Position Papers of the Federated Conference on Computer Science and Information Systems pp. 11–17 DOI: 10.15439/2016F73 ACSIS, Vol. 9. ISSN 2300-5963

Many-valued logic in manufacturing

Patrik Eklund
Umeå University
Department of Computing Science
SE-90187 Umeå, Sweden
Email: peklund@cs.umu.se

Magnus Löfstrand
Umeå University
Department of Computing Science
SE-90187 Umeå, Sweden
Email: maglof@cs.umu.se

Abstract—**This paper shows how to enrich the language used in the manufacturing industry regarding information structure and its representation for products and production processes. This is enabled because of our use of many-valued logic, and in order to complement the numerical approach commonly appearing in such representations. We underline the importance of utilizing mathematical disciplines like algebra and logic side-by-side with the utility of analysis and stochastics.**

I. Introduction

WITHIN a communicating resources perspective, many-valued logic will be applied to decrease Data *Complexity* for applications in *Big* Data. We will further introduce a structure for *functioning* classification in order to complement and interact with the traditional view of *faults and failures* in product and production subsystems. We will develop a prototype information structure demonstrating the potential use of a multi-valued logic enriched classification of *functioning in machines and manufacturing* (MCFu), as related to enriched classification of *faults in machines and manufacturing* (MCFa). Our *concept of digitalization*, in order to support manufacturing and machine simulation and modelling as e.g. part of design processes, whether e.g. in the case of cars and car production, related to subsystems such as powertrain, electronics or interior, will be based on nomenclatures, classification and ontology as part of production and in particular production where *information and process* is tightly connected. Our language and concept, with MCFu and MCFa classifications, will support transformation and transferal of data between robots, humans and companies (Internet of Things 2.0), thus strengthening industry.

For machines and vehicles, **faults** and **failures** reduce their function either partly or completely. The traditional enngineering view of functioning classifications does not always connect well and accurately with corresponding faults classifications. Machines and vehicles in the widest sense, e.g., including operators and end-users of a wide variety, need to interact with the environment, and therefore also need to appropriately connect *all resources*. To facilitate the connections, these respective resources need a *common language for representation of faults and functioning* and the relation between them. Further, within and across various machine subsystems, information and information structures are currently insufficiently connected with respect to standard information representation. Numerical approaches to standard and information structure development and utility within the industry promotes uncertainty and *many-valued considerations* mostly in directions of analyzing variability, and indeed variability as related to numerical values. Logical many-valuedness, and underlying structures are basically missing in most numerical approaches.

Our approach to many-valued logic can be seen also to enrich e.g. our anticipated combination of DSM [1] with the Axiomatic Design model [2], which structurally extends approaches dealing with the unstructured matrix view [3], [4]. The DSM view involves modularity of component, people and activity, whereas the Axiomatic Design model involves customer, function, product and production. The underlying logical scope appears in Section II, including the view of combination of models with respect to their granularity and modularity.

Respective logical enrichment of DSM and Axiomatic Design is prerequisite to a **logical merger of these two models into a unified conceptual framework for information and process for products and production**. Information basically resides within some use-case subprocess task, in turn being part of a larger process. Communication involving people performing activities is therefore not just a transmission but also requires transformation of data from a particular subprocess to be integrated and used within another subprocess. **We thereby iterate complexity of problems and integration of solutions**. Because of our logical model, industrial applications and required systems-oriented research are done in iteration, explained within a common interdisciplinary conceptual framework.

II. Logical Approach

A. Common Interdisciplinary Conceptual Framework

Smart [`systems`] basically means smart [`components-people-activities`] in combination and interaction, i.e., smartness of the multidomain in DSM. This builds upon smart [`components`], smart [*people*] and smart [*products*], and this also explicitly initiates the explanation how smartness resides within a logical framework. Smartness thus embraces being correct and consistent about information. Further, we make a distinction between smart [*system of systems*] and [*smart system*] of systems. Various definitions appear in these contexts. Smart products are typically expected at least to involve monitoring, allow control, invite optimization and to be autonomous, each capacity and capability building upon the preceding one, so that e.g. in order to have control capacity, a product must have monitoring capability.

