Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 119 (2017) 16–21

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Young Scientists Conference in HPC and Simulation, YSC 2017, 1-3 November 2017, Kotka, Finland

Two approaches to System-of-Systems from Lative Logic point of view

Patrik Eklund[a], Jari Kortelainen[b,*]

[a] *Umeå University, 90187 Umeå, Sweden*
[b] *South-Eastern Finland University of Applied Sciences, 50190 Mikkeli, Finland*

Abstract

The paper presents two approaches to model System-of-Systems on lative logic point of view. Lative logic is a general framework to construct building blocks of logic using Category Theory as its metalanguage. This approach reveals avenues to describe System-of-Systems themselves, and to model information and processes they posess, using some reasonable modelling languages in a computational manner, thus, touching foundations of computational science. After presenting some preliminary notes, the paper explains the main steps to construct lative logics, and then give two approaches to System-of-System modelling. Finally, the paper presents a survey to some applications.

Peer-review under responsibility of the scientific committee of the 6th International Young Scientist conference in HPC and Simulation

Keywords: Category theory; computational science; lative logic; System-of-Systems

1. Introduction

Computational Science is a wide approach to computational methods, usually mathematical methods implemented with some programming language, to understand, explain and analyse 'complex systems'. As it is well known, a *system* is a collection of objects with certain interactions, connections between objects. Additionally, a *System-of-Systems* (SoS) is a collection of systems with interactions, connections between systems. Is it then the case that one could think SoS just as a one system? The answer is 'yes' if the SoS consists of systems with no capability to make independent decisions (but they may be able to make decisions!), but the answer is 'no' if SoS consists of systems which independent from each other interact, exchange information and make decisions. We show in this paper that for a SoS modelling there is indeed two approaches basing on (mathematical) lative logic (cf. [4]).

What are then 'complex systems'? The authors think that in many cases 'complex systems' could be understood as SoS and, 'complexity' posesses the understanding of these two previously mentioned approaches to SoS. How

* Corresponding author. Tel.: +358-40-6550555.
E-mail address: jari.kortelainen@xamk.fi

1877-0509
Peer-review under responsibility of the scientific committee of the 6th International Young Scientist conference in HPC and Simulation
10.1016/j.procs.2017.11.155

computational science is taking account which kind of information and processes really produces the action, output, or behaviour of SoS when input is given? Is it really possible to have a reasonable model by using and studying *only* input-output data? The authors believe that a reasonable computational model of SoS needs also understanding about internal information processes of each system which often incorporate *operations* which, performed sequentially to some input of different *types*, form then *terms*. Some systems have an interesting internal property (roughly speaking): they produce *sentences* from terms, and have learned, perhaps during time, rules (sentences) acting as *axioms* defined by some given *entailment relation*. As a decision-making capability, there must be also *proof rules* such the system 'knows' which kind of behavior is *valid in the sense of given entailment*. In other words, many systems have internal *logic*.

The case is then as follows: A collection of systems may have a supersystem, which has a logic controlling the general actions of the SoS and the systems themselves still may have own logics (see e.g. [10, 6]). Or a collection of systems may have reasonable translations in such a way that one system is capable to 'understand' other systems logics. The paper introduces the latter approach, and the authors think that this approach is important when building *Internet-of-Things* (IoT) applications, for example. It is obligatory to mention here, that these two approaches are tightly bounded by the information and processes the systems posess. It is then natural that information and process standards appearing in the OMG (Object Management Group), languages like BPMN (Business Process Modeling Notation), SysML (Systems Modeling Language), etc., may be used to describe logics in the systems. Or other way around, our approach to logic gives a natural framework to Process Modeling Languages.

The paper is organized as follows: In Section 2 we present some categorical concepts (cf. [1, 11]) for convenience of the readers, and then in Section 3 we present the main building blocks lative logic (cf. [4, 10, 3, 5]). In Section 4 we propose a categorical framework to lative logics and in Section 5 we give some applications ([7, 6]).

2. Preliminary Notes

In this section we recall some preliminary category theoretical concepts for convenience of the readers. The interested readers are recommended to get familiar with [1] for basic concepts of Category Theory and then [11] for a more advanced study, for example, on monoidal categories as an enrichment of category theory.

