On a federated architecture for utilizing domain-specific descriptive models

Freddy Kamdem Simo, Dominique Ernadote, Dominique Lenne

On a federated architecture for utilizing domain-specific descriptive models[*]

Freddy Kamdem Simo
[1, 2]
frks@protonmail.ch

Dominique Ernadote
[2]Airbus Defence and Space
Elancourt, France
dominique.ernadote@airbus.com

Dominique Lenne
[1]Sorbonne Universités, Université de Technologie de Compiègne
Heudiasyc, CNRS, Centre de recherche Royallieu, CS 60319
Compiègne, France
dlenne@utc.fr

In a Systems Engineering setting, various models are produced using a variety of methods and tools. Focusing on a type of models – called descriptive models – which we shall describe, we argue that, while the clarity and precision of models are essential for their exchange and reuse, the way in which the data of these models are defined and the information conveyed is also essential for their (re) utilization – e.g., for analysis or synthesis purposes. Category Theory has made it possible to link seemingly separate fields or domains, so that anything that can be rewritten in this framework benefits from a level of abstraction and a relational viewpoint – essential to address complexity. We therefore take advantage of this framework to define a federated architecture for projecting and conveying these models without sacrificing clarity and precision. A federated architecture has two important advantages. On the one hand, it unifies these models from their structure; on the other hand, it allows a specific usage (instantiation or interpretation) of this structure within a business domain. We define the structure of these models as a symmetric multicategory. In particular, we rely on matrices over a semiring to define morphisms and their composition. The choice of matrices is intended to facilitate the application in practice.

1 Challenges with metamodels, data formats, interfaces and mappings

In a Systems Engineering setting, a model of a System-Of-Interest often refers to a partial, but necessary, view or aspect of that system, in pursuit of a goal. This model is not always based on a theory and need not be in many cases, e.g., in creative, preliminary or general communication phases. However, when it is necessary to reason or compute on a model, the formal content (sometimes called semantics or business semantics) of that model should (mathematically) be defined in order to ensure sound reasoning. We shall see that such a definition is not enough to ensure that different implementations in modelling tools are equivalent or, ideally, functionally identical. In this paper, we focus on descriptive models – hereafter referred to simply as models, unless otherwise specified – by which we mean a model whose structure – equipped with input and output interaction points – encodes the main (abstract) structure of the thing to be modelled. We examine how descriptive models produced using a particular tool might be re-utilized, for instance for analysis or synthesis purposes. The question is to what extent descriptive models can be

[*]This work was carried out and funded in the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02). This work was co-funded by the Airbus Defence and Space company.

decoupled from the logic of their implementation by a particular tool while retaining their full meaning and being reusable.

One solution for making models transferable and interoperable between some tools is to impose the following:

i) a common metamodel, ontology, schema or modelling language;

ii) a common data format;

iii) a common interface.

Metamodels support, through the definition of syntax, the graphical representation of models and descriptions of their semantics. However, when models are used outside the tool in which they originated, some assumptions present in the metamodels may be missing. For instance, Statecharts [10], a visual modelling technique, have a plethora of existing semantics implemented within tools, falling into three main categories [8]. Regarding the implementation of semantics by modelling tools, a missing assumption relating to a semantics will either be implicit in the metamodel, or explicit but proper to the modelling tool. It has been argued in [18] that even for the same modelling formalism, that is to say a mathematical object, different implementations (via an abstract syntax and a semantics) may exist, and so the implementation of the same semantics is not guaranteed. In addition it has been pointed out in [5] that it is hard to mathematically classify the numerous modelling languages and techniques, and that a number of MDE (Model-Driven Engineering) approaches are not grounded on formal semantics.

As for the data format, it is intrinsically related to the structure of data of models. Using a particular data format is useful when persisting models. The data format does not influence the semantics of a model.

An interface can be useful for manipulating models, but the question of how to define this interface remains. As with the above-mentioned problems with metamodels, it is necessary to ensure that the internal implementation associating to an interface reflects the modelling formalism. Indeed, when several implementations coexist, it is not certain that they behave in the same way. In the latter case, the only solution might be to use the same implementation – like software libraries.

