STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVIII**, Number 4, 2013

SOME ASPECTS ABOUT TOPOS THEORY AND TREE AUTOMATA

WILLIAM STEINGARTNER

Abstract. Category theory provides possibilities to model many important features of computer science. Toposes as specific categories enable to model theories over types. Another means used in computer science for term calculus are tree automata that enable evaluation of well-formed terms in this calculus. Our paper presents how the theory of algebraic structures, namely group theory, can be represented in a topos. We also specify its correspondence with tree automata enclosed in the category of sets as a mapping.

1. Introduction

Computer systems are playing crucial *rôles* in every aspect of human activities. The main idea of computer program for solving scientific problems is to formulate the instructions designed to carry on required computations and to generate the expected behavior [15]. To solve large scientific problems by mathematical machines we always start with with the formulation of their theoretical foundations. We need to formalize these theoretical foundations as logical reasoning of various mathematical theories because the programs should really prove the correctness of their results [6, 20]. A program consists of data structures and algorithm. Data structures always have some types. These types can frequently be very complex structures as algebraic structures, vector spaces, etc. [9, 11], and also in databases [3]. In such cases set theory does not suffice our needs to describe and represent them. Mathematics provides a useful discipline, category theory that enables us to work with the structures of arbitrary complexity and describe their properties and relations between them [21]. Using of category theory in computer science has extremely grown in the last decade [8]. Categorical methods are already well-established for the semantic foundation of type theory, data type specification frameworks

Received by the editors: April 7, 2013.

2010 *Mathematics Subject Classification.* 18B25, 20J99.

1998 *CR Categories and Descriptors.* G.2.2 [**Discrete mathematics**]: Graph theory – *Trees*; F.1.1 [**Computation by abstract devices**]: Models of Computation – *Automata*.

Key words and phrases. Group theory, topos, tree automata, term evaluation.

and graph transformation [8]. Categories are structures which enable to work with objects of arbitrary complexity. Fundamentals of category theory are the relations between objects [5]. These relations are expressed by morphisms. In many applications of category theory, morphisms usually correspond to computations, proofs, operations, etc. It depends on what we are modeling with categories. Very useful kind of categories are toposes [2] that enable to model theories. Toposes are special kind of categories defined by axioms saying roughly that certain constructions one can make with sets can be done in a category. Toposes are important also in theoretical informatics for modeling computations [5, 14, 17]. In our paper we deal with the algebraic structures, namely groups and we show how can be group theory represented in topos.
Automata provide useful means for recognizing languages, but they are very important in other branches [10]. In our approach we give to input symbols of sequential and tree automata some additional information, the types of input symbols and we extend automata to typed sequential and tree automata. Typed sequential automata can be used for type-checking of terms in typed term calculus over a type theory of a given solved problem [13]. On the other hand, typed tree automata enable evaluation of well-formed terms in this calculus. We have shown how typed automata can be depicted in the category $\mathcal{S}et$ of sets and functions [12]. In this paper we define the relation between typed tree automata and groups modeled by categories as a mapping from a topos to the category $\mathcal{S}et$.

Groups are very known algebraic structures that are often used in computer science. They are defined as triples $(G, \odot, u)$, where:

- G is the universum set, and
- $\odot$ is multiplicative operation, which is associative.

Multiplicative operation has a neutral element u so that

$$\forall x \in G.\ x \odot u = u \odot x = x.$$

To each element of the universum set G has to be defined an inverse element over the multiplicative operation with the property:

$$\forall x \in G.\ \exists x^{-1} \in G.\ x \odot x^{-1} = u.$$

Groups can be represented in categories by several ways. The most obvious way is to construct the category of groups $\mathcal{G}rp$, where category objects are groups and category morphisms are group homomorphisms. It is category, because homomorphisms are composable and each object has identity category morphism (identity homomorphism). We prefer another possibility - a representation of groups in a topos, because we would like to represent not only structures, groups, and homomorphisms between them, but the whole theory with the axioms.