Mathematics is widely used in computational sciences as a precise meta-language when modeling real world phenomena, and then precise mathematical descriptions enable to create a computer simulation by means of some reasonable programming language. What is then "Mathematics as a precise language"? Quite often one takes *Set Theory* as a meta-language when a modeling task is in hand. As it is well-known, Set Theory is not just a one unique theory: depending on adopted axioms one gets different kinds of theories. It is also the case that one then focuses on *sets as objects*. Indeed, relationships between sets are functions or other type of relations between sets, but in Set Theory they are sets themselves. Is this language rich enough when we have 'a complex system' in hand? Is this language capable enough when describing and studying (mathematical) logic, like algebras (as equational logics)?

Category Theory is built on a certain set theory (cf. [1]) which accepts sets as *small classes*, but also large classes called *proper classes* like 'the collection of all sets'. Moreover, Category Theory focuses not only on objects (like sets) but also on 'connectors' between objects, called *morphisms*. It should be mentioned now that a morphism may not be a function or other type of relation between sets.

A *category* C (see e.g. [1]) consists of the following data: $\mathrm{Ob}(\mathsf{C})$ is the class of *objects of* C. For any two objects $A, B \in \mathrm{Ob}(\mathsf{C})$, $\hom(A,B)$ is the set of C-*morphisms between A and B*, sometimes written also as $\hom_{\mathsf{C}}(A,B)$. A morphism $f \in \hom(A,B)$ is denoted by $f\colon A \to B$ or (preferably) $A \xrightarrow{f} B$. For any object $A \in \mathrm{Ob}(\mathsf{C})$, there is the identity morphism $\mathrm{id}_A \in \hom(A,A)$, which serves as identity with respect to $\circ$, the *composition of morphisms*. The composition of morphisms $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ is denoted by $A \xrightarrow{g\circ f} C$ and the composition satisfies the associativity law. One may now study properties of categories by defining categorical constructions like *products* and *coproducts*, etc.

Example 1. ([1])

1. The category `Set` has all sets as its objects and for any two sets A, and B, $\hom(A,B)$ is the set of all functions between A and B. The composition of morpisms is the composition of functions. Note that `Set` has products

as cartesian products of sets and coproducts as disjoint unions of sets. Because one may construct products and coproducts for any collection of sets, Set is *complete* and *cocomplete*.

2. The category Mat has all positive integers as its objects, and for any two positive integer m and n, $\hom(m,n)$ is the set of all $m \times n$ real matrices. The composition of morphisms $\mathbf{M}$ and $\mathbf{N}$ is the matrix multiplication, thus, $\mathbf{M} \circ \mathbf{N} = \mathbf{NM}$. Note that Mat is a *small* category since Ob(Mat) is a set.

A functor F *between categories* C *and* D is a function between classes of objects, $\mathsf{F}\colon \mathrm{Ob}(\mathtt{C}) \to \mathrm{Ob}(\mathtt{D})$, and a function between sets of morphisms such that we assign $\mathsf{F}(A \xrightarrow{f} B) = \mathsf{F}A \xrightarrow{\mathsf{F}f} \mathsf{F}B$. Composition of functors $\mathsf{F}\colon \mathtt{C} \to \mathtt{D}$ and $\mathsf{G}\colon \mathtt{D} \to \mathtt{E}$ is a functor $\mathsf{GF}\colon \mathtt{C} \to \mathtt{E}$. *A product category* $\mathtt{C} \times \mathtt{D}$ has pairs $(A, B) \in \mathrm{Ob}(C) \times \mathrm{Ob}(\mathtt{D})$ as its objects and pairs (f, g) as morphisms between pairs (A_1, B_1) and (A_2, B_2) in a natural way. *A bifunctor on* C is a functor $\mathtt{C} \times \mathtt{C} \to \mathtt{C}$. *A natural transformation* τ between functors F and G, where $\mathsf{F}, \mathsf{G}\colon \mathtt{C} \to \mathtt{D}$, denoted by $\tau\colon \mathsf{F} \to \mathsf{G}$, assigns for each C-object X a D-morphism $\mathsf{F}X \xrightarrow{\tau_X} \mathsf{G}X$ such that for all C-morphisms $X \xrightarrow{f} Y$, $\mathsf{G}f \circ \tau_X = \tau_Y \circ \mathsf{F}f$. *A monad* $\mathbf{F}$ is a triple (F, η, μ), where natural transformations $\eta\colon \mathsf{id} \to \mathsf{F}$ and $\mu\colon \mathsf{FF} \to \mathsf{F}$ satisfy certain laws in a way that $\mathbf{F}$ may be considered as an *algebraic theory* (see for example [12]). *A monoidal category* C consists of a category $\mathtt{C}_0$ and a bifunctor $\otimes$, called *tensor product*, on $\mathtt{C}_0$ with some natural isomorphisms satisfying certain laws (for details see for example [11]). We adopt the power notation A^n, $A \in \mathrm{Ob}(\mathtt{C}_0)$, in a natural way.