One might also think of mappings, translations or transformations between different metamodels, formats and interfaces. When feasible, this would imply the same problems – managing of mappings and even new metamodels, formats or interfaces acting as bridges – as with models. In some cases, although the magnitude of mappings or translations can be quadratic or linear (in the number of metamodels, for instance), it may be impossible (different tools specializing in different fields) or inappropriate (e.g. too expensive) to create or define a mapping. In order to address these various issues, this paper introduces and specifies a novel Federated Architecture (FA) for handling descriptive models.

In Section 2 we present our main idea, as well as a description of FA, and give some reasons (others are presented in Section 4) for adopting the category theory framework. In Section 3 we then give a formal definition of FA. In Section 4 we discuss the relevant related work. In Section 5, we present some concluding remarks.

2 Main idea: encapsulation-differentiation

It turns out from Section 1, that whatever the common interface, metamodel or data format used between tools, clarity and precision are not the only essential features. In today's software-intensive and computer-aided environments, the way in which models can be defined and conveyed is also essential since this has a direct impact on interoperability. We consider this a question that is better addressed at the level of the architecture of models. We adopt the ISO/IEC/IEEE 42010[1] standard, which defines the architecture of any system as the abstract description or the fundamental organization of that system, embodied in its components, the relations of those components to each other and to the environment, and the principles governing the design and evolution of that system.

FA is federated in that while it is based on a specific structure, it makes it possible to define domain specific-models via instantiations and interpretations – usages – of that structure. FA therefore unifies models on their structure and the links from the structure to its usages. FA equally differentiates models through domain-specific interpretations. Below we look at the basic organisation and principles governing the definition of a descriptive model.

Structure of a model Above we define a descriptive model as one whose structure – equipped with input and output interaction points – encodes the main (abstract) structure of the thing to be modelled or designed. This structure consists of boxes linked with incoming and outgoing wires. Input and output interaction points are ports associated to boxes ensuring connection with their surrounding environment. A representation of a tricky system can be managed using hierarchical decompositions and re-compositions.

Interpretations of the structure Since many systems are usually studied from different perspectives, namely: human, physical, structure, behaviour, cost, safety, etc., then different interpretations of their structure(s) are possible and necessary. At the level of models, these interpretations yield domain-specific languages or semantics. A structure hence constrains the definition of its interpretation, even if possible interpretations of the structure cannot be known by examining it per se. It may consequently be difficult, if not impossible, to impose or foresee all possible interpretations of the structure of a model. What is needed is a means of using that structure appropriately, and this starts with a clear and precise definition of that structure.

Instances of the structure Utilizing a structure involves data relating to an actual (or candidate) system that is modelled. These data should also relate to an interpretation of the structure, meaning that their precise meaning can be known and that they can be utilized.

The main components of FA are therefore (a) the model's structure; (b) the interpretations of the structure; (c) the corresponding instances of the structure. The architecture needs to support the ability (or to provide means) to define various interpretations of boxes. It does not model a concrete interpretation of them, but we will present some principles or guidelines for creating an interpretation. The same remark applies to instances of the structure.

Trivial example suppose the system is a research paper like the present one. A Structure of the paper comprises 8 boxes (that is to say, sections, including Abstract, References and Appendix). The paper itself is seen as a composite section that embodies these 8 sections. An Instance of this structure can be given by providing each section's title, size, or content. An interpretation of this structure might be a definition of what the size of (constituent and composite) boxes means. Note that this example does not consider the internal wiring pattern of composite boxes (sections).

Why the category theory framework? Category theory was created to unify and simplify mathematical systems [14] and has given rise to some universal constructions based on a relational viewpoint. This framework enables us to i) naturally define the notion of composite-box with its constituent-boxes via an arrow in a category; ii) abstract away details relating to a particular interpretation of the structure of models. At the same time, it allows a formal link to be specified between the structure of models and various targets like Instances and Interpretations. This formal link is ideally defined as a functor. We discuss further in Section 4.2, comparing and drawing on related work, why category theory seems to be the framework for appropriately defining FA. Besides the framework can guide us towards possible

[1]`http://ieeexplore.ieee.org/document/6129467/`

canonical extensions of FA with the enrichment of objects and category structure, see Section 5.

3 Components of FA

We define the structure of models in Section 3.1, and discuss the interpretation and usage of this structure in Section 3.2 and Section 3.3 respectively. To be able to define the structure of a model as a category, the objects will be the boxes and the arrows will show how boxes are linked to create composite boxes.

