the form (id_A, a, a) expressed by the commutative diagram at Fig. 3.
It also holds that morphisms are composable: for (f, a, b) and (g, b, c) we have $(g \circ f, a, c)$ as it is depicted at Fig. 4.
The initial algebra in the category $\mathscr{T}Alg$ is its initial object. It is the least fixed point of the functor T. The least fixed point of the functor T we denote also as ϕT. Seeing that it is the T-algebra, there exists the algebra operation in defined as

$$in : T(\phi T) \to \phi T.$$

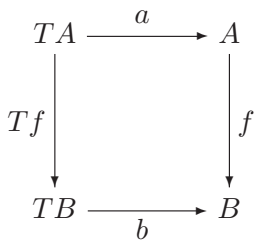


Figure 2: Morphism of algebras

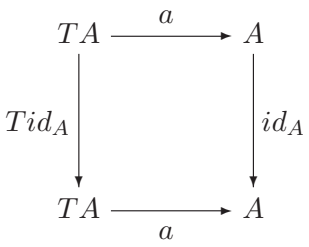


Figure 3: Identity morphism of algebras

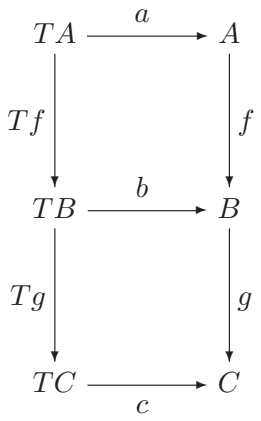


Figure 4: Composition of algebra homomorphism

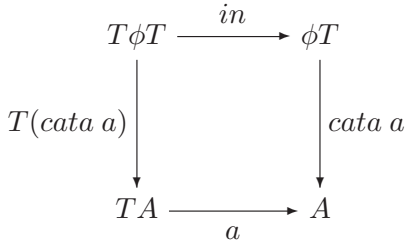


Figure 5: Diagram for initial algebra

The T-algebra $(\phi T, in)$ is the initial T-algebra, if for any T-algebra (A, a) there exists an unique arrow $cata\ a : \phi T \to A$ making the diagram at Fig. 5 to commute,
i.e. the following equality holds

$$cata\ a \circ in = a \circ T(cata\ a).$$

The morphism $cata(-)$ we call the *catamorphism*. The initial algebra $(\mu T, in)$ is the initial object in the category $\mathscr{T}Alg$ and the catamorphism $cata(-)$ is the mediating arrow out of it. It also holds that the initial algebra exists if T is ω-cocontinuous (i.e. it preserves the colimits of ω-chains) [4]. From the existence of the initial algebra it implies the property called the *reflection*, so it holds

$$id = cata\ in.$$

5. Monads

From one point of view, a monad is an abstraction of certain properties of algebraic structures. From another point of view, it is an abstraction of certain properties of adjoint functors. Theory of monads has turned out to be an important tool also for studying toposes [5].

5.1 Definition

A *monad*

$$\mathcal{T} = (T, \eta, \mu)$$

on a category $\mathscr{S}et$ is an endofunctor

$$T : \mathscr{S}et \to \mathscr{S}et$$

together with two natural transformations

- $\eta : Id_{\mathscr{S}et} \to T$ called *unit*;
- $\mu : T^2 \to T$ called *multiplication*.

subject to the condition that the diagrams at Fig. 6 and Fig. 7 commute.

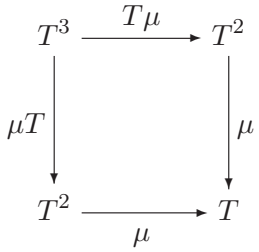


Figure 6: Coherence square for monad

If we consider a monad over category $\mathscr{S}et$ of sets then the unit transformation is a map $\eta X : X \to TX$ for each set X satisfying a suitable naturality condition. The multiplication transformation consists of functions $\mu X : T^2X \to TX$ with X ranging over sets. Next example illustrates monad that involves monoids.

Example. Let $\mathbf{M} = (M, \odot, e)$ be a monoid and the polynomial functor $T : \mathscr{S}et \to \mathscr{S}et$ be defined by $TX = M \times X$.

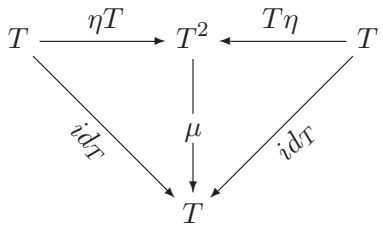


Figure 7: Coherence triangle for monad

Let $\eta X : X \to M \times X$ be the morphism assigning to x a pair (e, x) and let $\mu X : M \times M \times X \to M \times X$ be the another morphism that takes (m, n, x) to $(m \odot n, x)$. Identities at Fig. 7 follow from those on M.

