From Coalgebraic Logic to Modal Logic: An Introduction

Novitzká, Valerie, Steingartner, William and Perháč, Ján

Abstract: ***We present how modalities in coalgebraic logic can be introduced to describe behavior of the systems. We start with an analysis of coalgebraic approach together with several simply understandable examples of coalgebras. We concern on observable behaviour of the systems that can be described by coalgebraic formulas. We extend this logic with modalities of necessity and possibility to increase the expressive power of coalgebraic logic. We illustrate our approach on the same examples to compare both approaches.***

Index Terms: *coalgebra, modal logic, behaviour*

1. Introduction

LOGIC is one of the basic sciences used in everyday life and also in many scientific areas. In this work, we will be mostly concerned with coalgebraic and modal logic. Coalgebraic logic [6] is a logic, which main goal is to describe observable behaviour of systems. The foundation of coalgebraic logic is a coalgebra [4]. A coalgebra is dual term to algebra and this relationship reflects the duality between construction of systems (by algebras) and observing outer behaviour of systems (by coalgebras). A coalgebra examines and observes the outer behaviour of the systems. In any system we consider a set of states among which some transition functions are performed. These states are often hidden, not observable by a user. Coalgebra does not care about inner functions or inner structure of the systems. All what it can follow are the outer changes of the system, mainly the observable output values. The work of a system can be described by using coalgebraic logic [2]. This logic can be reformulated to modal logic [3, 9] that increases the expressive power of coalgebraic logic. Modal logic [7] is a one of a group of logical systems, which consists of deontic logic, temporal logic, epistemic logic, standard modal logic, and many others. In this paper we consider standard modal logic. Main attribute of standard modal logic is its modal nature by introducing new operators of neccessity and possibility.

In the next section, we introduce basic notions from category theory and coalgebras to define coalgebraic logic. Then we reformulate this logic adding modality operators. The main contribution of this paper is to illustrate using coalgebraic modal logic on the simple examples of real systems, which make coalgebraic approach and modal logic more understandable also for practical programmers and students that are not familiar in mathematics and logic.

2. Basic Notions

Algebras and coalgebras are modeled using notions of category theory. This discipline of mathematics define and investigate special mathematical structures called categories. It examines properties of objects, relationships between objects inside one category using morphisms and relationships between categories using functors [1]. Coalgebras are structures constructed over base category using appropriate polynomial endofunctors.
A category is a mathematical structure which

Manuscript received March, 2019.

This work was supported by the Slovak Research and Development Agency under the contract No. SK-AT-2017-0012: Semantics technologies for computer science education.

The authors' affiliations: Technical University of Košice, Slovakia, email: valerie.novitzka@tuke.sk, william.steingartner@tuke.sk,jan.perhac@tuke.sk.

consists of

- objects denoted by A, B, C,...
- morphisms between objects denoted by f, g, h,...

Each morphism f is defined between two objects: domain and codomain. A morphism is a function of the form

$$f : A \to B, \tag{1}$$

where A is a domain of the function and B is a codomain of it.

A functor is a morphism between two categories. Assume, we have two categories $\mathbf{C}, \mathbf{D}$. A functor F

$$F : \mathbf{C} \to \mathbf{D} \tag{2}$$

is a morphism between categories preserving the structure, identities and composition. That means, when F maps the objects A, B in a category $\mathbf{C}$ to the objects $F(A), F(B)$ in a category $\mathbf{D}$, then a morphism $f : A \to B$ in the category $\mathbf{C}$ is mapped to a morphism

$$F(f) : F(A) \to F(B)$$

in the category $\mathbf{D}$.
A functor, which we use in this paper is a special type of functors, which has as a domain and codomain the same category. Such functor is called endofunctor

$$F : \mathbf{C} \to \mathbf{C}.$$

The simplest example of an endofunctor over a category is the identity functor $Id : \mathbf{C} \to \mathbf{C}$, which maps each object of a category $\mathbf{C}$ to the same object and every morphism to the same morphism.