3.1 Structure of models

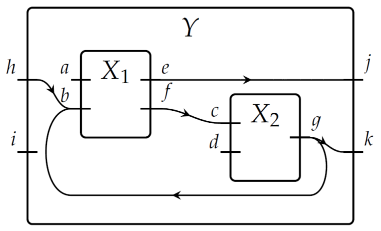

Figure 1: A box Y composed of 2 boxes X$_1$ and X$_2$

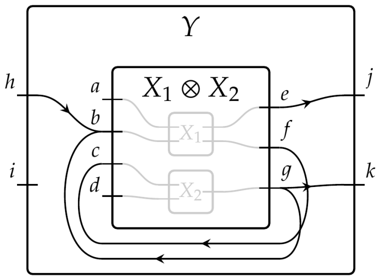

Figure 2: A box Y composed of $X_1 \otimes X_2$

Defining the structure of models as a symmetric multicategory denoted **Msc** is done in two stages. We start by defining a symmetric monoidal category (SMC) denoted (**Mscm**, $\otimes$, I) and prove that it is indeed an SMC. We then derive **Msc** using results from [12]. Since we will be using matrices to define the composition in **Mscm**, we may be able to derive a further symmetric multicategory denoted **MatMsc**, where arrows are given by matrices that might be useful for computations.

Why is it not enough simply to derive an SMC? Consider the box Y in Figure 1, composed of 2 boxes X_1 and X_2 . In an SMC Y can be seen to be composed of $X_1 \otimes X_2$ (i.e. there is an arrow $X_1 \otimes X_2 \to Y$), yielding Figure 2. Figure 1 is a lot clearer than Figure 2, in which the modularity (i.e. the clear distinction between X_1 and X_2 in Y) is not reflected graphically. Figure 1 corresponds to an arrow $X_1, X_2 \to Y$ in the multicategory. However, it is more convenient to use the SMC than the multicategory insofar as subscripts can be avoided.

The box and arrow representation that we use should not be confused with string diagrams [17]. String diagrams are a graphical language of categories [17] in which the objects and morphisms of categories are represented by wires and boxes respectively. In our paper, on the other hand, objects correspond to boxes, and morphisms correspond to how composite boxes are internally built (using wiring patterns) with its constituent boxes.

Figure 1 looks like a Wiring Diagram WD [19] [24], but in fact it differs from a WD in the following respects. In figure 1 we allow unconnected ports (ports without incoming or outgoing wires), converging wires (separate wires with the same target port), and diverging wires (same source port). Unconnected ports and converging wires are prohibited in a WD. Figure 1 would also allow several links between a source port and a target port. WDs may therefore be considered as special cases of figure 1 that include certain restrictions; this is discussed in Section 4.2. The point of allowing these various things is to encompass as many wiring patterns as possible that may occur in practice. For instance, it might be the case, structurally, that an actual component is not plugged into its surrounding environment via any of its interfaces. See for instance a simple Ptolemy model [23, Figure 9], similarly permissive.

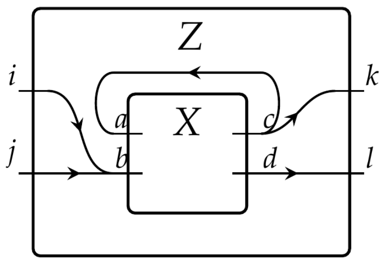

Figure 3: A box Z composed of X

Definition 3.1. The category **Mscm** has as constituents

• Objects $\mathbf{Mscm}_0$. An object $a \in \mathbf{Mscm}_0$ is a box and consists of a tuple $(in(a), out(a))$, where $in(a)$ and $out(a)$ are the finite sets of input (left) and output (right) ports of (the box) a such that $in(a) \cap out(a) = \emptyset$. We additionally require that **R1**: the set of ports (i.e. $in(a) \cup out(a)$) be pairwise disjoint for all objects in $\mathbf{Mscm}_0$.

• Arrows $\mathbf{Mscm}(b;a)$. An arrow $b \xrightarrow{\theta} a$ indicates how a box a is internally built from the box b. It consists of a tuple $(\theta^{in}, \theta^{out})$

$$\begin{array}{llcll} \theta^{in}: & L^{in} \to & in(b) & \times & (out(b) \cup in(a)) \\ \theta^{out}: & L^{out} \to & out(a) & \times & out(b) \end{array} \tag{1}$$

where L^{in} and L^{out} are the abstract finite sets of links coming respectively into an input port and an output port of one of the boxes b and a.