In the light of Example 1, the notion of 'morphism' gives more freedom, flexibility to create models, especially, if the actual programming language supports categorical thinking. Moreover, it is interesting that one can describe uncertain or unprecise phenomena using Category Theory. Indeed, Category Theory gives a suitable metalanguage to study, for example, term constructions in unprecise environments (cf. [3]) or, generally speaking, *monoidal biclosed and cocomplete categories* (for details see [11, 5]. Thus, when modeling complex systems or System-of-Systems Category Theory seem to serve as a suitable metalanguage.

3. Lative Logic and System-of-Systems Applications

In this section we first recall what is meant by lative logic (see e.g. [4, 2, 6] and then concentrate on two views to System-of-Systems applications.

As it is folklore, *logic* from syntactical point of view is a structure containing a *signature*, which consists *sorts* (or types) and *operators*. By choosing a signature and some set of variables, one constructs *terms* and *sentences*. One needs also *theoremata* (structured sets of sentences) to have a relation between theoremata and sentences, that is, *entailment* $\vdash$. Then, from semantical point of view, one has *algebras* (models), where equations of expressions (in this algebra) model sentences constructed from terms. *Satisfaction* $\Vdash$ in the current algebra model entailment. Then, one has (syntactical) *axioms* and *theories*, and syntactical proof rules, *proof calculi*. These should correspond with laws and propositions in the algebra that serves as a model. Usually, these constructions are given using Set Theory as a metalanguage and, because of lack of suitable tools, terms and sentences are constructed by writings in some *natural language*. In this context one often has self-referentiality such that sentences are considered again as terms.

Example 2. In first order logic (predicate logic) one has operators, like $\neg$, and also quantifiers, like $\exists$. Having then a variable x, one has a term $\neg x$. Quite often quantified terms are understood as sentences, so $(\exists x)\neg x$ is a sentence, but one further writes $\neg(\exists x)\neg x$.

By *lative logic* the authors mean *disable self-referentiality* when performing constructions of a logic, such that in a biclosed monoidal and cocomplete category C there are objects S and Ω which abstractly are sorts and operators, there is then a signature Σ over C, which is constructed by means of S and Ω satisfying certain properties (see [3, 5]). Then, one has a *term functor* $\mathsf{T}_\Sigma\colon (\mathtt{C}_0)_S \to (\mathtt{C}_0)_S$ which constructs all terms, technically $(\mathtt{C}_0)_S$ is a product category where each object is a system of objects modeling variables of different sorts. Note that T_Σ is also a monad $\mathbf{T}_\Sigma = (\mathsf{T}_\Sigma, \eta, \mu)$ and the state-of-art constructions are given in [5]. One may also generalize the concept of terms using some other monads composable with $\mathbf{T}_\Sigma$ (in a well-known way) like the well-known conventional powerset monad or many-valued powerset monad (see [12]). However, to produce 'random terms' with Giry monad (see [8]) should be studied in the future.