Firstly, we shortly describe the group theory and in the next section we express it in a topos. We present the notion of logical theory which was introduced in [2]. The logical theory is a triple $\mathcal{T} = (X, c, \alpha)$, where

- X denotes the basic type;
- c is the list of basic constants;
- α is the list of sentences - the axioms.

The group theory is a triple $(G, [u, m, i], \alpha)$, where

- G is the basic type;
- u, m, i are constants of intended types G, $G^{G\times G}$, G^G respectively;
- α is the list of axioms significant for groups.

A type G^G represents the function space $G \to G$ and a type $G^{G\times G}$ the function space $G\times G \to G$. To ensure the constants u, m, i to have the proper types, we must start with the constant (u, m, i) of the type $G\times \mathcal{P}(G\times G\times G)\times \mathcal{P}(G\times G)$, where $\mathcal{P}$ denotes powerset. We formulate the appropriate axioms.
For groups we have usual axioms for associativity, neutral element and inverse element. The axiom for associativity (here denoted α_1) is

$$\alpha_1 \equiv \forall x, y, z \in G.\ m(x, m(y, z)) = m(m(x, y), z), \tag{1}$$

where $m \in G^{G\times G}$ is binary operation on G, $m : G \times G \to G$.

The axiom of neutral element (α_2) says that there exists an element $u \in G$ (also expressed as a constant or nullary operation on G, $u :\to G$) such that

$$\alpha_2 \equiv \forall x \in G.\ m(x, u) = x \wedge m(u, x) = x. \tag{2}$$

The last one is the axiom of inverse element (α_3). For each x in G there exists an element $i(x)$ (or x^{-1}, also known as unary operation on G, $i : G \to G$;) in G where $i \in G^G$, such that

$$\alpha_3 \equiv \forall x \in G.\ m(x, i(x)) = u \wedge m(i(x), x) = u, \tag{3}$$

where u is a neutral element (also known as *identity element*).
Now we are able to form new axiom,

$$\alpha_4 \equiv (u, m, i) \in G \times G^{G\times G} \times G^G, \tag{4}$$

expressing the properties of constants and include it into axiom α. Thus α consists of the conjunction of previous axioms of associativity, neutral and inverse element and of constants u, m and i; and it is the conjunction of those four sentences,

$$\alpha = \alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4.$$

2. Semantics

For the reader's convenience and to fix the notation we begin by recalling some facts about categories and about toposes as special categories that serve for our purposes. A category $\mathcal{C}$ consists of two classes:

- the class of objects $A, B, \ldots$;
- the class of arrows (or morphisms) denoted $f, g, \ldots$

For an arrow $f : A \to B$ the objects A and B are the domain and codomain, respectively. A homset $Hom(A, B)$ is the set of all arrows with domain A and codomain B in $\mathcal{C}$. The arrows $f : A \to B$ and $g : B \to C$ are composable, their composition $f \circ g = fg : A \to C$ is associative in $\mathcal{C}$ and each object A has identity arrow $id_A : A \to A$. We say that a category $\mathcal{C}$ has a terminal object $\mathbf{1}$ (initial object $\mathbf{0}$) if for every category object A there exists unique arrow $A \to \mathbf{1}$ ($\mathbf{0} \to A$). A *monomorphism* at Fig. 1 (also called a monic morphism or a mono) is a morphism denoted $f : A \rightarrowtail B$ such that

$$g_1 \circ f = g_2 \circ f$$

for all morphisms $g_1, g_2 : X \to A$.

$$\begin{tikzcd} X \arrow[r, shift left, "g_1"] \arrow[r, shift right, "g_2"'] & A \arrow[r, tail, "f"] & B \end{tikzcd}$$

Figure 1. Monomorphism f

A monomorphism in a category generalizes the concept of *injection* in set theory.