We require that **R2**: $L^{in}, L^{out} \in K$ and $L^{in} \cap L^{out} = \emptyset$. $(K, \cup, \cdot, \emptyset, \{\varepsilon\})$ is a semiring introduced below to define composition using matrices and detailed in Appendix (Section 6). We additionally require that **R3**: the domains (or more precisely $L^{in} \cup L^{out}$) of the set of morphisms be pairwise disjoint. R1, R2 and R3 ensure that objects and morphisms are all unambiguously defined. This will also be important in the definition of the monoidal product. Note also that (1) does not allow a link to go from an output port of a to one of its input ports. This is a modelling choice consistent with the fact that any box only exposes its input and output ports. An arrow is intended to give the internal wiring pattern of a composite box. A link that goes from an output port of a to one of its input ports should, rather, give rise to a new object a' and a new arrow $a \to a'$ which models a kind of self-feedback, see Figure 4.

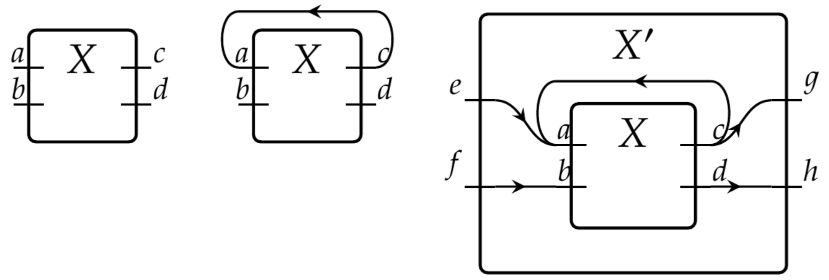

Figure 4: From left to right: a box X, an invalid-box X and an arrow $X \to X'$

In the same way, (1) does not allow a link to go directly from an input port of a to an output port. This is also a modelling choice that allows us to consistently zoom in on a composite box, avoiding the presence of isolated links when applying composition. If there were an isolated link, we would simply need to create a basic box with one input port and one output port and link them to the input and output ports respectively of a, such that this box outputs onto its output port the data that comes into its input port.

Example Consider Figure 3. We have two boxes X and Z given by $(\{a,b\},\{c,d\})$ and $(\{i,j\},\{k,l\})$ respectively. There is an arrow $X \to Z$ given by $(\{l_1 \mapsto (a,c), l_2 \mapsto (b,i), l_3 \mapsto (b,j)\}, \{l_4 \mapsto (k,c), l_5 \mapsto (l,d)\})$.

• Identities $\mathbf{Mscm}(a;a)$. An identity $a \xrightarrow{1_a} a$ consists of θ^{in} and θ^{out} given by $(1_a^{in}, 1_a^{out}) := (\{lin_i \mapsto (i,i), i \in in(a)\}, \{lout_j \mapsto (j,j), j \in out(a)\})$; i.e. links connect identical (input to input and output to output respectively) ports from the domain (a) to the codomain (a) of 1_a.

• Composition formula $\circ$. Composition enables a constituent of a composite box to be replaced by a more detailed constituent(s). For boxes $a,b,c \in \mathbf{Mscm}_0$, we have a function $\circ$:

$$\mathbf{Mscm}(b;a) \times \mathbf{Mscm}(c;b) \to \mathbf{Mscm}(c;a).$$

We must define an arrow $c \xrightarrow{\theta \circ \theta_1} a$. This amounts to finding a function $\theta_0 := \theta \circ \theta_1$, i.e. θ_0^{in} and θ_0^{out} following (1).

In order to define the composition formula, we use matrix operations. We start by establishing a connection between arrows and certain matrices over a semiring $(K, \cup, \cdot, \emptyset, \{\varepsilon\})$; see Appendix 6 for more details. We are now ready to work with the matrix representation of an arrow.

The data (1) of an arrow can be rewritten as follows (see Appendix 6). We define

$$\begin{array}{rrcl} M^{out}: & Y^{in} \times (Y^{out} \cup Z^{in}) & \to & 2^{L^{in}} \\ M^{in}: & Z^{out} \times Y^{out} & \to & 2^{L^{out}} \end{array} \tag{2}$$

where $Y^{in} = in(b)$, $Y^{out} = out(b)$, $Z^{in} = in(a)$ and $Z^{out} = out(a)$. 2^L is the powerset.