□

The monad structures play a crucial rôle in modeling "branching". Intuitively, the unit η embeds a non-branching behavior as a trivial branching with only one possibility to choose. The multiplication μ "flattens" two successive branching into one branching, abstracting away internal branching [9].

The notion of "algebras for a monad" generalizes classical notions from universal algebra, and in this sense, monads can be thought of as "theories". Every monad is defined by its T-algebras [3]. T-algebras for a monad $\mathcal{T}$ should interact properly with the extra structure on $\mathcal{T}$. A T-algebra is an arrow $a : TA \to A$ as before, such that the diagrams at Fig. 8 and Fig. 9 commute.

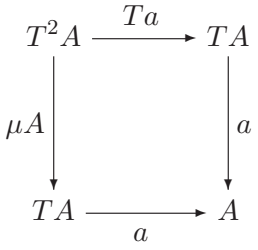


Figure 8: Algebra in monad via multiplication

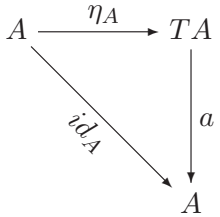


Figure 9: Algebra in monad via unit

In our approach we want to make use of the useful categorical structures - arrow category and Kleisli category.

5.2 Kleisli categories

Kleisli category is the kind of category which should be investigated for the functional programming paradigm or for the generalizing the structures in category [5, 13]. This category is an extremal solution of the problem of constructing an adjunction that gives rise to the given monad. A monad is defined as an endofunctor which can be considered as the composition of two adjoint functors. Its dual concept, the comonad has useful properties for behavioral theory [16]. Recognizing the categories of coalgebras for a comonad is an important tool of topos theory .

The relevance of Kleisli categories in usual coalgebraic approach is that Kleisli category can be thought of as a category where the branching is implicit [9, 16].

Given any monad $\mathcal{T}$, its *Kleisli category* $\mathscr{K}(\mathcal{T})$ is defined as follows. Its objects are the objects of the base category, hence sets in the current setting. An arrow $A \to B$ in $\mathscr{K}(\mathcal{T})$ is the same thing as an arrow $A \to TB$ in the base category, here $\mathscr{S}et$:

$$\frac{A \to B \ \text{ in } \mathscr{K}(\mathcal{T})}{A \to TB \ \text{ in } \mathscr{S}et}.$$

Identities and compositions of arrows are defined using the unit and the multiplication of $\mathcal{T}$. Moreover, there is a canonical adjunction

$$F \dashv U$$

where functors are:

$$F : \mathscr{S}et \to \mathscr{K}(\mathcal{T}) \qquad U : \mathscr{K}(\mathcal{T}) \to \mathscr{S}et.$$

In this adjunction the right adjoint U assigns an arrow $f : A \to B$ in $\mathscr{K}(\mathcal{T})$ (i.e. a function $f : A \to TB$ in $\mathscr{S}et$) to a map

$$TA \xrightarrow{Tf} T^2B \xrightarrow{\mu B} TB$$

in $\mathscr{S}et$. Moreover, compositions of arrows in category $\mathscr{K}(\mathcal{T})$ are given by

$$A \xrightarrow{f} B \xrightarrow{g} C$$

as the composition

$$A \xrightarrow{f} TB \xrightarrow{Tg} T^2C \xrightarrow{\mu C} TC$$

in the category $\mathscr{S}et$. The composition $\mu C \circ Tg$ is the unique lifting of g to the free T-algebra on its domain [10]. The Kleisli category $\mathscr{K}(\mathcal{T})$ is equivalent to the subcategory of $\mathscr{T}Alg$ consisting of the free algebras of the form

$$\mu A : T^2A \to TA.$$

Every object A of the category $\mathscr{K}(\mathcal{T})$ uniquely generates free algebra TA and actions μA. An arrow f from the Kleisli category lifts uniquely to arrow $\mu B \circ Tf$ (Fig. 10).

6. Codomain functor

Codomain functor is the special functor defined for arrow category. Codomain functor is always defined for the arrow category and the appropriate base category [10]. The arrow category over the base category is a mathematical structure consisting of

- an object of arrow category is an arrow (morphism) of the base category;

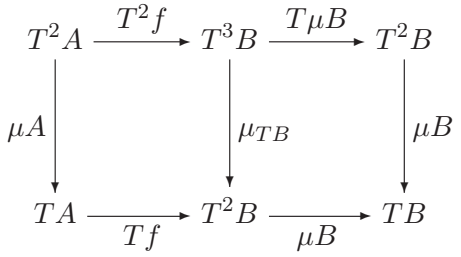
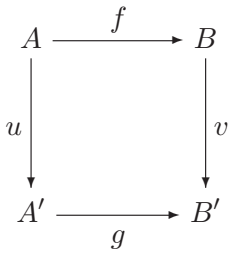


Figure 10: Algebras in Kleisli category

- given two objects $A \xrightarrow{f} B$, $A' \xrightarrow{g} B'$, a morphism from f to g consists of an ordered pair (u, v), where $A \xrightarrow{u} A'$, $B \xrightarrow{v} B'$ such that the following diagram



  is a commutative diagram. For purpose of this approach we can consider the morphism of an arrow category also in the form (v, f, g).