Logically connecting activities and functioning, components and production, and enabling information structures to enrich availability, provide the foundations for describing the wide variety of smartness. In a concept like *time to failure* it is important to realize how *failure* needs to specified and enhanced by the underlying signature. Otherwise a failure may remain seen as an event, i.e., simply just a constant operator in the signature.

Maintenance strategy, preventive schedule (prevention guideline), predictive schedule (risk estimation), corrective (intervention), and inspection (monitoring and assessment scales), all appear in one form or another in the private and public sector. From stakeholder point of view, the private and public sectors are similar in that they **share the logic of information structures** even if the content is entirely different. The private and the public sectors, on the other hand, differ in that the private sector is a B2B matter, whereas the public sector is Triple Helix [5], including also a formal political dimension in university-industry-government relationships.

B. Extending the relational-logical view

We define a logical SoS involving dialogue of various form, interaction and integration, interfacing and transferral of information and structures, within and between subsystems. Further, system availability [3] and task scheduling [4] are important disciplinary aspects. The provision of a functioning standard will be extended to explain the potential use of a multi-valued logic enriched classification of functioning (MCFu) as related to a similarly enriched classification of failures (MCFa).

The logical enhancement of DSM is briefly outlined as follows. DSM uses a set X of *system elements* to create the set product $X \times X$ viewed as a *matrix*. Bivalent *interactions* between elements form a relation $R \subseteq X \times X$ sometimes viewed as a digraph. That bivalent relation can equivalently be represented as a function $\rho_R : X \times X \to \{0, 1\}$, so that aRb, i.e., $(a, b) \in R$ if and only if $\rho_R(a, b) = 1$. The set X is initially unstructured, i.e., no order or operations of any kind are imposed on X. Elements $x \in X$ therefore have no initial annotation or attribution of any kind. However, once elements are given names so as to represent components, granularity is intuitively recognized, and respective modelling approaches decide about levels of detail. A typical view is that *rolling up into lesser detail* [1] is accepted if significant information or insight isn't lost. We formalize this by viewing system elements as *terms over a signature* $\Sigma = (S, \Omega)$, where S is the structure of sorts (types) and Ω is the structure of operators. The set of system elements is then initially enriched to a set of terms $\mathsf{T}_\Omega X$, where X is a set of variables. As an example, an element $actuator$ as a point in set without structure is symbolically meaningless, whereas $actuator(f(x_1, \dots, f(x_n))$ as term builds upon $aactuator$ and f as operators, and $x_1, \dots, x_n$ as variables. Similarly, a temperature control could be a term $EATC(g(y, z))$. A many-valued interaction between the actuator and the temperature control could then be given as

$$\rho(actuator(f(x_1, \dots, f(x_n)), EATC(g(y, z)))$$

where $\rho : \mathsf{T}_\Omega X \times \mathsf{T}_\Omega X \to \{no, weak, strong\}$ is a three-valued relation over $\mathsf{T}_\Omega X$, which unravels hidden information as compared to modelling using X only. This notation is based on category theory, where $\mathsf{T}_\Omega : \mathtt{Set} \to \mathtt{Set}$ is a *term functor* [6] that can be extended to a *monad*. The monad properties allow substitutions of expressions within a term to be composable, so that the substitutions $x = g(y, z)$ and $z = 22$ as composed and applied to $EATC(x)$ leads to the term $EATC(g(y, 22))$. This is obvious when we use relations over $\mathsf{T}_\Omega X$, but once we want to have various structured sets of terms, then we need *monad compositions* [6]. Powersets are typical examples, where system elements in a rolled up view are clustered into subsets of system elements, and that subset is given a new name as a new element in the rolled up DSM. This is the first steps in the logical modelling of SoS.

In a many-valued logical system for describing a DSM, components and subcomponents can be viewed in the same logical framework, e.g., with $piston(\dots)$ and $crankshaft(\dots)$ as parts of a $crank(\dots, piston(\dots), \dots, crankshaft(\dots), \dots)$. In a bivalent view, this enrichment is still not dramatic. However, when adding the possibility that the term functor acts over various *Goguen categories* [7], $\mathsf{T}_\Omega : \mathtt{Set}(Q) \to \mathtt{Set}(Q)$, where Q is the algebraic structure for the selected view of many-valuedness, then many-valuedness can be invoked in a wide variety of ways in the DSM, Axiomatic Design and TRIZ models. In order for this paper to be more self-contained, we included some detail of the term functor in the Appendix.