Let $X^{in} = in(c)$, $X^{out} = out(c)$.

We denote $\emptyset_M$ as the matrix with $\emptyset$'s everywhere and $\mathbb{I}_M$ the square matrix with $\{\varepsilon\}$'s on the main diagonal and $\emptyset$'s elsewhere.

Given (9) in Appendix 6 and the above notations, we are now ready to define arrow composition.

The map $\theta \circ \theta_1$ (i.e., $c \xrightarrow{\theta_0} a$ in $\mathbf{Mscm}$) of an arrow is rewritten as (O^{in}, O^{out}), given by the dashed arrows in the following diagrams. In accordance with Appendix 6 an arrow $I \xrightarrow{M} J$ in this diagram defines the matrix $M(I,J)$.

$$\begin{tikzcd} X^{in} \arrow[rr, dashed, "O^{in}"] \arrow[d, "N_{in}"'] & & Z^{in} \cup X^{out} \\ Y^{in} \cup X^{out} \arrow[rr, "M'^{in}"'] & & Z^{in} \cup Y^{out} \cup X^{out} \arrow[u, "N'^{out}"'] \end{tikzcd} \qquad \begin{tikzcd} Z^{out} \arrow[r, dashed, "O^{out}"] \arrow[d, "M^{out}"] & X^{out} \\ Y^{out} \arrow[ur, "N^{out}"'] & \end{tikzcd}$$

$$\begin{array}{rl} O^{out} = & M^{out} \times N^{out} \\ O^{in} = & N^{in} \times M'^{in} \times N'^{out} \end{array} \tag{3}$$

where N^{in}, N'^{out}, M'^{in} are respectively given by:

$$\begin{array}{cc} & Y^{in} \cup X^{out} \\ X^{in} & \left(\; N^{in} \; \right) \end{array} \qquad \begin{array}{c|cc} & Z^{in} & X^{out} \\ Z^{in} & \mathbb{I}_M & \emptyset_M \\ Y^{out} & \emptyset_M & N^{out} \\ X^{out} & \emptyset_M & \mathbb{I}_M \end{array} \qquad \begin{array}{c|cc} & Z^{in} \cup Y^{out} & X^{out} \\ Y^{in} & M^{in} & \emptyset_M \\ X^{out} & \emptyset_M & \mathbb{I}_M \end{array}$$

and M^{out}, N^{out} are respectively given by:

$$\begin{array}{cc} & Y^{out} \\ Z^{out} & (M^{out}) \end{array} \qquad \begin{array}{cc} & X^{out} \\ Y^{out} & (N^{out}) \end{array}$$

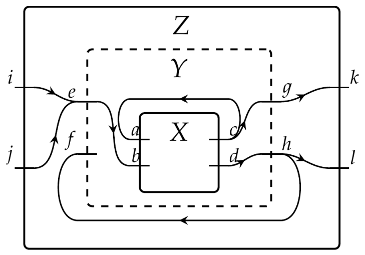

Figure 5: A box Z composed of Y, itself composed of X

Example Figure 5 is the representation of two arrows $X \xrightarrow{\theta_1} Y$ and $Y \xrightarrow{\theta} Z$. The arrow $X \xrightarrow{\theta\circ\theta_1} Z$ is given by the matrices O^{in} and O^{out} computed as follows. The matrices M^{out}, N^{out}, N^{in}, N'^{out} and M'^{in} are the following.

$$M^{out} = \begin{array}{c} \\ k \\ l \end{array}\begin{array}{c} \begin{array}{cc} g & h \end{array} \\ \begin{pmatrix} \{l_1\} & \emptyset \\ \emptyset & \{l_2\} \end{pmatrix} \end{array} \quad N^{out} = \begin{array}{c} \\ g \\ h \end{array}\begin{array}{c} \begin{array}{cc} c & d \end{array} \\ \begin{pmatrix} \{l_3\} & \emptyset \\ \emptyset & \{l_4\} \end{pmatrix} \end{array} \quad N^{in} = \begin{array}{c} \\ a \\ b \end{array}\begin{array}{c} \begin{array}{cccc} e & f & c & d \end{array} \\ \begin{pmatrix} \emptyset & \emptyset & \{l_5\} & \emptyset \\ \{l_6\} & \emptyset & \emptyset & \emptyset \end{pmatrix} \end{array}$$