2.1. **Toposes.** A *topos* is a special category $\mathcal{E}$ satisfying the following conditions [4, 14]:

1. $\mathcal{E}$ has a *terminal object* $\mathbf{1}$, and for every corner of morphisms $X \to Z \leftarrow Y$ in $\mathcal{E}$ there is a *pullback* P (fibred product) at Fig. 2.
2. $\mathcal{E}$ has a *subobject classifier*: an object (traditionally denoted) Ω with a monomorphism $true : \mathbf{1} \rightarrowtail \Omega$ such that for any monomorphism $m : M \rightarrowtail X$ in $\mathcal{E}$ there is an unique morphism $\chi_m : X \to \Omega$ such that the following diagram is a pullback (Fig. 3).

   The morphism $\chi_m : X \to \Omega$ is called *characteristic* (or *classifying*) *morphism* of the monomorphism $m : M \rightarrowtail X$.
3. $\mathcal{E}$ has *power objects*: for each object X in $\mathcal{E}$, there exists an object Ω^X and a morphism $eval_X : X \times \Omega^X \to \Omega$ with the universal property:

$$\begin{tikzcd}
P \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & Y \arrow[d] \\
X \arrow[r] & Z
\end{tikzcd}$$

Figure 2. Fibred product

$$\begin{tikzcd}
M \arrow[r, dashed] \arrow[d, tail, "m"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbf{1} \arrow[d, tail, "true"] \\
X \arrow[r, dashed, "\chi_m"'] & \Omega
\end{tikzcd}$$

Figure 3. Subobject classifier

for any morphism $f : X \times Y \to \Omega$ in $\mathcal{E}$ there is an unique morphism $\lambda f : Y \to \Omega^X$ such that the diagram at Fig. 4 commutes.

$$\begin{tikzcd}
X \times \Omega^X \arrow[r, "eval_X"] & \Omega \\
X \times Y \arrow[u, "id_X \times \lambda f"] \arrow[ur, "f"']
\end{tikzcd}$$

Figure 4. Truth valuation of the object

2.2. **Models.** Every topos has its own "internal language", also called the Mitchell-Bénabou language [2]. We make use of this language to define the notion of a model of a theory in topos. Let $\mathcal{T} = (X, c, \alpha)$ be a fixed theory with just one basic type X, one basic constant C and one axiom α, and let $\mathcal{E}$ be a topos. The language of theory $\mathcal{T}$ we denote $\mathfrak{L}[X, c]$. We begin by interpreting the language $\mathfrak{L}[X, c]$ in the topos $\mathcal{E}$. Firstly, given any object E in $\mathcal{E}$, we associate to each type symbol X of $\mathfrak{L}[X, c]$ an object X_E of $\mathcal{E}$, its *interpretation* with respect to E, by setting [2]:

$$\begin{aligned} X_E &\longmapsto E \\ P_E &\longmapsto \Omega \\ (PY)_E &\longmapsto \Omega^{Y_E} \\ \big(Y \times Y'\big)_E &\longmapsto Y_E \times Y'_E \end{aligned}$$

where Ω is the subobject classifier of the topos $\mathcal{E}$. Now let C be the type of the constant c, and let

$$e : \mathbf{1} \to C_E$$

be any morphism of topos $\mathcal{E}$ with the indicated domain and codomain. Let τ be a term in language $\mathfrak{L}[X, c]$, with type U and free variable type V. The *interpretation* of τ with respect to the pair (E, e) is a morphism

$$\tau_{(E,e)} : V_E \to U_E$$

of the topos $\mathcal{E}$, defined by induction with rules which can be found in [2].
A model M of the theory $\mathcal{T}$ in $\mathcal{E}$ consists of an object X_{M} and a suitable morphism c_{M} in $\mathcal{E}$, satisfying the axiom α. In other words, a *model* of the theory $\mathcal{T} = (X, c, \alpha)$ in the topos $\mathcal{E}$ is a pair (E, e) such that

$$\alpha_{(E,e)} = \mathit{true} : \mathbf{1} \to \Omega.$$

If $\mathsf{M} = (E, e)$ is such a $\mathcal{T}$-model, let $Z_{\mathsf{M}} =_{def} Z_E$ for each type symbol Z in $\mathfrak{L}[T]$; so in particular $\mathsf{M} = (X_{\mathsf{M}}, c_{\mathsf{M}})$.