Note that typing is prerequisite to modularity, i.e., the set of sorts in the signature, with related type constructors [6]. Granularity then comes from using a more elaborate structure of operators, ranging from the most coarse-granular situation where Ω contains only constants (of different type) to finer-granularity involving a wide range of operators with various arities, and operating over a rich structure of sorts and constructed types.

Rolling up into lesser detail will actually provide a name for a subset or cluster of system elements. It is then immediately important to note how symmetry and associativity, with trivial reflexivity, of a relation, means that we have an equivalence relation. These relational properties mean that X subdivides into equivalent classes. Again, in the bivalent case, this is apparent and straightforward, but when moving to the many-valued case, symmetry and associativity are then also many-valued, and the subdivision of $\mathsf{T}_\Omega X$ becomes a non-trivial matter. Another matrix approach example is the Pugh selection matrix, where the arithmetic calculation ignores the intuitive order among the design criteria, where e.g. ease and time of an implementation precedes cost of it. Here is yet another example where many-valued approaches to types and operations enable modelling of order, i.e., how different criteria have precedence over other criteria, hidden in the matrix.

We further provide a logical enrichment of the matrix/relational models involved, so that relational types and strengths are further enhanced by **symmetries, associativities and granularities**. This brings in the mathematical disciplines of logic and algebra as a complement to numeric and stochastics. The **lativity**[1] **of systems** is a key feature where design precedes simulation, logic precedes design, statistics precedes analytics, subsupply precedes supply.

III. System-Of-Systems

A. System-of-systems definition

The definition of system of systems (SoS) is shown to be **logically extendable** over current definitions [8], which mostly describe types of systems of systems rather than their information content. In doing so, we are thereby closer to a system of systems views described as a collection of task-oriented or dedicated systems that pool their resources and capabilities together to create a new, more complex system, which offers **more functionality** and **performance** than simply the sum of the constituent systems [9].

The goal of a SoS architecture is to get **maximum value out of a large system**, comprised of smaller systems, by understanding how each of the smaller systems work, and developing industrially suitable interfaces based on industrial needs and usability studies. Since industrial applications are significantly diverse, and manufacturing industries communicate both within and across their boundaries, they are therefore complex and involve people with diverse backgrounds. Hence, complexity and cooperation are very important challenges, and robust cooperation and coordination between systems of humans and computing devices is crucial for progress in development of research results and in application development. Additionally, and perhaps most importantly, in particular applications, **implementation, verification and validation** activities must **attain high robustness and trustworthiness**, which requires underlying logical structure from complexity and facilitating cooperation as well as improved product and production development process data management.

B. The information and process view of SoS

Our language and concept, given the logic framework with MCFu and MCFa classifications, will support transformation and transferal of data between robots, humans and companies (Internet of Things 2.0), to optimally support the manufacturing industry in their contractor-subcontractor business relations, where sustainable production is measurable by key performance as described by our logical framework.

[1]'Lative' is related to motion, and more specifically, motion 'to' and 'from', so when terms appear in sentences, terms 'move into' sentence, and 'appear within' sentences. At the same time, sentences 'move away from' terms, and separates terms from sentences. In comparison, 'ablative' is motion 'away', and nominative is static.

Formal process modelling languages, like UML with its Behavioral Modeling, SysML and BPMN, from the Object Management Group (OMG), will enable formal information structures to be integrated with the process structures in manufacturing applications. The logic and ontology of information structures in industrial applications will be used for markup purposes. Upscaling of applications should follow an overall strategy as well as concrete suggestions and specific steps for the upscaling process and progress.

C. Application domains

For example, in forest products and mining industries, the general need is to ascertain optimal operation and utility of their subsystems, i.e., within the normal operation parameters. The goal is to maintain normal operation and optimal efficiency e.g. by avoiding unplanned stops and energy loss. The supplier-customer relation e.g. with respect to service and maintenance is important in particular from a SoS perspective.