Coalgebras are constructed over a base category. This base category has as objects states of a system and morphisms are operations that change states. These operations can not only change the states, but also to produce some output, or to use some input values from the environment. Therefore in [5] coalgebras use polynomial endofunctors of the form that is similar to polynomials known in algebra. For instance the polynomial endofunctor:

$$F(X) = (1 \times X + O \times X)^I.$$

characterizes a system which has a set of states X, a set of outputs O, a set of inputs I and an error state or message 1. This endofunctor represents a type of systems, which can produce either an error message or it is working using input values from I to change the states and possibly to produce output values from O. A state is changed in each step of system execution, that is why we have X in both members of the equation. More formally, 1 describes abnormal end of a system execution, X describes that a state is changed and $O \times X$ expresses a change of a state together with producing some output value. The exponent I describes that every step of execution needs some input value. Each polynomial endofunctor characterizes one kind of the systems.

An algebra is the essential part of mathematics, it deals with sets with operations satisfying certain properties. Induction is logical reasoning principle for algebras and initiality is a property for algebras. Standard algebraic techniques have been proved to be useful in capturing various aspects of data structures in computing. It turned out to be difficult sometimes to describe dynamic structures. We are talking about structures which involve sets od states. As time goes by, it turned out that dynamic systems which involve sets of states will not be described as algebras anymore, but they will be described as coalgebras. Coalgebras [10] are dual concepts to algebras. The logical reasoning principle for coalgebras is coinduction which is a dual notion to induction and also finality is a property for coalgebras which is dual notion to initiality.

A system is a running process which communicates with an environment. Communication between a system and its environment is held by interface. User can describe a certain system exactly by its interface. This outer view of the system is also called black box view. It is called black box, because a user does not know what is exactly going on in the system. All he sees, is the system's behaviour. A system can be abstracted as a set of states X with a transition function ξ. This transition function describes transition for each state $x \in X$:

$$X \xrightarrow{\xi} F(X). \tag{3}$$

F is a polynomial endofunctor, X is a set of states and ξ is a transition function.

When we are starting to observe a certain system, we are expecting that it starts in an initial state x_0. A system including first state is called a process. We write a process as a triple

$$(X, \xi, x_0). \tag{4}$$

We do not always need to write it as a triple if we already know what system we describe. In some case, we write only a sttate space and transition function.

Now, we describe how we can observe a behaviour of the several types of the systems by coalgebras. Assume that we have basic system, which will produce one output value. This value $a \in A$ is from set A. Transition function for this type of system will look like

$$X \xrightarrow{\xi} A, \tag{5}$$

where $F(X) = A$. When this type of system is running, it can execute a transition function only once per run.
Let us assume another type of system which will produce infinite number of output forever.

$$X \xrightarrow{\xi} A \times X. \tag{6}$$

We assume that a system starts in a state x_0 and one of the transition functions is $\xi(x_0) = (a, x_1)$. The next transition function has x_1 as initial state and this will be repeated in the all following steps. This type of system is called a stream.

In the next section we introduce coalgebraic logic for coalgebras.

3. Coalgebraic Logic

Coalgebraic logic is used for specifying properties of classes of coalgebras. Each polynomial endofunctor determines a type of systems, i.e. a class of coalgebras. Coalgebraic logic uses a corresponding polynomial endofunctor, which get an actual state of a system and returns a next state.

A formula is a finite sequence of symbols, which is constructed by exact formal logical language. We use formulas satisfying syntax of propositional logics. We denote formulas by small letters from greek alphabet

$$\varphi, \psi, \phi.$$

The coalgebraic formulas reflect the behaviour of a system, each step of execution provides a new coalgebraic formula. We illustrate coalgebraic logic on the following examples of the simple systems.

