FROM DEVS TO FORMAL METHODS: A CATEGORICAL APPROACH

D. E. STEVENSON, CLEMSON UNIVERSITY, 442 R. C. EDWARDS HALL, DEPARTMENT OF COMPUTER SCIENCE, CLEMSON UNIVERSITY, CLEMSON, SC 29634-1906

STEVE‘CS.CLEMSON.EDU

Abstract. The DEVS formalism developed by Zeigler and Sarjoughian is extended by the use of category theory. Several functors are needed to produce the correct categories. Verification and validation is modeled by adjunction and the information-gap approach to uncertainty.

Keywords. Simulation, DEVS, Category Theory, Information-Gap Uncertainty

1. Motivation

Two conclusions from the Foundations '02 workshop [12] were that (1) the formalisms of Zeigler, *et al.* [17] and Sarjoughian *et al.* [13] (collectively referred to as ZS) is the most widely held formalism by the community represented at the workshop and (2) there is a need for more formal methods to support the development of models and simulations.

The ZS formalism has some roots in general systems theory (GST) as developed by Klir [9, 8, 10] and many others and is based on a dynamical systems theory vocabulary. ZS posits algebraic concepts between specification levels, behavior, states, and components. ZS relies on compositionally closed systems of components. The concepts form a natural tie-in with category-theoretic concepts [1, 3, 4, 5, 6, 11].

The history of science and engineering is replete with stories of great advances occurring after notions are examined by new methods for formulation; for example, Hamilton's development of quaternions revolutionized physics. My intent is to explore DEVS through category theory with the goal of developing formal methods for modeling and simulation.

The goal is to transform the development framework (ZS) to a logical framework. In order to utilize ZS as a basis for formal methods, I must show how ZS can be translated to a it is to a formal system (one with axioms and rules of inference) in an acceptable meta-system. This makes modeling and simulation accessible to mathematicians and theoretical computer scientists by translating discipline dependent concepts into a broader setting. Reasoning occurs in *topoi* (singular *topos*) that form algebraic settings for logic and computation. Verification, validation, and accreditation can be approached as reasoning and as an adjunction. I develop my framework as described in [16].

My goal is to identify category-theoretic concepts that directly support ZS. These concepts will be used to guide research into developing formal methods for VV&A. In Section 2 I discuss concepts of the ZS framework and introduce categories related to the specification levels with the identification of possible categories. Section 3 presents an overview of the category theoretic concepts used in this paper. Section 4 describes the categorical formulation.

2. ZS Formalism

The fundamental objects of study is the *system*. Systems have inner structure and exhibit external behavior. In the system theoretic framework, system structure can be modeled as state transition systems and system behavior can be represented by input/output relationships. Furthermore, systems can be modeled as components that are coupled compositionally.

ZS suggest that systems theory focuses on three fundamental questions:

SA Systems analysis: the system is given and its behavioral characteristics must be determined.

SS Systems synthesis: the behavior is given, a set of components and rules are given

This work was partially supported by the Shodor Computational Science Fund funded by the Shodor Education Foundation, Inc., Durham, NC and NSF Grant DUE-0127488.

and the system must be constructed to match the behavior.

SI System inference: the system is given and we must infer its structure from its behavior.

Engineers study these classical systems theory problems, although a very small, informal poll of scientists shows they are not familiar with the terminology. I outlined in [15] why I believe modeling and simulation (M&S) goes beyond systems theories. For M&S, there are at least three more fundamental problems:

SR Systems representation: a system can be represented in many ways. The manner in which the system is represented often determines how well the system can be utilized.

SVR Systems verification: verification of a system definition is the determination of veracity with respect to rules and conventions.

SVA System validation or accreditation: validation or accreditation of a system determines how the system fits into the real world.

Actually, accreditation spans verification and validation since its goal is to certify a particular use.

Classical systems theory is primarily interested in problems SA and SS. In this context, the fundamental operations are *decomposition* for SA and *composition* for SS. But for VV&A, the issue is SI and the three other problems: SR, SVR, and SVA.

The ZS program specifies five specification categories in a hierarchy:

$H1$ Observational Frame. The observational frame specifies the resolution of the system: the variables, how the variables are measured with respect to accuracy and the time base.

$H2$ I/O Behavior. The time-indexed data collected from system observations as tuples. I take $H2$ to a category in or presentation, where we will use the symbol $\mathcal{B}eh$ for *behavior*

$H3$ I/O Function. The I/O function includes knowledge of the initial state. Given this initial state, the input stimulus produces a unique output.