Further, the automotive industry faces new SoS challenges e.g. due to the ambition to develop self-driving cars. Doing so brings developments much more outside the car itself, with needs to consider traffic and the environment as parts of the overall SoS. Customer experience, including safety and quality, also becomes extended with other aspects still not considered. From SoS point of view, traffic as a conglomerate of cars is composed differently as compared to a car being assembled from its components. Apart from industrial applications, also in health care there are several well-known systems-of-systems that can be logically treated similarly with respect to modelling of **information and process**. Ageing is a typical area where health and social care need to interact and become integrated in common care pathways. Falls prevention can be view as a specific example. Falls represent a major cause of burden and death in older adults [10]. Enhanced models of care pathways, with enriched data attached to targeted subsystems based on the herein presented logics framework, will enable monitoring of the overall care SoS performance, including macro-, meso- and micro- levels.

IV. State-Of-The-Art

Originality of our **logical approach** is two-fold. On the one hand, the use of classification of functioning, as clearly separated from but connected with taxonomies of products and classifications of structure, is an original approach in the manufacturing industry. Further, many-valuedness annotated with codes, and the way codes as well as structures of codes are many-valued, is a novelty not yet seen within manufacturing. On the other hand, and still from disciplinary research point of view, our approach to enable many-valuedness at all levels and modules within a logical machinery, is unique within the logic and in particular within the many-valued logic community. Traditional many-valuedness e.g. in form of fuzzy sets and fuzzy logic, and in particular as appearing within fuzzy control techniques, is an untyped and numeric based technique that further relies only on unstructured relations, like in Zadeh's compositional rule of inference [11], and invokes many-valuedness only as far as truth values are concerned. Whereas statistics produce uncertainty quantification, **many-valued logic provides algebraic computing with uncertainties**.

Logic is a structure containing signatures and constructed terms and, and statements or sentences *latively* constructed based on terms. Similarly, sentences and conglomerates of sentences are fundamental for entailments, models and satisfactions, in turn part of axioms, theories and proof calculi. This lativity is suitably expressed in category theory using functors and monads, where constructions act over underlying categories in form of monoidal categories. Category theory is thus a suitable metalanguage for logic, in particular when applications and typing of information must be considered. Uncertainty may reside in generalized powerset functors, and may be internalized in underlying categories. In both cases, suitable algebras must motor this uncertainty representation, and quantales are very suitable in this context. [20]

From a systems-oriented research point of view, logical enrichment of **availability simulation, faults and functioning**, DSM, Axiomatic Design, TRIZ, and other information and process related models, connect not only to methods in modern logic, but also to the historical traditions in logic such as represented by sets and types in Principia Mathematica or type theory as initiated by Schönfinkel's Bausteine [12], Curry's functionality [13] and Church's simple typing [14]. Göttingen and Hilbert's foundations of mathematics were a driving force for Gödel and his work in Vienna, and many-valued logic was started by the Lwow-Warsaw school. Kolmogorov's Aufgabe [15], related to the TRIZ view of inventiveness, as a task rather than a problem is a broader foundation for algorithm as later developed by Turing [16].

For classification purposes, type constructors must make use of category in order to enable underlying categories that represent uncertainties, using a three-level signature [6], where structured powerset constructors can be handled properly. The generic scale of uncertainties resides in those underlying categories, and the algebraic use of the scale [19] is a technique that is unavailable in traditional type theory approaches e.g. within homotopy type theory (HoTT) [17]. A typical first step to many-valuedness is adding an unspecified to two-valuedness.

Relations like trees and matrices are basically unstructured sets and relations, which e.g. means that our approach potentially enriches design methodologies which basically use matrix computations to relate physical product structure with function. Values in matrices are always just numerical values, and also **detached** from any nomenclatures. Logic enables representation where relations become enriched with more structure and attached with classifications. Enrichment of structure for products then spills over to enrichment of structures in production. Thus, engineers are potentially provided with tools that enables improved quantification for design evaluation as well as for quality assurance and control, e.g. as those appearing in failure mode and effects analysis (FMEA) [18].