Example 1: *Basic calculator*

Our first example is a simple basic calculator. The calculator executes four basic arithmetical operations: addition, subtraction, multiplication and division of some natural numbers.

We construct the category of states $State$, where objects are states and morphisms are operations. The suitable polynomial endofunctor $F : State \to State$ is

$$F(X) = ((E_m + O_m) \times X)^I,$$

where we use the following symbols:

- set E is a set of errors,
- set O is a set of output values,
- set X is a set of states,
- set I is a set of input values,
- m indicates the corresponding executed arithmetical operation.

We are also using categorical operation exponent, because the functor is defined as a function from input values to an output value (or error) for a given operation m.

Let us assume the category $State$ that consists of states $x_0, x_1, ..., x_n$, which are also called objects of the category. A coalgebra is a pair (X, ξ) where X is called a state space and ξ is coalgebraic structure as defined above.

In this example, the state space will be state space of the calculator and the coalgebraic structure gets input values in an actual state and produce a result of an operation together with the change of state.

Consider that we add on calculator two numbers 4 and 5. We construct the coalgebraic formulas as the sequences for each application of the functor F, i.e. for every change of state:

$$\begin{aligned}
\varphi_1 = &\quad ((x_0, \bot), true),\\
\varphi_2 = &\quad ((x_1, 4), ((x_0, \bot), true)),\\
\varphi_3 = &\quad ((x_2, 5), ((x_1, 4), ((x_0, \bot), true))),\\
\varphi_4 = &\quad ((x_3, 9), ((x_2, 5), ((x_1, 4),\\
&\quad ((x_0, \bot), true)))).
\end{aligned}$$

This sequence of coalgebraic formulas describes the whole work of a system in the case of addition of two numbers. The system ends its work in four steps.

Example 2: *Kettle with a jug of water*

The second example, we present, is a kettle with a jug of water. We are heating the jug because we want the water to get in the boiled state.

For this simple system we use the same category of states $State$ but the coalgebraic model differs in a polynomial endofunctor. We define the polynomial endofunctor $F : State \to State$ for this type of system as

$$F(X) = (O \times X)^I,$$

where

- X is a set of water states,
- I is a set of input values (temperature),
- O is a set of output values (boiled or not).

Category $State$ consists of states $x_0, x_1, .., x_n$ and a coalgebra has same structure (X, ξ) like in previous example. In this example we are going to use one more set

$$B = \{true, false\},$$

which has two boolean values indicating boiled state of the water. The formulas for this type of example will look like this

$$\begin{aligned}
\varphi_1 = &\quad ((x_0, \bot), true),\\
\varphi_2 = &\quad ((x_1, false), ((x_0, \bot), true)),\\
\varphi_3 = &\quad ((x_2, false), ((x_1, false),\\
&\quad ((x_0, \bot), true))),\\
\varphi_4 = &\quad ((x_3, true), ((x_2, false),\\
&\quad ((x_1, false), ((x_0, \bot), true)))).
\end{aligned}$$

Here we are heating the jug so the water is getting hotter and hotter. When the temperature gets 100 degrees Celsius, the water is boiled and the value of last state is set to true.

4. Coalgebraic Modal Logic

Modal logic deals with the study of linguistic structures which evaluate the truth conditions of terms, encompassing abstract directions like beliefs, ethics and learning. We can describe this logic as a logic of necessary and possible truths. This logic is called standard modal logic. Nowadays, modal logic is also used in computer science, where it is talking about different behaviours of systems or properties of transitions between states of the systems. Aristotle in the ancient times was the first, who mentioned two important words of modal logic: necessity and possibility. This was the start of formulating of standard modal logic which is a part of a big family of modal logics. Introducing modality operators enables to increase expressive power of logic. Without them we can deal with formulas containing modalities as with unstructured propositions.