It is clear that if $\mathcal{T}' = (X, c, [\alpha_1, \ldots, \alpha_k])$ is a theory with more than one axiom then we can construct a conjunction $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_k$ as a single axiom of $\mathcal{T}'$ and consider the theory $\mathcal{T} = (X, c, \alpha)$ instead of $\mathcal{T}'$. We can simply define a model of $\mathcal{T}'$ to be a model of $\mathcal{T}$ in the sense of preceding definition, because the difference between the two theories is very slight. Similarly, a theory $\mathcal{T}' = (X, (c_1, \ldots, c_m), \alpha)$ with several basic constants can be brought into the desired form $\mathcal{T} = (X, c, \alpha)$ by putting $c = (c_1, \ldots, c_m)$ as m-tuple. Again, the difference between the two theories may not seem to warrant a separate definition of a model of a theory with several basic constants and the resulting distinction between $\mathcal{T}'$-models and $\mathcal{T}$-models.

2.3. **Group theory.** The theory of groups has the language $(G,\ [u, m, i])$, where the constants u, m, i have types G, $G^{G\times G}$, G^G respectively, and axiom α is the conjunction of the usual group axioms. A model $\mathsf{M} = (G_{\mathsf{M}}, [u_{\mathsf{M}}, m_{\mathsf{M}}, i_{\mathsf{M}}])$

of this theory in topos $\mathcal{E}$ thus consists of an object $E = G_{\mathsf{M}}$ of $\mathcal{E}$ with morphisms

$$\begin{aligned} u_{\mathsf{M}} &: 1 \to E, \\ m_{\mathsf{M}} &: 1 \to E^{E\times E} \\ i_{\mathsf{M}} &: 1 \to E^{E} \end{aligned}$$

in $\mathcal{E}$. The latter two correspond by transposition to unique morphisms:

$$\begin{aligned} \mu : E \times E &\to E, \\ \iota : E &\to E. \end{aligned}$$

Since M is a model, $\alpha_{\mathsf{M}} = \mathit{true} : \mathbf{1} \to \Omega$ in $\mathcal{E}$. This condition is easily seen to be equivalent to the statement that Figure 5 imagines a group in a topos $\mathcal{E}$.

$$\begin{tikzcd} E \times E \arrow[r, "\mu"] & E \arrow[r, "\iota"] & E \\ & \mathbf{1} \arrow[u, "u_{\mathsf{M}}"'] & \end{tikzcd}$$

FIGURE 5. Group in a topos

More explicitly, the sentence α, recall, is a conjunction $\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_4$. So clearly $\alpha_{\mathsf{M}} = \mathit{true}$ just if each $(\alpha_i)_{\mathsf{M}} = \mathit{true}$ (for $i = 1, 2, 3, 4$). Let us show by way of example that for, say, α_1 the associativity axiom

$$\forall x, y, z \in G.\ m(x, m(y, z)) = m(m(x, y), z),$$

$(\alpha_1)_{\mathsf{M}} = \mathit{true}$ just if $\mu : E \times E \to E$ is associative.

From the definition of the interpretation of terms, one easily sees that
$(\alpha_1)_{\mathsf{M}} =$

$$\begin{aligned} & (\forall x, y, z \in G.\ m(x, m(y, z)) = m(m(x, y), z))_{\mathsf{M}} \\ = \quad & \forall_{G\times G\times G}.\ (m(x, m(y, z)) = m(m(x, y), z))_{\mathsf{M}} \qquad (5) \\ = \quad & \forall_{G\times G\times G}.\delta_G \, \langle \mu(id_G \times \mu), \mu(\mu \times id_G) \rangle \end{aligned}$$

where $\forall_{G\times G\times G}$ is a quantification over types known from higher-order logic [2].