When all terms are constructed, one has a *sentence functor* $\mathsf{S}\colon (\mathtt{C}_0)_S \to \mathtt{C}_0$. Indeed, the authors' approach is that sentences are unsorted (or is of its own 'sort'). So, ST_Σ is a functor which constructs all sentences from variables

objects. On sentences one may construct theoremata using a functor $\mathsf{P}\colon \mathsf{C}_0 \to \mathsf{C}_0$, which abstractly models structured sets. For example in $\mathtt{Set}_S$ we have an object $X = (X_s)_{s\in S}$, sets of variables of different sorts, and then $\mathsf{PST}_\Sigma X$ is a set of all theoremata: if P is the *powerset functor* then we have a conventional powerset of sentences (as we have for example in Propositional Logic), or if $\mathsf{P} = \mathsf{L}$, the many-valued powerset functor, then we have a set of all L-sets. In this context we point out again, that the working category may be indeed any monoidal biclosed and cocomplete category (like the Goguen category [9]). After constructing theoremata it is possible to construct abstractly an entailment such that $\vdash$ is a *subobject* of $(\mathsf{PST}_\Sigma X) \otimes (\mathsf{ST}_\Sigma X)$, and other structures needed to build a logic may be considered in a similar manner (for more details see [10]). It is now clear that on categorical point of logic $\mathcal{L}$ is a structure over a monoidal biclosed and cocomplete category C. The advance of this approach is that using functors as described above one cannot have self-referentiality as it can be seen in Example 2.

System as a supersystem of systems [10, 6]. Suppose all the systems are such that it is reasonable to consider one signature Σ over a monoidal biclosed and cocomplete category C. The sorts object S is modelling all *data types* used in the systems and operators object Ω models all actions to be performed on the *information* produced in each *process*. Thus, information and process standards are possible to incorporate as modelling tools. It is the variables object X which is considered to carry data in the process, thus, X is understood to serve as *data objects*. Indeed, actions (operations) are performed on X and, when data flows in the system, on $\mathsf{T}_\Sigma X$ too. Naturally, the whole logic structure $\mathcal{L}$ may be given, thus, the logic of the supersystem.

Practically, each system uses only those sorts, operators and data objects which are typical to the system in hand. Thus, each system i has then a signature Σ_i which is a subobject of Σ. It is now interesting to find out that each system i may (or may not) have its own logic structure $\mathcal{L}_i$ which *may not be considered as a sublogic of* $\mathcal{L}$.

How information is then transferred from the system i to the system j? In logic we have a possibility to *substitute variables by terms*. So, consider system i with variables object X^i and a system j with variable objects X^j. Because $X^i, X^j \in \mathrm{Ob}(\mathsf{C}_0)$, we have a *variable substitution* σ given in a supersystem, a morphism $X^i \xrightarrow{\sigma} \mathsf{T}_\Sigma X^j$. Because we have a monad $\mathbf{T}_\Sigma = (\mathsf{T}_\Sigma, \eta, \mu)$, we can then extend substitution to terms, thus, we have a morphism $\mathsf{T}_\Sigma X^i \xrightarrow{\mu_X \circ \mathsf{T}_\Sigma \sigma} \mathsf{T}_\Sigma X^j$. It should be noted that to transfer information between systems in this way is a property of the supersystem, and it is the supersystem which in this sense is responsible to handle information transfer between systems.

System as Independent interacting systems. Let C be a small monoidal biclosed and cocomplete category, thus, C_0 is a small category. It is then known (see for example [11]) that $\mathtt{Moncat}$, objects are all small monoidal biclosed and cocomplete categories and morphisms $\mathsf{C} \xrightarrow{\mathsf{F}} \mathsf{C}'$ are functors satifsying for all $A, B \in \mathrm{Ob}(\mathsf{C}_0)$ and $f, g \in \hom_{\mathsf{C}_0}(A, B)$, $\mathsf{F}(A \otimes B) = \mathsf{F}A \otimes \mathsf{F}B$ and $\mathsf{F}(f \otimes g) = \mathsf{F}f \otimes \mathsf{F}g$, is a category. Note that we write the two (perhaps different) tensor products by the same symbol $\otimes$, for simplicity.