$$N'^{out} = \begin{array}{c|cccc} & i & j & c & d \\ \hline i & \{\varepsilon\} & \emptyset & \emptyset & \emptyset \\ j & \emptyset & \{\varepsilon\} & \emptyset & \emptyset \\ g & \emptyset & \emptyset & \{l_3\} & \emptyset \\ h & \emptyset & \emptyset & \emptyset & \{l_4\} \\ c & \emptyset & \emptyset & \{\varepsilon\} & \emptyset \\ d & \emptyset & \emptyset & \emptyset & \{\varepsilon\} \end{array} \quad M'^{in} = \begin{array}{c|cccccc} & i & j & g & h & c & d \\ \hline e & \{l_7\} & \{l_8\} & \emptyset & \emptyset & \emptyset & \emptyset \\ f & \emptyset & \emptyset & \emptyset & \{l_9\} & \emptyset & \emptyset \\ c & \emptyset & \emptyset & \emptyset & \emptyset & \{\varepsilon\} & \emptyset \\ d & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{\varepsilon\} \end{array}$$

It is easy to check that $O^{out} = M^{out} \times N^{out}$ and $O^{in} = N^{in} \times M'^{in \times} N'^{out}$ are given by the following matrices:

$$O^{out} = \begin{array}{c|cc} & c & d \\ \hline k & \{l_1 l_3\} & \emptyset \\ l & \emptyset & \{l_2 l_4\} \end{array} \quad O^{in} = \begin{array}{c|cccc} & i & j & c & d \\ \hline a & \emptyset & \emptyset & \{l_5\} & \emptyset \\ b & \{l_6 l_7\} & \{l_6 l_8\} & \emptyset & \emptyset \end{array}$$

The composition in this example enables us to zoom in on Z, in other words to know how Z is built from X, itself a constituent box of Y. After the composition the link in Figure 5 (a representation of the two arrows $X \xrightarrow{\theta_1} Y$ and $Y \xrightarrow{\theta} Z$) from the port h of Y to its port f, vanishes in Figure 3 (the arrow $X \xrightarrow{\theta\circ\theta_1} Z$) resulting from the composition. There are two reasons for this: the port f is not connected in $X \xrightarrow{\theta_1} Y$, and the way $\cdot$ acts on links (or elements of K) in the semiring $(K, \cup, \cdot, \emptyset, \{\varepsilon\})$. In practice this could mean that f is not connected, even though it appears to be.

Proposition 3.1. *The identity law holds for the constituents of* ***Mscm****.*

Proof. The matrices corresponding to $\theta \circ 1_b$ and $1_a \circ \theta$ are seen to be equal to the matrix corresponding to θ, whenever $\theta : b \to a$ is an arrow. □

Proposition 3.2. *The associativity law holds for the constituents of* ***Mscm***.

Proof. The associativity law follows from the associativity of matrix multiplication. □

Now we define the monoidal product of **Mscm**.

Definition 3.2. Let X_1, X_2, Y_1, Y_2 and $\theta_1 : X_1 \to Y_1$, $\theta_2 : X_2 \to Y_2$ be objects and arrows respectively of **Mscm**. The monoidal product (or tensor product) $\otimes$: **Mscm** $\times$ **Mscm** $\to$ **Mscm** is given by:

$$X_1 \otimes X_2 := (in(X_1) \cup in(X_2), out(X_1) \cup out(X_2)), \quad \theta_1 \otimes \theta_2 := \theta_1 \cup \theta_2 \tag{4}$$

The unit I is the box without input and output ports i.e. I:=$(\emptyset,\emptyset)$. R2 and R3 ensure that $\theta_1 \otimes \theta_2$ is well-defined.

Note here that the effect of the tensor on different objects and arrows is simply to stack them straightforwardly and without losing (thanks to R1, R2 and R3) the origins of ports and links. For instance, stacking the unit I with any non-unit object X will have no effect on X, since the box corresponding to I does not have any input or output ports. The effect of tensor on the same object (resp. arrow) is: stacking a box (resp. an arrow) with itself has no effect, i.e., results in the same object (resp. arrow).