$H4$ State Transition. Given a state and an input, the state transition system produces output state(s). I take $H4$ as a category that I denote $\mathcal{ST}$

$H5$ Coupled Component. Components are subsystems with their own behavior and the rules under which they may be composed. $H5$ is taken as a category that denoted $\mathcal{CC}$

The last fundamental concept of system specification to describe is the concept of a *morphism*. The ZS program defines morphisms in terms of the specification hierarchy above.

3. Category Theoretic Approach

What does category theory have to do with ZS or formal methods?

Formal methods are mathematical approaches to software and system development which support the rigorous specification, design and verification of computer systems. The use of notations and languages with a defined mathematical meaning enable specifications, (statements of what the proposed system should do) to be expressed unambiguously. Because they are formal mathematical structures there must be underlying formal (axiomatic) theories. Current formal methods are concerned with the *verification* but not validation. Using the concept first proposed in [16], we classify the verification direction as *covariant* functor and the validation direction as *contravariant.*

Category theory becomes a tool to understand the development process as a series of transformations. In order to be useful, formal methods must be able to represent complex structures as well as convert various viewpoints. By extending ZS into this structure, we have a complete mechanism for describing systems and then reasoning about them.

The ZS approach strongly suggests *category theory*, with its emphasis on *morphisms*, as a natural formalization setting. Category theory also played a role in development of General Systems Theory, primarily by Goguen [3, 4, 5, 6]. The usefulness of the categorical approach is that it links ZS and system theory on the one hand to abstract algebra, logic, and theoretical computer science on the other.

3.1. **Basic Categories.** A *category* $\mathcal{C}$ consists of *objects* $\mathsf{ob}\mathcal{C}$ and arrows $\mathsf{ar}\mathcal{C}$. There is no fixed notion of what the objects are; they could be sets or algebras or logics or data structures. The arrows are transformations, either as functions or morphisms. To give a concrete example, the category $\mathcal{S}et$ has sets as objects and total functions between sets as arrows.

For ZS, there are several obvious categories: $\mathcal{S}ys$, the category of systems; $\mathcal{B}eh$, the category of behaviors; $\mathcal{ST}$, the category of state-transitions; and $\mathcal{CC}$, the category of component-coupling systems. $\mathcal{C}om$ has $\text{OB}\mathcal{C}om$ as components, while if a, b are components, then there is an arrow from a to b, denoted $a \rightarrow b$, if component a can be transformed into component b. $\mathcal{B}eh$ objects might be different forms for displaying the behaviors and the the arrows would be morphisms that preserve the structure but change the way the behavior is written down.

We can algebraically operate on categories to form new categories in a way that mimic sets: *products* (Cartesian products) and *co-products* (disjoint unions). For example the category $\mathcal{C}om \times \mathcal{B}eh$ would pair every possible component with every possible behavior.

3.2. **Functors.** *Functors* map one category to another. A *functor* maps the objects of the domain category to the objects of the codomain category and maps the morphisms of the domain to the codomain. Functors transform mathematical structures while, in general, preserving some properties. A compiler is a functor which transforms algorithms written in one form (the higher level language) to algorithms written in another form (machine language) while preserving correctness. In the context of the paper, the levels of specifications are converted by functors because there may be a change is structure at each level. For example,

$$H1 \longrightarrow H2$$

could be a change in structure because mathematical conditions are imposed or there could be a preservation of structure.

I take the natural development direction as *covariant* arrows. A *contravariant* functor produces the *opposite* category $\mathcal{C}^{op}$. If $f : A \rightarrow B$ then $f^{op} : B \rightarrow A$. The category $\mathcal{C}^{op}$ has the same objects as $\mathcal{C}$ but has arrows as the opposites. The opposite category captures the concept of *duality* while the opposite arrows capture *inverse.*

3.3. **Topoi.** Topoi embody many of the properties of topologies and sets. From sets they collect concepts like initial objects (null sets) and terminal objects (closures). A topos is guaranteed to have certain functions, called *exponentials*, products, and co-products. The existence of products and co-products gives rise to concepts of sequence, limits, and co-limits. Operationally, the idea of subset is the same as the *equalizers* and quotients (partitions) as *coequalizers.*