Consider there are now two systems which have sorts, S and S', of their own in the corresponding categories. We define a category $\mathtt{Sort}$ such that objects are all pairs (C, S), where $\mathsf{C} \in \mathrm{Ob}(\mathtt{Moncat})$ and $S \in \mathrm{Ob}(\mathsf{C}_0)$. Morphisms between objects (C, S) and (C', S') are $(\mathsf{C}, S) \xrightarrow{(\mathsf{F}, m')} (\mathsf{C}', S')$ such that F is a morphism in $\mathtt{Moncat}$ and $\mathsf{F}S \xrightarrow{m'} S'$. The composition of the morphisms $(\mathsf{C}, S) \xrightarrow{(\mathsf{F}, m')} (\mathsf{C}', S')$ and $(\mathsf{C}', S') \xrightarrow{(\mathsf{G}, m'')} (\mathsf{C}'', S'')$ is given by

$$(\mathsf{G}, m'') \circ (\mathsf{F}, m') = (\mathsf{GF}, m'' \circ \mathsf{G}m'). \tag{1}$$

Proposition 1. *The composition given in Equation* (1) *is associative and the pair* $(\mathsf{id}_\mathsf{C}, id_S)$ *serves as the identity morphism.*

Then, one may have in two different systems operators objects Ω and Ω' in the corresponding categories. *A signature over* C (or C-signature) is a triple $\Sigma = (\Omega, t, S)$, where $\Omega \xrightarrow{t} \widehat{S} \otimes S$. The object $\widehat{S} \otimes S$, where $\widehat{S} = \coprod_{n\in\mathbb{N}} S^n$ is the coproduct of objects S^n, $n \in \mathbb{N}$, serves as sorts for different kinds of operators and Σ satisfies certain laws (see [3] for details). Between two signatures over the same category one has also *signature morphisms* as pairs $(s, o)\colon \Sigma_1 \to \Sigma_2$ such that morphisms $S_1 \xrightarrow{s} S_2$ and $\Omega_1 \xrightarrow{o} \Omega_2$ satisfies certain laws (see [3]).

As mentioned earlier, the category $(\mathsf{C}_0)_S$ is technically a product category. Indeed, if $\mathbf{1}_\mathsf{C}$ is the unit object of C and $\mathsf{S} = \hom(\mathbf{1}_\mathsf{C}, S)$ then we consider $X = (X_\mathsf{s})_{\mathsf{s}\in\mathsf{S}}$ as an object of $(\mathsf{C}_\emptyset)_S$, and we have similarly for morphisms [3].

We are now able to define a category `GenTerm` (generalized terms) such that objects are pairs $\mathcal{T} = (\mathcal{S}, \mathbf{M}_\Sigma)$, where $\mathcal{S} = (\mathsf{C}, S)$ is an object of `Sort` and $\mathbf{M}_\Sigma$ is a monad over $(\mathsf{C}_\emptyset)_S$, $\mathbf{M}_\Sigma = \mathbf{M} \bullet \mathbf{T}_\Sigma$ where $\bullet$ is the *monad composition* and $\mathbf{T}_\Sigma$ is the term monad. Morphisms between objects $\mathcal{T}$ and $\mathcal{T}'$ are pairs $((\mathsf{F}, m'), \tau)$, where $\mathsf{F}(\widehat{S}) = \widehat{\mathsf{F}S}$, $\mathsf{F}\mathbf{1}_\mathsf{C} = \mathbf{1}_{\mathsf{C}'}$, F is an injection $\hom(\mathbf{1}, S) \to \hom(\mathbf{1}_{\mathsf{C}'}, S')$ and m' is a *monomorphism*. Note that this guarantees, that the functor $\mathsf{F}\colon \mathsf{C}_0 \to \mathsf{C}'_0$ can be extended to $\mathsf{F}^\star\colon (\mathsf{C}_0)_S \to (\mathsf{C}'_0)_{S'}$, $\mathsf{F}X_\mathsf{s} = Y_{m'\circ\mathsf{F}\mathsf{s}}$ or initial object of C' if there is no preimage for some $\mathsf{t} \in \mathsf{S}'$. Finally, τ is a natural transformation $\tau\colon \mathsf{F}^\star\mathsf{M}_\Sigma \to \mathsf{M}'_{\Sigma'}\mathsf{F}^\star$. Composition of `GenTerms`-morphisms is given componentwise.