Now $\mu(id_G \times \mu)$ and $\mu(\mu \times id_G)$ are the two paths round the familiar associativity diagram at Fig. 6.

$$\begin{tikzcd}
G \times G \times G \arrow[r, "\mu \times id_G"] \arrow[d, "id_G \times \mu"'] & G \times G \arrow[d, "\mu"] \\
G \times G \arrow[r, "\mu"'] & G
\end{tikzcd}$$

FIGURE 6. The associativity in topos

But this diagram commutes, i.e. μ is associative, just if

$$\delta_G \langle \mu(id_G \times \mu), \mu(\mu \times id_G) \rangle =$$
$$true_{G\times G\times G} : G \times G \times G \to \Omega,$$

that implies

$$\forall_{G\times G\times G} \langle \mu(id_G \times \mu), \mu(\mu \times id_G) \rangle = true : \mathbf{1} \to \Omega,$$

hence, by (5)

$$(\alpha_1)_{\mathsf{M}} = true : \mathbf{1} \to \Omega.$$

Analogously we can prove the remaining group axioms. Then we can say that group theory $(G, [u, m, i])$ can be represented in a topos $\mathcal{E}$, which objects are types and morphisms are mappings between types.

3. Trees and Tree automata

In this section we briefly introduce basic notions about trees and tree automata working with trees.

3.1. **Trees.** In computer science, a tree is a widely-used data structure that emulates a tree structure with a set of linked nodes. It is a special case of a acyclic directed graph. Each node has zero or more child nodes, which are below it in the tree (by convention, trees grow down). A node that has a child is called the child's parent node (or ancestor node, or superior). Every node has at most one parent. We denote by $\mathbb{N}$ the set of positive integers. We denote the set of finite strings over $\mathbb{N}$ by $\mathbb{N}^*$. The empty string is denoted by ε. A *ranked alphabet* is a couple $\Sigma = (F, Arity)$, where F is a finite set of function symbols and *Arity* is a mapping from F into $\mathbb{N}$. The arity of a symbol $\sigma \in F$ is $Arity(\sigma)$. The set of symbols of arity p is denoted by F_p. Elements of arity $0, 1, \ldots, p$ are respectively called constants, unary, ..., p-ary symbols, respectively. We assume that F contains at least one constant.

Let X be the set of variables. We assume that sets X and F_0 are disjoint. The set $T(F,X)$ of terms over ranked alphabet Σ with variables from X is the smallest set satisfying the following requirements:

- $F_0 \subseteq T(F,X)$ and
- $X \subseteq T(F,X)$ and
- if $p \geq 1$, $\sigma \in F_p$ and $t_1,\ldots,t_p \in T(F,X)$, then $\sigma\,(t_1,\ldots,t_p) \in T(F,X)$.

3.2. **Tree automata.** Tree automata are devices which treat the labeled trees analogously as sequential automata handle sequences (words) of input symbols. The internal structure of a sequential automaton is an unary algebra; for a tree automaton it is an algebra of an arbitrary type [1]. As *finitary type* we usually take a ranked alphabet: the finitary set of operations and the arity map. A Σ-tree automaton is a sixtuple $A = (Q, \{\delta_\sigma\}_{\sigma\in\Sigma}, \Gamma, \gamma, I, \lambda)$ where:

- Q is set of states;
- $\delta_\sigma : Q^n \to Q$ are operations Q;
- Γ is an output alphabet; $\gamma : Q \to \Gamma$ is an output map; I is set of variables; $\lambda : I \to Q$ is an initialization function.

The external behavior of the automaton A is expressed by the map β assigning to each tree t the value $\beta(t)$ from Γ which returns after the computation of t.