Industrial product development challenges relate, on the one hand, to information structures, respectively, for customer, function, product and production, and, on the other hand, to compatibility and transformations between these structures. DSM is typically used for internal structure descriptions, whereas Axiomatic Design is typically used for the transformations. See [1], [2] for details and examples. In both models, matrices only, with numerical and untyped values, are used to represent relational structures. Transformation of information within these models should indeed not just restrict to mapping of matrix content in an unstructured manner, but rather start from identifying the relational content, and thereby enable expansion and enrichment towards using structure preserving transformations.

A. DSM

In its most rudimentary form, a Design Structure Matrix [1] is a relation $\rho : X \times X \to \{no, yes\}$, where X is a set of *system elements*. The engineering understanding of such a system element may be very complex, but in the mathematical model of it, it is just an element, or actually a name of an element. A matrix is more appealing if yes values appear close to the diagonal, which implies a certain **nearness** between system elements. With values distant from the diagonal, clustering can provide conglomeration points with values that make the clustered matrix appear more diagonal.

The bivalent interaction is sometimes extended to a three-valued interaction with names in the three-valued set of truths being 'strong interaction', 'weak interaction' and 'no interaction'. There are no logical connectives, so there is no explicit propositional logic annotated with DSM. In the traditional DSM model there is also no considerations for a 'not specified', which would invite to viewing that three-valued truth set as a non-commutative quantale [20] or commutative Bocvar-Kleene algebra [21], [22]. Kleene called than in-between value 'unspecified', whereas Bocvar called it 'senseless'. Kleene used his three-valued logic e.g. to model partial functions within recursion, so 'unspecified' in logical connection with something specified is an interchangeable (commutative) operation.

In addition to bivalent interaction, multivalent interaction exists also as related to the use of typed interactions, like the one with four types, respectively, for 'spatial', 'energy', 'information' and 'materials'. Each type is valuated within a 5-scale $\{-2, -1, 0, +1, +2\}$, with -2 for 'detrimental', and $+2$ for 'required'. The internal algebraic structure of the set of components is, however, not given.

Product, organization and process structures, respectively, involving components, people and activities, also appear as integrated in a multidomain architecture.

B. Axiomatic Design

Information and process development addresses the challenge to combine components and product taxonomies with activity and process hierarchies, in order to support decision-makers, engineers and customers. Many-valued logic enriches the anticipated combination of DSM with the Axiomatic Design model [2], which in effect goes far beyond approaches dealing only with the unstructured matrix view. Whereas the DSM view involves modularity of component, people and activity, the Axiomatic Design model involves respective domains for customer, function, (physical) product and (observable) process.

Decoupled design, in case of a triangular matrix, is desirable, as this enables sequential consideration within the domains. Axiomatic Design theory is product design which start from using customer attributes (CA) as a basis for functional requirements (FR), in turn to map over to design parameters (DP), and from there arriving at process variables (PV). The FR to DP mapping is the most critical one.

Attributes, requirements, parameters and variables are all identified by names only, i.e., they are logically constants (0-ary operators). Independence and Information Axioms are formulated using these constants, and decomposition within and between (zigzagging) domains is simply saying that a constant becomes the name of a set of new constants, building up a hierarchy of constants.

Axiomatic Design involves a matrix view between domains, but not within a domain as in the case of DSM. Integrating DSM into Axiomatic Design using the unstructured and traditional models is suggested in [23]. Integration based on our structured approach will additionally involve structure-preservation.

Respective logical enrichment of DSM and Axiomatic Design is prerequisite to a logical merger of these two models into a unified conceptual framework for information and process for products and production. Information basically resides within some use-case subprocess task, in turn being part of a larger process. Communication involving people performing activities is therefore not just a transmission but also requires transformation of data from a particular subprocess to be integrated and used within another subprocess. We can thereby iterate complexity of problems and integration of solutions.

C. TRIZ

TRIZ [24] as a *theory of inventive problem solving* is a general model, and an informal model as more formal data, logical or computational models are not included. For algorithms in ARIZ [25], data models are required, but are still on a very general level. Objects and classes, like those modeled e.g. by Class Diagrams in UML, can be used, but will not suffice in order to represent and solve contradictions in TRIZ' contradiction matrix, since TRIZ inherently involves processes and behavior. The UML Class Diagram is simply a data model. Furthermore, the features in the contradiction matrix are of a wide variety of types, which are lumped together into one "set of features". This makes it unsuitable to view the matrix as a basis for a relational model. The set of features must be dissected and structured, and the features themselves must be enriched and further specified.