In standard modal logic we use two basic modal operators. The operator $\Box$ indicates necessity and the operator $\Diamond$ indicates posibility. A formula

$$\Box\varphi$$

can be read as: it is necessary that φ holds, and a formula

$$\Diamond\varphi$$

can be read as: it is posible that φ holds. These modal operators are dual and satisfy the following equivalences:

$$\begin{aligned}
\Box\varphi &= \neg\Diamond\neg\varphi,\\
\Diamond\varphi &= \neg\Box\neg\varphi.
\end{aligned}$$

In this section we would like to show how modalities can be used on the examples in previous section. To formulate modal coalgebraic logic, we need to use Kripke semantics. Saul Kripke is a famous philosopher who brought a new way of looking on the semantics of modal logic. Kripke's main rule in semantics of modal logic is that we are considering existence of more than one world. The truth of the formulas can be different in different worlds. He considers a set of possible worlds and define the rules when a modal formula is true in a given world.

A Kripke's model is a triple

$$Model = \langle W, \leq, \models \rangle,$$

where

- W is a set of possible worlds,
- $\leq$ is an accessibility relation between worlds,
- $\models$ is a satisfaction relation, i.e. it defines when a formula φ is true in a world w.

Very important thing in this work is to assign elements from coalgebraic logic to elements in modal logic. We consider that any state from the state space X in coalgebraic model corresponds some world $w \in W$, which means

$$x_i \equiv w_i.$$

Accessibility relation will reflect the relationship between two states and we will write this relationship as

$$x_1 \leq x_2,$$

which means state x_2 is accessible from state x_1. Last element of Kripke's model is satisfaction relation. We will write it as

$$x_n \models \varphi_n,$$

which means, formula φ_n is true in state x_n. This relation is exactly defined in Kripke model [8].

Example 3: *Coalgebraic modal logic formulas for calculator*

We formulate coalgebraic modal logic for the basic calculator from Example 1. We work with a more difficult operation than addition. We will create coalgebraic formulas with modalities for division. We need to consider that we can not divide with zero because we will get an error.

The calculator starts its work in the state x_0 and φ_0 is signalizing that calculator is turned on.

$$x_0 \models \Box\varphi_0.$$

We are getting into a next state x_1 where we are entering numbers. Formula φ_1 is true if and only if formula φ_0 is neccessary true. That

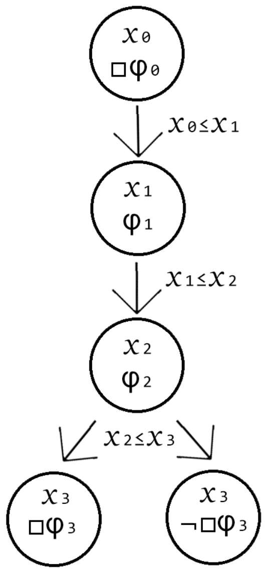

Figure 1: Kripke's model for division

means we can enter numbers only to a turned on calculator. Formula φ_2 is true only if the second entered value is not zero.

$$\text{if } x_1 \models \varphi_1 \wedge x_2 \models \varphi_2 \text{ then } x_3 \models \Box\varphi_3.$$

This means that if formula φ_1 is true in state x_1 and formula φ_2 is true in state x_2 then it is necessary true that formula φ_3 is true in state x_3 which means we have a result.

$$\text{if } x_1 \models \varphi_1 \wedge x_2 \models \neg\varphi_2 \text{ then } x_3 \models \neg\Box\varphi_3.$$

Otherwise, if formula φ_1 is true in state x_1 and formula φ_2 is not true in state x_2 than we it is necessary not true that we have a good result, which logically means we got an error. In Fig.1 we can see a coalgebraic modal Kripke's model for divison which represents coalgebraic formulas with modalities.

□

Example 4: *Coalgebraic modal logic formulas for kettle with a jug of water*

Let's get back to the example with the kettle and jug of water. We added one special formula φ_4 to this case which is signalizing if the kettle is connected to the electricity. Formula is true if it is connected and is false if is not.