Example 1. Assume a set $\mathbb{Z}_4 = \{0,1,2,3\}$ and two operations: constant $\tau = 1$ and operation $\odot$ = multiplication modulo 4. Let $\Sigma : F = F_0 \cup F_2$, where $F_0 = \{1\}$ and $F_2 = \{\odot\}$ and let $(\mathbb{Z}_4, \odot)$ be the multiplicative Σ-algebra of integers. Put $\Gamma = \{0,1\}$, and let γ be the parity map:

$$\gamma(z) = \begin{cases} 0 & \text{if } z \text{ is odd;} \\ 1 & \text{if } z \text{ is even.} \end{cases}$$

Then we can construct a Σ-tree automaton $A = (\mathbb{Z}_4, \{1,\odot\}, \Gamma, \gamma, \{x,y\}, \lambda)$ with the initial assignment $\lambda\,(x) = 3$ and $\lambda\,(y) = 2$. The "action" of this automaton consists of taking any binary tree with leaves x and y, computing the tree and giving an output [1, 7]. For example, the tree at Fig. 7 has the computational sequence depicted at Fig. 8.
We used operation $\odot$ for multiplication modulo 4. The result of computation of the tree is 2, resulting output is $\gamma(2) = 1$. □

3.3. **Tree automata in category.** The category $\mathcal{S}et$ of sets has sets as objects and functions between sets as morphisms. Now we represent Σ-tree automata in $\mathcal{S}et$. Assume, for simplicity, that Σ has just one binary operation. Then a Σ-tree automaton is a diagram in Fig. 9 [1].
The sets Q, I, Γ and Q^n (where n is arity) are objects, and functions δ, λ, γ are morphisms in category $\mathcal{S}et$. So category of sets is able to represent any

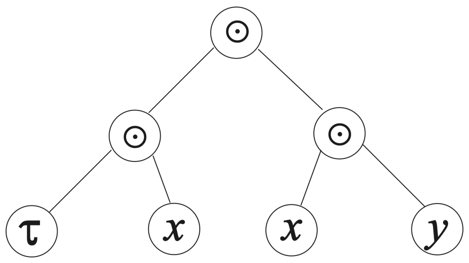

Figure 7. Example of tree

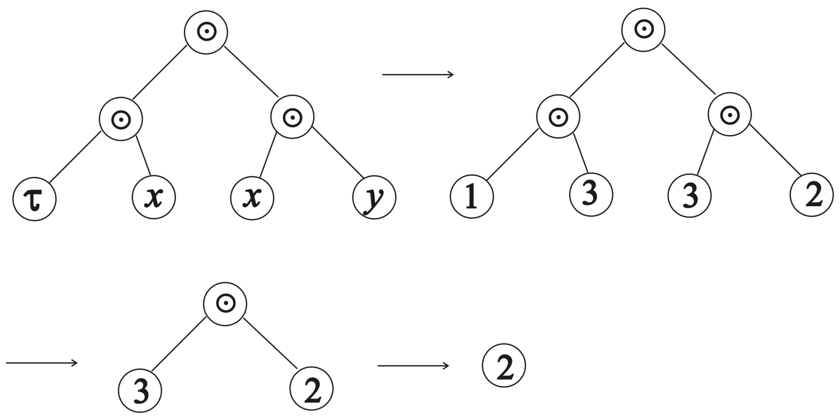

Figure 8. Computational sequence of the tree

$$\begin{tikzcd}
 & I \arrow[d, "\lambda"] & \\
Q^n \arrow[r, "\{\delta\}"] & Q \arrow[r, "\gamma"] & \Gamma
\end{tikzcd}$$

Figure 9. The diagram of Σ-tree automaton in a category of sets

tree automata. Our goal is to assign to any tree automaton the group over which is automaton able to make computations with trees. We will construct this assignment as a mapping.