The 40 TRIZ Principles are also very different in nature and content, and are more like principle of common sense than principles of reasoning. The 'Preliminary action' principle for the related features 'Reliability' and 'Loss of time' basically recommends prevention of faults, prior to detection of faults, should they happen. Improving feature 'Ease of repair' as related to worsening feature 'Device complexity' involves the 'Segmentation' principle with respect to the need for increasing transparency and modularity. In logic we would say that "this logic is sound but not complete" is a metalogical statement, whereas "if you have to use a logic that sound but not complete, try to use a logic that is as complete as possible" is a principle more in the style of common sense.

Several features in the contradiction matrix are physical and/or geometrical, which logically can be expressed by structured terms rather than as unstructured concepts, as typically seen within description logic. However, description logic can be enriched [26] in order to make it better fit as a logic for TRIZ like ontologies. Features like 'measurement accuracy' and 'manufacturing precision' can be managed by adopting a logical framework with underlying signatures acting over a monoidal category [27] representing multivalence and uncertainty. Features like 'loss of ...' and 'reliability' can be logically detailed for particular problem contexts, where 'loss of ...' features are related to functioning rather than faults and failures. The 'Ease of ...' features can only be described by subprocesses, so in the case of UML we would need to consider to use Behavior Diagrams, or, if staying within OMG standards, move to using SysML. The 'Productivity' feature is closer to being analyzed within the BPMN model. Examples with applications in crisis management have been developed in [28].

V. UPSCALING

Industrial upscaling is more than just replication or multi-piloting, i.e., not just going from a small pilot to a large pilot. Specific pilots and cases expand within and across companies, and this expansion requires availability and acceptance of the common language, so that when scaling up, geographically distributed teams and cooperating stakeholders realize the need to be even more precise about underlying logical-relational information structures as compared to just doing specific and local pilots. Thus, the logical approach presented here supports improved upscaling.

A. Upscaling strategy

Upscaling should follow an overall strategy as well as concrete suggestions and specific steps for the upscaling process and progress. Guidelines are required in order to manage the framework that involves the critical elements in scaling up. Logic as an ingredient in manufacturing has its upscaling focus (at least) on the following:

- Design and manufacturing issues, which are supported by company specific and locally generated well proven practices of systematic effectiveness and feasibility, are key factors that help to increase the likelihood for successful and sustained upscaling.

- The balance between the rigid scaling up models and the pilot implementations residing within that scaling up process and structure needs to be maintained.
- Scaling up may often require additional managerial and financial input, and also an acceptance that upscaling usually implies a longer timeframe as compared to typically seen in ordinary company project cycles.

B. Steps in upscaling

A general upscaling model, building upon our logical-relational machinery, includes the following steps:

i. Descriptive identification of good practices using our language and classification, and enabling refined and broadened data collections and monitoring.
ii. Prescriptive analytics (numerics) and assessment (logics) of the viability and benefits of upscaling within domains of industries.
iii. Logical classification of good practices for replication, whenever feasible, and for suitably adapted implementation with respect to a variety of competences and industrial circumstances.
vi. Facilitation of partnership for scaling up within and across corporations, making resources available.
v. Identify key success factors for generalized implementation, and recognize lessons learnt.

Compliance within upscaling and reinforced changes in working pattern is strongly emphasized, where also personal skill and personalized behaviour is fundamental. Upscaling should build upon excellence achieved within information structuring cultures in the production industry, and aims at enriching these structures using the logical-relational approach.

VI. Conclusions

We have shown how to enrich the language used in the manufacturing industry regarding information structure and its representation for products and production processes. This is achieved using many-valued logic, in order to complement the numerical approach commonly appearing in such representations. We underline the importance of utilizing mathematical disciplines like algebra and logic side-by-side with the utility of analysis and stochastics. Within a communicating resources perspective, many-valued logic will potentially contribute to *information structuring* and *management of data complexity* in big data applications.