We start in the state x_0 with formula φ_0 which is not true here. Formula in this example signalizes that the water is hot. We are in the first state and the water is not hot yet, because we are starting with cold water, the formula is false. As we are heating our water, the temperature is raising and we are getting into state x_1.

$$\text{if } x_0 \leq x_1 \text{ then } x_1 \models \diamond\varphi_1,$$

which means that if state x_1 is accessible from state x_0 then the formula φ_1 is possibly true. In the next state x_2 the water is already hot so we are writting a formula

$$\text{if } x_1 \leq x_2 \text{ a } x_1 \models \diamond\varphi_1 \text{ then } x_2 \models \varphi_2,$$

which means that if state x_2 is accessible from state x_1 and formula φ_1 is possibly true in state x_1 then formula φ_2 is true in state x_2. In the last state x_3 the water gets 100 degrees Celsius and it is in the boiled state.

$$\text{if } x_2 \leq x_3 \text{ a } x_2 \models \varphi_2 \text{ then } x_3 \models \Box\varphi_3.$$

Here we have the next formula which says that if state x_3 is accessible from state x_2 and formula φ_2 is possibly true in state x_2 then φ_3 is necessary true in the state x_3, what means, it is boiled. If we unplug the kettle from the electricity, it gets in state x_4 and this state is accessible from all of the previous worlds (states).

$$x_4 \models \neg\Diamond\varphi_4.$$

This means that if we get into state x_4 formula φ_4 is necessary not true, because the water in this state is not hot for sure.

In Fig.2 we can see Kripke's model for the kettle with jug of water which represents coalgebraic formulas with modalities.

□

5. Conclusion

In our paper we shortly presented the principles of coalgebraic modeling of systems, which enables to model observable behavior. We concerned on logical systems used in coalgebras and introduced the principles of coalgebraic logic. Using the equivalence between possible worlds and state space in Kripke's model, we can describe the behavior of systems by coalgebraic modal logic that has higher expressive

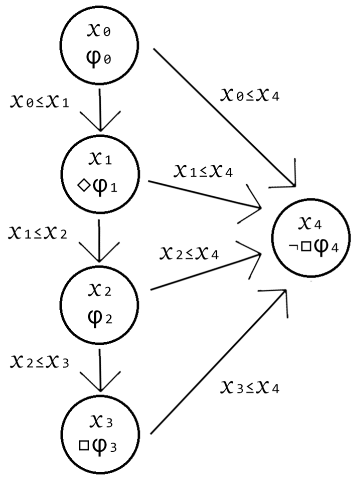

Figure 2: Kripke's model for the kettle with jug of water

power and more readable formulas then coalgebraic logic.

Our approach can be extended also for more complex systems using the methods presented in this paper.

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall International, 1990.

[2] C. Cirstea. *Logics are Coalgebraic*. BCS Visions in Computer Science, 2008.

[3] I. Hasuo. Modal logics for coalgebras. Tokyo Institute of Technology and AIST, 2004.

[4] B. Jacobs. *Introduction to Coalgebra. Towards Mathematics of States and Observations*. Cambridge University Press, 2016.

[5] J. Kock. Data types with symmetries and polynomial functors over grupoids. *Proceedings of the 28th Conference on the Mathematical Foundations of Programming Semantics, Electronic Notes in Theoretical Computer Science*, 286:351–365, 2012.

[6] E. Komendantskaya, J. Power, and M. Schmidt. Coalgebraic logic programming: from semantics to implementation. University of Dundee, 1999.

[7] C. Kupke and D. Pattinson. Coalgebraic semantics of modal logics: An overview. *Theoretical Computer Science*, 412:5070–5094, 2011.

[8] A. Kurz. Coalgebras and modal logic. University of Leicester, 2001.

[9] A. Kurz. Specifying coalgebras with modal logic. *Theoretical Computer Science*, 260:119–138, 2001.

[10] J. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249, 2000.