4. Relation between tree automata and groups

Our goal is to express the relation between tree automata and mathematical structures - groups. Model M of logical theory from chapter 3.3 maps an universum set G into object G_{M} of topos $\mathcal{E}$. This object we can denote also E, *i.e.* $G_{\mathsf{M}} = E$. Objects of our topos $\mathcal{E}$ are images of universum sets. As objects we have products of objects ($E \times E$, because topos is cartesian closed category [4]), and special objects 1 and Ω (traditionally). Let morphisms of our topos $\mathcal{E}$ represent the operations over universum sets, *i.e.* they are images of operations. So we have morphisms that are images of nullary, unary and binary operations (in model M).
All that we said we can list in Tab. 1.

Operation	Operation in group	Representation in topos
multiplicative operation	$m(x,y)$	m_{M}
inverse operation	$i(x)$	i_{M}
neutral element	u	u_{M}

Table 1. Correspondence between group operations and their representations in topos

Mappings m and i correspond by transposition to unique morphisms μ and ι from chapter 3.3 [2]. Now we can formulate assumptions for the automaton A:

- we need to represent values from universum set by states in which automaton operates;
- the set of automaton operations contains just one binary and one unary operation;
- a nullary operation is a constant.

We are able to find elements with similar properties and behavior over groups. Let for each object E of topos (of groups) we have such a set of states Q that automaton working with values of universum set (depicted in object E) can reach states from Q. For morphisms of topos we assign corresponding automaton mappings - according to Table 1 - in category of sets. Let $\delta^{(i)}$ be the operation from automaton with arity i. Let A_0 be the simplified automaton of A such that $A_0 = (Q, \{\delta\})$. We identify it as a *computing part* of automaton A. Now we construct the correspondence (Tab. 2).

as a mapping

$$\mathcal{U} : \mathcal{E} \to \mathcal{S}et.$$

	In topos $\mathcal{E}$	In category $\mathcal{S}et$
Objects	$G_{\mathsf{M}} = E$	sets of states
Morphisms	$u_{\mathsf{M}} : \mathbf{1} \to E$	$\delta^{(0)} : \mathbf{1} \to Q$
	$\iota : E \to E$	$\delta^{(1)} : Q \to Q$
	$\mu : E \times E \to E$	$\delta^{(2)} : Q \times Q \to Q$
Identity	$id_E : E \to E$	$id_Q : Q \to Q$

Table 2. Correspondence between groups in toposes and automata in category of sets

whereby

$$\mathcal{U} : (E, \{\mu, \iota, u_{\mathsf{M}}\}) \mapsto \left(Q, \left\{\delta^{(i)}\right\} |_{i \in \{0,1,2\}}\right)$$

where $\left(Q, \left\{\delta^{(i)}\right\} |_{i \in \{0,1,2\}}\right)$ is a substructure of the automaton A. By extending of this structure with initial variables, initialization and output alphabet we obtain complete automaton.

5. Conclusion

In our paper we have presented how can be a theory of algebraic structures, namely group theory, represented in a topos. We showed how tree automata can be depicted in the category of sets and we defined correspondence between groups iand tree automata as a morphism. Our approach integrates algebraic structures and tree automata into categorical environment. We can treat in similar manner also more complex structures that are important in our research, *e.g.* for type theory and logical system constructed over it as categorical fibration.

Acknowledgement

This work has been supported by KEGA grant project No. 050TUKE-4/2012: "Application of Virtual Reality Technologies in Teaching Formal Methods".

References

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

[2] Awodey, S. *Logic In Topoi: Functorial semantics for higher-order logic.* PhD thesis, The University of Chicago, Chicago, IL, March 1997.

[3] Aleksić, S., Ristić, S., Luković, I., Čeliković, M. A Design Specification and a Server Implementation of the Inverse Referential Integrity Constraints. In *Computer Science and Information Systems*, Vol. 10, No. 1, 283-320, (2013).