A necessary concept relating to a topos is that of a subobject classifier, signified Ω. The subobject classifiers play the role of boolean values. The term *subobject classifier* arises from the manner in which the concept of subset arises in topoi. In the category $\mathcal{S}et$, we would symbolize $Y \subseteq X$ as $X \xrightarrow{\subseteq} \Omega$. Ω has a minimal structure somewhat looser than Boolean algbra called *Heyting algebra.*

I interpret $H4$ and $H5$ to be specifications and not the total of all possible implementations. This latter concept is captured in the concept of *free algebras.* Therefore, $H4 \longrightarrow ST$ and $H5 \longrightarrow \mathcal{CC}$ represent functors from the specification to the category of state-transition systems and component-coupling systems, respectively. These two functors are *realizability functors* [11]. The concept of realizability is central to automata (a fundamental concept in ZS).

4. Development

Let $\mathcal{S}ys$ be the category whose objects are anything that anyone designates as a system. Not all the objects need have mathematical models: certainly the human brain is a system but we do not currently have accepted mathematical models of thinking processes. A model for a system is generally not considered to be the system itself but to have been derived from observations at resolution; therefore, we need an $\mathcal{O}bs$ category, the categorical equivalent of $H1$, and a morphism $\mathcal{S}ys \longrightarrow \mathcal{O}bs$. The $\mathcal{O}bs$ are further manipulated through a discovery process into $\mathcal{B}eh$, the category of behaviors equivalent to $H2$. $H3$, $H4$, and $H5$ should be isomorphic categories the are various representations of what we normally call the conceptual model (See 4.1). I define the conceptual model as the category $\mathcal{C}on$ defined by

$$\begin{tikzcd}
\mathcal{B}eh \arrow[rr, "SI"] \arrow[dr, "Insight"'] & & \mathcal{ST} \arrow[dl, "SS"] \\
& \mathcal{CC} &
\end{tikzcd} \tag{1}$$

The meaning of diagram 1 is that the three categories are interrelated.

Finally, any of the theoretical statements can be mapped to a simulator, which is an object is $\mathcal{E}ff$. The running of the simulator produces a category of values $\mathcal{O}bs'$ that can be compared with $\mathcal{O}bs$ as described in [16]. The whole process can be visualized by

$$\begin{tikzcd}
\mathcal{S}ys \arrow[r, "resolution"] & \mathcal{O}bs \arrow[r, "discovery"] & \mathcal{C}on \arrow[r, "program"] & \mathcal{E}ff
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{E}ff \arrow[r, "verification"] & \mathcal{O}bs'
\end{tikzcd}$$

At a minimum, the verification functor must preserve the initial observations.

4.1. Formal Treatment of Verification and Validation.

In [16], I develop a framework within the current understanding of the philosophy of science that characterize modeling. The approach recognizes two distinct attributes of a model: correspondence and coherence. There are three subconcepts to coherence: logical coherence (verification); inferential coherence (soundness, consistency, and metrics/measurements); and explanation (relevance). Verification is a question of either testing (science) or proof (mathematics).

Validation is characterized in [16] as the coherent fit of model with current knowledge or inferential coherence. The overarching consideration is that models should be *lawful*:

- Objects in theories are idealizations.
- There can be inferred entities; i. e., items that occur in the model whose physical existence are not observed.
- Theoretical postulates are given to conceptually connect objects. These postulates can take the form of definitions, axioms, or rules of inference.

All three of these considerations must be validated: they must *correspond* to the appropriate system attributes. Validation is *contravariant* because the question of validation starts with the observed/computed values and these identify a particular system. Coherence requires topoi to work in that were provided above: Observations, models, and simulations.

[16] also describes the formal languages involved in theories: a logical language $\mathcal{L}_l$, a theoretical language $\mathcal{L}_T$, and an observational language $\mathcal{L}_o$. The set of all possible statements $\mathcal{T} = (\mathcal{L}_l + \mathcal{L}_T)^*$ is what one normally calls the "language" of the science or engineering discipline. The *semantics* of $\mathcal{T}$ is the functor from $\mathcal{T}$ to $\mathcal{L}_o^*$.

Finally, what needs to be a topos? Scientists and engineers typically reason about observations (via statistics). $\mathcal{T}$ is symbolic (logical) so it too is a topos. If $\mathcal{T}$ is too complex to solve system problems analytically, then a simulator is needed.

The fundamental algebraic view I take is the following: (1) any possible observation can be reproduced (validation) and (2) any produced output can be associated with a real system (verification).