4. Applications

Our approach to logic connected to information and processes of System-of-Systems seems to be fruitful on application point of view. This section recalls briefly two very different kinds of applications using 'system as a supersystem of systems' approach.

We have already mentioned that OMG standards may be used as modeling tools. In fact, the underlying logic of BPMN may be given as described in 'system of supersystem of systems' pragraph. BPMN's syntax enables data flow with underlying tokens, which in our approach can be seen as substitutions. This is applied to social and health care such that variables are Data Objects as *structured documents* (see Chapter 6 in [10] for more details) and to energy market modeling (see [6]).

Another application area of 'system as a supersystem of systems' approach is the logical extension of design structure described as a matrix (DSM). Conventionally, components, people and activities in DSM are simply names, so they are logically just constants in the underlying signature of the logic, i.e., there are no operators with arity beyond zero to describe components as terms. This means that task volatility in DSM is connected only with information variability and likelihood, so DSM is not logical but statistical with respect to modeling rework and redesign. To apply many-valuedness and logic we need to apply the ideas recalled in this paper (see [7] for more details).

5. Conclusion

In this paper we recalled principles of Lative Logic with application to information and processes of System-of-Systems. We considered two approaches:

Firstly, 'system as a supersystem of systems' , which has been developed earlier in several papers. This approach has been applied to social and health care, energy market modeling and to logical extension of design structure described as a matrix.

Secondly, 'system as independent interacting systems' is still under development, but we described how sorts and generalized terms may be transferred from system to system. Much more development and more detailed mathematical descriptions are needed. However, already at the current state of development one observes the following: it is the receiver's responsibility to interpret, by the morphism m', the sorts such that the receiver can produce generalized terms. Who owns the 'transmission channel', thus, basically the functor F?

References

[1] Adámek, J., Herrlich, H., Strecker, G.E., 1990. Abstract and Concrete Categories. Wiley.

[2] Eklund, P., Galán, M.A., Helgesson, R., Kortelainen, J., 2014a. From Aristotle to Lotfi, in: Seising, R., Trillas, E., Moraga, C., Termini, S. (Eds.), On Fuzziness. Springer. volume 298 of *Studies in Fuzziness and Soft Computing*. chapter From Aristotle to Lotfi, pp. 147–152.

[3] Eklund, P., Galán, M.A., Helgesson, R., Kortelainen, J., 2014b. Fuzzy terms. Fuzzy Sets and Systems 256, 211–235.

[4] Eklund, P., Höhle, U., Kortelainen, J., 2014c. The fundamentals of lative logic, in: Abstracts in Linz 2014, 35th Linz Seminar on Fuzzy Set Theory, Linz, Austria.

[5] Eklund, P., Höhle, U., Kortelainen, J., 2016. A survey on the categorical term construction with applications. Fuzzy Sets and Systems 298, 128–157.

[6] Eklund, P., Johansson, M., Kortelainen, J., 2018. The logic of information and processes in System-of-Systems applications, in: Collan, M., Kacprzyk, J. (Eds.), Soft Computing Applications for Group Decision-making and Consesus Modeling. Springer. Studies in Fuzziness and Soft Computing, pp. 89–102.
[7] Eklund, P., Johansson, M., Kortelainen, J., Salminen, V., 2017. The logic of DSM, in: 17th International Dependency and Structure Modeling Conference, DSM 2017, Espoo, Finland.
[8] Giry, M., 1981. A categorical approach to probability theory, in: Banaschewski, B. (Ed.), Categorical Aspects of Topology and Analysis, Springer. pp. 68–85.
[9] Goguen, J.A., 1967. *L*-fuzzy sets. Journal of Mathematical Analysis and Applications 18, 145–174.
[10] Helgesson, R., 2013. Generalized General Logic. Ph.D. thesis. Umeå University. Umeå, Sweden.
[11] Mac Lane, S., 1971. Categories for the Working Mathematician. Springer-Verlag.
[12] Manes, E.G., 1976. Algebraic Theories. volume 26 of *Graduate Texts in Mathematics*. Springer-Verlag.