Proposition 3.3. $\otimes$: ***Mscm*** $\times$ ***Mscm*** $\to$ ***Mscm*** *is a (bi)functor.*

Proof. $\otimes$ preserves identity morphisms and composition of morphisms. It suffices to write the corresponding morphisms, i.e. let $X_1, X_2, Y_1, Y_2, Z_1, Z_2$ and $\alpha_1 : Y_1 \to Z_1$, $\alpha_2 : Y_2 \to Z_2$, $\theta_1 : X_1 \to Y_1$, $\theta_2 : X_2 \to Y_2$ be objects and arrows respectively of **Mscm**

Identity $\otimes(1_{X_1}, 1_{X_2}) = 1_{X_1} \cup 1_{X_2} = X_1 \cup X_2 \xrightarrow{1_{X_1 \cup X_2}} X_1 \cup X_2 = 1_{\otimes(X_1,X_2)}$

Composition $\otimes(\alpha_1 \circ \theta_1, \alpha_2 \circ \theta_2) = \otimes(X_1 \xrightarrow{\alpha_1 \circ \theta_1} Z_1, X_2 \xrightarrow{\alpha_2 \circ \theta_2} Z_2) =$
$X_1 \cup X_2 \xrightarrow{(\alpha_1 \circ \theta_1) \cup (\alpha_2 \circ \theta_2)} Z_1 \cup Z_2$ and
$\otimes(\alpha_1, \alpha_2) \circ \otimes(\theta_1, \theta_2) = (Y_1 \cup Y_2 \xrightarrow{\alpha_1 \cup \alpha_2} Z_1 \cup Z_2) \circ (X_1 \cup X_2 \xrightarrow{\theta_1 \cup \theta_2} Y_1 \cup Y_2) =$
$X_1 \cup X_2 \xrightarrow{(\alpha_1 \circ \theta_1) \cup (\alpha_2 \circ \theta_2)} Z_1 \cup Z_2$

□

Proposition 3.4. *The category (**Mscm**,⊗,I) with the constituents given in Definition 3.1 and the tensor product given in Definition 3.2 is a symmetric monoidal category.*

Proof. We have already shown that **Mscm** is a category (see Definition 3.1, Proposition 3.2 and Proposition 3.2). It remains to show that the tensor product is effectively a symmetric monoidal structure on **Mscm**.

Given $X_1, X_2, X_3 \in \mathbf{Mscm}_0$, we have

$$\begin{aligned} (X_1 \otimes X_2) \otimes X_3 = X_1 \otimes (X_2 \otimes X_3) &\quad (associativity) \\ \mathrm{I} \otimes X_1 = X_1 = X_1 \otimes \mathrm{I} &\quad (left \text{ and } right\ units) \\ X_1 \otimes X_2 = X_2 \otimes X_1 &\quad (commutativity) \end{aligned} \tag{5}$$

The associator, unitors and inverse map (braiding) are identities. □

Proposition 3.5. ***Msc*** *is an underlying symmetric multicategory of (**Mscm**,$\otimes$). Its objects are the same objects as **Mscm**, and its arrows are the set **Msc**$(a_1,....,a_n;a)$. An arrow $a_1,....,a_n \to a$ is defined as an arrow $a_1 \otimes \cdots \otimes a_n \to a$ in **Mscm**.*

Proof. *Any monoidal category (A,$\otimes$) has an underlying multicategory C* [12, Example 2.1.3]. Moreover, *any symmetric monoidal category is naturally a symmetric multicategory, via the symmetry maps $\sigma \cdot - :$ $a_{\sigma(1)} \otimes \cdots \otimes a_{\sigma(n)} \xrightarrow{\sim} a_1 \otimes \cdots \otimes a_n$*.[12, Definition 2.2.21] □

Example In Figure 1, there is an arrow $X_1,X_2 \to Y$ in **Msc**, which is defined as $X_1 \otimes X_2 \to Y$ in (**Mscm**,$\otimes$).

Since we rely on matrix multiplication to define composition in **Mscm**, we can redefine the components of **Msc** using matrix without altering their initial meaning. This might be useful to automate computations with morphisms. Again a connection between morphisms of **Mscm** and certain matrices over a semiring is presented in Appendix 6.

Definition 3.3. **MatMsc** is an underlying symmetric multicategory of (**Mscm**,$\otimes$).