There are three basic concepts needed to develop verification systems: logic, algebra, and topology. Logic describes truth in the abstract while algebra describes truth in the concrete. Semantics (the meaning of formal statements) is represented as functors from the syntactic (logical) language to the concrete application (algebra). Topology naturally enters through the concept of consistency, which requires the idea of subsets. The three can be linked isomorphically and then these can be generalize to include topoi [7]. The generalization of logical systems, then, are topoi.

4.2. Verification, Validation, and Accreditation.

In mathematics, terms like *verification* and *validation* are called notions. *Notions* are concepts that must be formalized in a particular theory to gain meaning. For example, the glossary entry for verification at the DMSO web site is a notion whereas a statement, say in system theory, like "a model M is verified if and only if Theorem X is satisfied" is more what we seek.

How the notions of *verification*, *validation*, and *accreditation* defined in an algebraic approach? Suppose system S is in $\text{OB}\mathcal{S}ys$ and that has truly linear behavior (slope and origin). Let model M be in $\text{OB}\mathcal{C}on$ and that is linear linear function with the correct slope and origin. We want (S, M) to be verified, validated, and accreditable at any level because M correctly predicts S's behavior and the modeling process correctly described S. This is the essence of the algebraic view: equality defined by the two maps. In words, "A validated verified model produces the system and the verified validated system produces the model."

$$\begin{tikzcd}\mathcal{S}ys \arrow[r, "validation"] & \mathcal{C}on \arrow[r, "verification"] & \mathcal{S}ys\end{tikzcd}$$

$$\begin{tikzcd}\mathcal{C}on \arrow[r, "verification"] & \mathcal{S}ys \arrow[r, "validation"] & \mathcal{C}on\end{tikzcd}$$

In category theory, such a structure is an adjunction, although there is more technical requirements than just the two functors.

If, on the other hand, S requires an infinite series to correctly describe its behavior, the map from S to $\text{OB}\mathcal{C}on$ is one-to-many, based on such things as resolution level and the number coefficients. This system has multiple degrees of freedom based on the "correctness" of the coefficients in the series. We now have uncertainty (the number and value of the coefficients). More and more observations can be made with the possibility of recovering the coefficient values. Regardless, we always have an approximation if S truly requires an infinite number of coefficients. For these reasons other reasons develop in [16], I view VV&A as a type of risk analysis problem.

The solution reached intuitively in [16] is developed theoretically in Ben-Haim [2]. An important theoretical advantage from Ben-Haim is that the development *demands* a topos. The direct break with conventional risk analysis is the view that analysis must be driven by available information. In other words, the decision maker needs a model of uncertainty. The information-gap approach is a non-probabilistic approach arising from uncertainty considerations in the engineering and sciences such as uncertain dynamical systems going back to 1973 [14]. This formulation falls into the basic approach in this paper. Ben-Haim defines uncertainty in terms of nesting of sets that are defined by an *uncertainty parameter* α. Such a formulation includes uncertainty parameters that are probabilities. A simple example shows the approach.

Recall our example of the system S with model M where S requires an infinite series but M has only a finite series. In a mathematical context we would say that *M approximates S* and would consider the convergence of the two series. In conventional terminology, we are asking about limits. Let $s(\mathbf{x})$ be the true description for a vector of parameters $\mathbf{x}$ and $m(\hat{\mathbf{x}})$ be the model function for its parameterization $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the approximation to the parameters as developed, say, through statistical methods. The *information gap model of uncertainty* for this can be defined by

$$U(\alpha, m) = \{s(\mathbf{x}) : |s(\mathbf{x}) - m(\hat{\mathbf{x}})| \leq \alpha\}$$

The categorical analogue of [2] is that models of uncertainty should be at least a partial order in the category of partial orders $\mathcal{POS}et$.

The information-gap formulation makes clear that virtually any concept that can be made into a partial order can be used to model uncertainty, including probabilistic models and numerical approximations. Returning to the decision-theoretic concepts in [16], the decision makers in any given instance must develop their own models of uncertainty; the decisions are make based on the various αs chosen.

5. Conclusion

I have presented a description of the DEVS framework in a category-theoretic setting. We only gain a slight amount of abstraction by considering $H1$–$H5$ as categories directly but do gain in abstraction as we move $H4$ and $H5$ to the the categories $\mathcal{ST}$ and $\mathcal{CC}$, respectively. Some of the ramifications of considering the categorical version were discussed. One new concept is the inclusion of the in information-gap model of uncertainty. The concept allows us to consider reasoning with uncertainty in a non-probabilistic setting.