In future papers we will further introduce a structure for functioning classification in order to complement and interact with the traditional view of faults and failures in product and production subsystems. Our concept of logic modelling is based on nomenclatures, classification and ontology as information structures part of and supporting production, and indeed in production where information and process is tightly connected. Support for manufacturing, machine simulation, and modelling, then becomes part of design processes, e.g. in cases like cars and car production, as related to subsystems such as powertrain, electronics or interior,

Acknowledgement

This work is supported by the *Logic in Manufacturing* (LiM) project, with gratefully acknowledged funding from the Swedish Innovation Agency (VINNOVA) PRODUKTION2030 programme.

References

[1] S. D. Eppinger, T. R. Browning, *Engineering Systems: Design Matrix Methods and Applications*, MIT Press, 2012.

[2] N. P. Suh, *Axiomatic Design: Advances and Applications*, Oxford University Press, 2001.

[3] M. Löfstrand, M. Karlberg, J. Andrews, L. Karlsson, *Functional product system availability: simulation driven design and operation through coupled multi-objective optimization*, International Journal of Product Development **13** (2011), 119-131.

[4] S. Reed, J. Andrews, S. Dunnett, P. Kyösti, B. Backe, M. Löfstrand, L. Karlsson, *A modelling language for maintenance task scheduling*, In: 11th International PSAMS and ESREL 2012 Conference. vol. 1, 201-211.

[5] H. Etzkowitz, L. Leydesdorff, *The Triple Helix—University-Industry-Government Relations: A Laboratory for Knowledge-Based Economic Development*, EASST Review **14** (1995), 14-19.

[6] P. Eklund, M.A. Galán, R. Helgesson, J. Kortelainen, *Fuzzy terms*, Fuzzy Sets and Systems **256** (2014), 211-235.

[7] P. Eklund, J. Kortelainen, L. N. Stout, *Adding fuzziness using a monadic approach to terms and powerobjects*, Fuzzy Sets and Systems **192** (2012), 104-122.

[8] M. Jamshidi, *System-of-Systems Engineering - A Definition*, IEEE SMC 2005, 10-12 Oct. 2005.

[9] S. Popper, S. Bankes, R. Callaway, D. DeLaurentis, *System-of-Systems Symposium: Report on a Summer Conversation*, July 21âĂ¿22, 2004, Potomac Institute for Policy Studies, Arlington, VA.

[10] H. Blain, F. Abecassis, P. Adnet, B. AlomÃÎne, M. Amouyal, B. Bardy, et al., *Living Lab Falls-MACVIA-LR: The falls prevention initiative of the European Innovation Partnership on Active and Healthy Ageing (EIP on AHA) in Languedoc Roussillon*, Eur Geriatr Med. **5** (2014), 416-425.

[11] L. Zadeh, *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Trans. Systems, Man and Cybernetics **3** (1973), 28âĂ¿44.

[12] M. Schönfinkel, *Über die Bausteine der mathematischen Logik*, Mathematische Annalen **92** (1924), 305-316.

[13] H. B. Curry, *Functionality in combinatory logic*, Proc Natl Acad Sci USA **20** (1934), 584-590.

[14] A. Church, *A formulation of the simple theory of types*, The journal of symbolic logic **5** (1940), 56-68.

[15] A. N. Kolmogorov, Zur Deutung der intuitionistischen Logik, Mathematische Zeitschrift **35** (1932), 58-65.

[16] A. M. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society. Ser. 2, Vol. 42 (1937), 230âĂ¿65.

[17] *Homotopy Type Theory: Univalent Foundations of Mathematics*, The Univalent Foundations Program, Institute for Advanced Study, 2013.

[18] *MIL-P-1629 - Procedures for performing a failure mode effect and critical analysis*, United States Department of Defense (9 November 1949).

[19] P. Eklund, U. Höhle, J. Kortelainen, *A Survey on the categorical term construction with applications*, Fuzzy Sets and Systems, Available online 13 July 2015.

[20] P. Eklund, U. Höhle, J. Kortelainen, *Non-commutative quantales for many-valuedness in applications*, Proc. IPMU 2016, to appear.

[21] D. A. Bocvar, *Ob odnom trechznacnom iscislenii i ego primenenii k analizu paradoksov klassiceskogo funkcional'nogo iscislenija*, Mat. Sbornik **4** (1938), 287-308.