We construct codomain functor from category of algebras $\mathscr{T}Alg$ into category of sets $\mathscr{S}et$ which objects are also the carrier-sets of the algebras:

$$Cod : \mathscr{T}Alg \to \mathscr{S}et.$$

The functor Cod sends an object of the category $\mathscr{T}Alg$ (algebras) into the category $\mathscr{S}et$: it assigns an appropriate carrier-set to the given algebra:

$$Cod(TA \xrightarrow{a} A) = A.$$

The functor Cod according to the definition sends the morphism of the category $\mathscr{T}Alg$ - the algebra homomorphism into the appropriate morphism of the category of sets. Let $TA \xrightarrow{a} A$, $TB \xrightarrow{b} B$ and $TC \xrightarrow{c} C$ be the objects of the category $\mathscr{T}Alg$ and let (f, a, b), (g, b, c) be the morphisms where f and g are the codomain maps

$$f : cod(a) \to cod(b) \qquad g : cod(b) \to cod(c).$$

Then it holds

$$Cod(f, a, b) = f.$$

For the identity morphism id_a we have

$$Cod(id_a) = (id_A, a, a) = id_A$$

which satisfies the definition of the codomain functor. Functor Cod also preserves the composition of the morphisms:

$$Cod(g \circ f, a, c) = g \circ f.$$

The codomain functor from arrow category into appropriate base category always exists. We do not need to define extra adjoint functors to formulate the relation between algebras and the carrier-sets.

6.1 Codomain functor for the Kleisli category

We defined the Kleisli category $\mathscr{K}(\mathscr{T})$ of a monad $\mathcal{T} = (T, \mu, \eta)$. Now we construct the arrow category over $\mathscr{K}(\mathscr{T})$ denoted $\mathscr{K}(\mathscr{T})^{\rightarrow}$ as follows:

- objects are algebras of the form

$$\mu A : T^2 A \rightarrow TA;$$

- morphisms are algebra homomorphisms of the form $(Tf, \mu A, \mu B)$ such that the following diagram at Fig. 11 commutes;

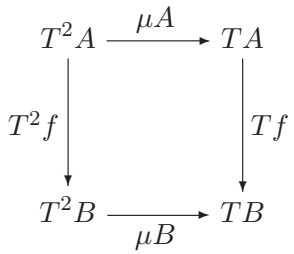


Figure 11: Morphism of algebras in category $\mathscr{K}(\mathscr{T})$

- identity has the form $(Tid_A, \mu A, \mu A)$;
- composition of two algebras $(Tf, \mu A, \mu B)$ and $(Tg, \mu B, \mu C)$ is $(Tg \circ Tf, \mu A, \mu C)$.

Next we define the codomain functor $\mathscr{K}od$ for the Kleisli arrow category. The functor has the form

$$\mathscr{K}od : \mathscr{K}(\mathscr{T})^{\rightarrow} \rightarrow \mathscr{S}et.$$

Codomain functor $\mathscr{K}od$ sends the objects of Kleisli arrow category of algebras into the category of carrier-sets $\mathscr{S}et$ such that it assigns to any algebra $\mu A : T^2 A \rightarrow TA$ the appropriate carrier-set:

$$\mathscr{K}od\left(T^2 A \xrightarrow{\mu A} TA\right) = TA.$$

Functor $\mathscr{K}od$ according to definition maps the morphisms of category $\mathscr{K}(\mathscr{T})^{\rightarrow}$ (the algebra homomorphisms) into the appropriate morphisms of the category of carrier-sets. Let's have the algebras $\mu A : T^2 A \rightarrow TA$, $\mu B : T^2 B \rightarrow TB$, $\mu C : T^2 C \rightarrow TC$ with their morphisms $(Tf, \mu A, \mu B)$ and $(Tg, \mu B, \mu C)$, where Tf and Tg are the codomain maps

$$Tf : cod(\mu A) \rightarrow cod(\mu B)$$

$$Tg : cod(\mu B) \rightarrow cod(\mu C).$$

For the algebra homomorphisms according to the definition of Kleisli category it holds that

$$\mathscr{K}od\,(Tf, \mu A, \mu B) = \mu B \circ Tf.$$

For identity homomorphisms it holds that

$$\mathscr{K}od\,(Tid_A, \mu A, \mu A) = \mu A \circ \eta_A.$$

Codomain functor $\mathscr{K}od$ also preserves the composition of algebra homomorphisms:

$$\mathscr{K}od\,(Tg \circ Tf, \mu A, \mu C) = (\mu C \circ Tg) \circ (\mu B \circ Tf).$$