While I take the ZS framework as the fundamental the further two disciplines are apart the harder it is to see analogies. Category theoretic developments can be used as a communication medium between M&S areas. The value of the categorical approach is not to supplant ZS but to discover what should be of universal concern and present those concerns to the communities that can bring knowledge to bear. For example, how much of the theory required to verify numerical solution of a differential equation describing the flight of a missile is applicable to a war-gaming simulation with no such mathematics of this type may occur?

The focus on functors and uncertainty rather than development can be illustrated by the following situation. The move from $H1$ to $H2$ is often by statistical regression techniques; i. e., imposing a model on the observations. What we then must consider is a *category* of possible models. Let $\mathcal{L}in$ be the category of linear models and $\mathcal{N}on$ be the category of There are decided differences between linear and nonlinear regression models and the eventual model development. It is easy to formulate a question categorically:

$$\begin{tikzcd}
 & \mathcal{O}bs \arrow[dl, "lin"'] \arrow[dd, "non"] \\
\mathcal{L}in \arrow[dr, "?"'] & \\
 & \mathcal{N}on
\end{tikzcd}$$

The relationship between $\mathcal{L}in$ and $\mathcal{N}on$ is injective (monomorphic); $\mathcal{L}in$ is an approximation of $\mathcal{N}on$. There must be a sequence of non-linear models based on (thinking formal power series) the degree of the models. Approximation is a form of uncertainty and this nesting is the centerpiece of the information-gap approach.

It is clear that this development has only begun. The obvious next step is to put detail into the characterization.

References

[1] M. A. Arbib and E. G. Manes. Machines in a category: An expository introduction. *SIAM Review*, 16:163–192, 1974.

[2] Yakov Ben-Haim. *Information-Gap Decision Theory.* Academic Press, Sep. 2001.

[3] J. A. Goguen. Mathematical representation of hierarchically organized systems. In E. Attinger, editor, *Global Systems Dynamics*, pages 112–128. S. Karger, Basel, 1971.

[4] J. A. Goguen. Categorical foundations of general systems theory. In F. Pichler and R. Trappl, editors, *Advances in Cybernetics and Systems Research*, pages 121–130. Transcripta Books, London, 1973.

[5] J. A. Goguen. Objects. *International Journal of General Systems*, 1(4):237–243, 1975.

[6] J. A. Goguen and Suzanna Ginaldi. A categorical approach to general systems theory. In *Applied General Systems Research*, pages 257–270. Plenum, 1978.

[7] R. Goldblatt. *Topoi: The Categorial Analysis of Logic.* North-Holland, 1984.

[8] George Klir. *Trends in General Systems Theory.* Wiley-Interscience, 1972.

[9] George J. Klir. *An Approach to General Systems Theory.* Van Nostrand Reinhold, 1969.

[10] George J. Klir. *Archtecture of Systems Problem Solving.* Plenum Press, 1985. ISBN 0-306-42867-3. QA295.K55 1985.

[11] E. G. Manes, editor. *Category Theory Applied to Computation and Control.* Springer Verlag, 1975. Proccedings of the First International Symposium, San Francisco, February 25–26, 1974.

[12] Dale K. Pace, D. E. Stevenson, and Simone Youngblood, editors. *Foundations '02: Foundations of Verification and Validation for the 21st Century, October 22–23, 2003, Laurel Maryland*, San Diego, CA, 2003. Society for Computer Simulation.

[13] Hessam S. Sarjoughian and Francois E. Cellier. *Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and Ai-Based Theories and Methodologies.* Springer Verlag, June 2001.

[14] Fred C. Schweppe. *Uncertain Dynamic Systems.* Prentice-Hall, 1973.

[15] D. E. Stevenson. Interdisciplinary knowledge for education in modeling and simulation. In *Proc. of SCSC '02, San Diego, CA, 14–18 Jul. 2002*, San Diego, CA, 2002. Society for Computer Simulation. CD Version has no page numbers.

[16] D. E. Stevenson. The Michelson-Morley experiment as a case study in validation. *Computers in Science and Engineering*, pages 40–51, Nov-Dec 2002.

[17] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Academic Press, 2nd edition, January 2000.