[4] Barr, M., Wells, C. *Toposes, Triples and Theories.* Springer-Verlag, 2002.

[5] Barr M., Wells C. *Category Theory for Computing Science.* Prentice Hall International, 1990. ISBN 0-13-120486-6.

[6] Blute, R., and Scott, P. *Category theory for linear logicians.* T.Erhard, J.-Y.Girard, P.Ruet (eds.): Linear Logic in Computer Science. London Mathematical Society Lecture Note Series, Cambridge Univ.Press, 2004.

[7] Comon, H. *Tree Automata Techniques and Applications.* free book, Sept 2005.

[8] 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.

[9] Gladišová, I. Trellis coding of multidimesional signals. In *AEI 2011: International conference on applied electrical engineering and informatics 2011* Sept 2011, Italy, Technical University of Košice, 2011, pp. 32–35.

[10] Gladišová, I. - Kocur, D. SC-FDMA: Základné princípy. In *Electrical Engineering and Informatics : Proceeding of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice*, (Sept 2010), Košice, Slovak Republic, pp. 765–769.

[11] Jakubčo, P. and Šimoňák S. Petri Net approach for algorithms design and implementation. Acta Electrotechnica et Informatica, Vol.11, No.4, 2011, ISSN 1335-8243, pp. 46–51

[12] Novitzká, V., Slodičák, V., Mihályi, D. Tree automata in the mathematical theory. In *SAMI 2007 Proceedings, 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics, Poprad - Aquacity* (2007), pp. 447–456.

[13] Novitzká, V., Samuelis, L., and Slodičák, V. Type theory and typed finite automata. In *Journal of Information, Control and Management Systems*, Vol. 5, No. 1 (2007), pp. 91–100, ISSN 1336-1716.

[14] Phoa, W. *An introduction to fibrations, topos theory, the effective topos and modest sets.* The University of Edinburgh, 2006.

[15] Reichel, H. Behavioural equivalence - a unifying concept for initial and final specification methods. In *3rd Hungarian Computer Science Conference* (1981), no. 3, Akadémia kiadó, pp. 27–39.

[16] Slodičák, V. Tree automata in the rôle of evaluating in categorical terms. In *Pietriková, A., Slodičák, V., Főző, L. editors: Proceedings from 7th PhD Student Conference and Scientific and Technical Competition of Students of Faculty of Electrical Engineering and Informatics Technical University of Košice* (2007), Košice, pp. 120–121, ISBN 978-80-8073-803-7.

[17] Slodičák, V. and Novitzká, V. Coalgebraic Approach for Program Behavior in Comonads over Toposes. *Studia Universita Babeş-Bolyai, Series Informatica* Vol. 50, No. 2, 2010, pp. 15–26, ISSN 2065-9601.

[18] Slodičák, V. and Macko, P. New approaches in functional programming using algebras and coalgebras. In *European Joint Conferrences on Theory and Practise of Software - ETAPS 2011* (March 2011), Universität des Saarlandes, Saarbrücken, Germany, pp. 13–23. ISBN 978-963-284-188-5.

[19] Slodičák, V. Some useful structures for categorical approach for program behavior. *Journal of Information and Organizational Sciences Vol. 35*, No. 1 (2011), University of Zagreb, Varaždin, Croatia, ISSN 1846-9418.

[20] Slodičák, V. and Macko, P. Some New Approaches in Functional Programming Using Algebras and Coalgebras. Electronic Notes in Theoretical Computer Science (ENTCS), Volume 279, Issue 3, pages 41–62, Elsevier, ISSN 1571-0661.

[21] Steingartner, W. The rôle of categorical structures in integral calculus. In: CSE 2012 : International Scientific Conference on Computer Science and Engineering (Proceedings), October 3rd-5th, (2012), Stará Lesná, High Tatras, Slovakia. pp. 24-30, ISBN 978-80-8143-049-9.

Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovak Republic

E-mail address: `william.steingartner@tuke.sk`