If we want to formulate the relation between algebras and their carrier-sets, we need to construct the pair of adjoint functors $F \dashv U$. For construction of this adjunction is a non-trivial matter and its existence has to be proven. The codomain functor from Kleisli arrow category $\mathscr{K}(\mathcal{T})^{\rightarrow}$ into the base category always exists. This functor expresses explicitly the relation between algebras and their carrier-sets. It assigns to each algebra its appropriate carrier-set.

7. Conclusion

In this paper we formulated the expression of algebras in the arrow category. The relation between algebras and their carrier-sets we constructed as codomain functor from the arrow category $\mathscr{T}Alg$ into the base category $\mathscr{S}et$ of sets. We also formulated another approach of expressing algebras in Kleisli arrow category. We defined that relation between algebras and their carrier-sets as codomain functor from $\mathscr{K}(\mathcal{T})^{\rightarrow}$ into the base category of sets. The codomain functor for the arrow category is always defined, that's why our approach does not need to construct the couple of adjoint functors and to prove the construction. In our next research we will focus on suitable categorical structures as a base for algebraic description of construction and coalgebraic behavior of program systems. We would like to apply achieved theoretical results to real non trivial program systems from the area of computer networks, database systems and distributed systems.

Acknowledgments

This work has been supported by VEGA Grant No.1/0015/10: Principles and methods of semantic enrichment and adaptation of knowledge-based languages for automatic software development.

References

[1] ADÁMEK, J., ROSICKÝ, J., AND VITALE, E. Algebraic theories: a categorical introduction to general algebra. With a Preface by F.W. Lawvere, April 2008.

[2] AWODEY, S. *Category Theory*. Carnegie Mellon University, 2005.

[3] BANDA, A. Algebras over monads, 2007. African Institute for Mathematical Sciences, University of Cape Town.

[4] BARR, M. Terminal coalgebras for endofunctors on sets, 1999.

[5] BARR, M., AND WELLS, C. *Toposes, Triples and Theories*. Springer-Verlag, 2002.

[6] EHRIG, H. Applied and computational category theory. In *Bulletin of the EATCS no 89* (June 2006), European Association for Theoretical Computer Science, pp. 134–135.

[7] GRABARA, J. *IT Solutions for power energy production and distribution*. Polskie Towarzystwo Informatyczne, Katowice, Poland, 2006.

[8] GRZYBOWSKI, A. Computer simulations in constructing a coefficient of uncertainty in regression estimation - methodology and results. In *Lectures Notes in Computer Science*, vol. 2328/2006. Springer-Verlag, Berlin.

[9] HASUO, I. *Tracing Anonymity with Coalgebras*. PhD thesis, Radboud University Nijmegen, 2008.

[10] JACOBS, B. *Categorical Logic and Type Theory*. No. 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999.

[11] JACOBS, B., AND RUTTEN, J. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, No. 62 (1997), 222–259.

[12] LAMBEK, J., AND SCOTT, P.-J. Introduction to higher-order categorical logic. *Studies in Adv. Math, Cambridge University Press*, No. 7 (1986).

[13] MIHÁLYI, D., AND NOVITZKÁ, V. A coalgebra as an intrusion detection system. *Acta Polytechnica Hungarica, Journal of Applied Sciences 7*, 2 (2010). Óbuda University, ISSN 1785-8860.

[14] NIVELA, M. P., AND OREJAS, F. Initial behavior semantics for algebraic specifications. In *Lecture notes in Computer Science on Recent trends in data type specification* (New York, NY, USA, 1987), Springer-Verlag New York, Inc., pp. 184–207.

[15] NOVITZKÁ, V., MIHÁLYI, D., AND VERBOVÁ, A. Coalgebras as models of systems behaviour. In *International Conference on Applied Electrical Engineering and Informatics, Greece, Athens* (2008), pp. 31–36.

[16] SLODIČÁK, V., AND MACKO, P. New approaches in functional programming using algebras and coalgebras. In *European Joint Conferences on Theory and Practice of Software* (2011), Universität des Saarlandes, Saarbrücken. (accepted).

[17] WISBAUER, R. *Algebras versus coalgebras*. University of Düsseldorf, Germany, 2007.