[22] S. C. Kleene, *On notation for ordinal numbers*, J. Symbolic Logic **3** (1938), 150-155.

[23] D. Tang, R. Zhu, S. Dai, G. Zhang, *Enhancing axiomatic design with design structure matrix*, Concurrent Engineering: Research and Applications **17** (2009), 129-137.

[24] G. S. Altshuller, *Creativity as an Exact Science*, Gordon & Breach, 1984.

[25] G. S Altshuller, *The Innovation Algorithm: TRIZ, systematic innovation and technical creativity*, Technical Innovation Center, Worcester, MA., 1999.
[26] P. Eklund, *The syntax of relations*, Proc. IPMU 2016, to appear.
[27] S. Mac Lane, *Categories for the Working Mathematician*, Graduate Texts in Mathematics **5** (second ed.), Springer, 1998.
[28] P. Eklund, M. Johansson, J. Karlsson, R. Åström, *BPMN and its Semantics for Information Management in Emergency Care*, Fourth 2009 International Conference on Convergence and Hybrid Information Technology (ICCIT 2009), IEEE Computer Society, 273-278.

Appendix

In the following we briefly introduce notation and constructions needed in our descriptions related to our SoS related logic, and in particular for its underlying signatures and terms. The many-sorted term monad $\mathbf{T}_\Sigma$ over $\mathtt{Set}_S$, the many-sorted category of sets and functions, where $\Sigma = (S, \Omega)$ is a signature, can briefly be described as follows. For a sort (i.e. type) $\mathtt{s} \in S$, we have sort specific functors $\mathsf{T}_{\Sigma,\mathtt{s}} : \mathtt{Set}_S \to \mathtt{Set}$, so that

$$\mathsf{T}_\Sigma(X_\mathtt{s})_{\mathtt{s}\in S} = (\mathsf{T}_{\Sigma,\mathtt{s}}(X_\mathtt{s})_{\mathtt{s}\in S})_{\mathtt{s}\in S}.$$

The important recursive step in the term construction is

$$\mathsf{T}^{\iota}_{\Sigma,\mathtt{s}}(X_\mathtt{s})_{\mathtt{s}\in S} =$$
$$\coprod_{\mathtt{s}_1,\ldots,\mathtt{s}_m} (\Omega^{\mathtt{s}_1\times\cdots\times\mathtt{s}_m\to\mathtt{s}})_{\mathtt{Set}_S} \times \mathsf{arg}^{\mathtt{s}_1\times\cdots\times\mathtt{s}_m} \circ \bigcup_{\kappa<\iota} \mathsf{T}^{\kappa}_{\Sigma}(X_\mathtt{s})_{\mathtt{s}\in S}$$

and then with

$$\mathsf{T}^{\iota}_\Sigma(X_\mathtt{s})_{\mathtt{s}\in S} = (\mathsf{T}^{\iota}_{\Sigma,\mathtt{s}}(X_\mathtt{s})_{\mathtt{s}\in S})_{\mathtt{s}\in S},$$

we finally arrive at the term functor

$$\mathsf{T}_\Sigma = \bigcup_{\iota<\bar{k}} \mathsf{T}^{\iota}_\Sigma.$$

The term functor construction can be extended so that $\mathsf{T}_\Sigma : \mathtt{C} \to \mathtt{C}$ operates more generally over monoidal biclosed categories $\mathtt{C}$. If $\mathtt{C}$ is $\mathtt{Set}$, we have the construction above, and with the Goguen category $\mathtt{Set}(Q)$, where Q is a quantale, we have a multivalent and typed situation enabled by the signature acting over the selected underlying category.

Term functors constructed in this way can be extended to monads, so that substitution can be composed. A substitution in this categorical context is a morphism in the Kleisli category of the related monad.

Monad compositions further enable to arrive at generalized sets of terms, where the typical example is composing the term functor T_Σ with the powerset functor P in order to obtain the monad $\mathbf{P} \circ \mathbf{T}_\Sigma$. More elaborate generalized set functors Φ can be applied in order to make use of the composition $\mathbf{\Phi} \circ \mathbf{T}_\Sigma$.

For more detail, and for the purely categorical constructions of the corresponding term monads, the reader is referred to [6].