• Its objects are the same objects as **Mscm**. An object X is rewritten as (X^{in},X^{out}).

• Its arrows **MatMsc**$(a_1,....,a_n;a)$. An arrow $a_1,....,a_n \to a$ is rewritten as follows. Let $b = a_1 \otimes \cdots \otimes a_n$, $X^{in} = in(b)$, $X^{out} = out(b)$, $Y^{in} = in(a)$ and $Y^{out} = out(a)$, the arrow $b \to a$ in **Mscm**, noted $X \to Y$ is given by the couple of matrices (M^{in},M^{out}) where $M^{in}$and M^{out} are given by the functions $X^{in} \times (Y^{in} \cup X^{out}) \to K$ and $Y^{out} \times X^{out} \to K$ respectively. Objects and arrows are subject to R1, R2 and R3 as in **Mscm**.

• Identities **MatMsc**$(a;a)$. For any object (X^{in},X^{out}), the identity arrow
$1_X := (1^{in}_X : X^{in} \times (X^{in} \cup X^{out}) \to K, 1^{out}_X : X^{out} \times X^{out} \to K)$, rewritten as follows. 1^{in}_X yields

$$\begin{array}{cc} & \begin{array}{cc} X^{in} & X^{out} \end{array} \\ X^{in} & \begin{pmatrix} \mathbb{I}_M & \emptyset_M \end{pmatrix} \end{array}$$

and 1^{out}_X yields

$$\begin{array}{cc} & X^{out} \\ X^{out} & \begin{pmatrix} \mathbb{I}_M \end{pmatrix} \end{array}$$

• The composition formula is the same as for **Mscm**, but rewritten as follows. Consider the objects X, Y, Z and arrows $n : X \to Y$ and $m : Y \to Z$ given by matrices (N^{in},N^{out}) and (M^{in},M^{out}) respectively. The arrow $m \circ n : X \to Z$ is given by (O^{in},O^{out}):

$$\begin{aligned} O^{in} &= N^{in} \times M'^{in} \times N'^{out} \\ O^{out} &= M^{out} \times N^{out} \end{aligned}$$

The identity law holds in **MatMsc** with this rewriting (already shown in relation to **Mscm**) i.e. $m \circ 1_X = m = 1_Y \circ m$ whenever $m : X \to Y$ is an arrow. The matrices defining $m \circ 1_X$ and $1_Y \circ m$ are both equal to M i.e. (M^{in},M^{out}). The arrow $m \circ 1_X$ associates M to m and N to 1_X. Therefore O^{in} equals $N^{in} \times M'^{in} \times N'^{out} =$

$$\begin{array}{cc} & \begin{array}{cc} X^{in} & X^{out} \end{array} \\ X^{in} & \begin{pmatrix} \mathbb{I}_M & \emptyset_M \end{pmatrix} \end{array} \times \begin{array}{cc} & \begin{array}{cc} Y^{in} \cup X^{out} & X^{out} \end{array} \\ \begin{array}{c} X^{in} \\ X^{out} \end{array} & \begin{pmatrix} M^{in} & \emptyset_M \\ \emptyset_M & \mathbb{I}_M \end{pmatrix} \end{array} \times \begin{array}{cc} & \begin{array}{cc} Y^{in} & X^{out} \end{array} \\ \begin{array}{c} Y^{in} \\ X^{out} \\ X^{out} \end{array} & \begin{pmatrix} \mathbb{I}_M & \emptyset_M \\ \emptyset_M & \mathbb{I}_M \\ \emptyset_M & \mathbb{I}_M \end{pmatrix} \end{array}$$

O^{in} is indeed equal to M^{in}. We also have O^{out} equals $M^{out} \times N^{out} =$

$$\begin{array}{cc} & X^{out} \\ Y^{out} & (M^{out}) \end{array} \times \begin{array}{cc} & X^{out} \\ X^{out} & (\ \mathbb{I}_M\ ) \end{array}$$

O^{out} is equal to M^{out}. Hence, $m \circ 1_X = m$. In the same way: $m = 1_Y \circ m$.

MatMsc can be seen as the category where objects declare interfaces and where morphisms declare how composite interfaces are internally built from other interfaces. The composition formula lets us safely zoom in and zoom out on levels further down and up in the hierarchy of interfaces.

It is worth noting that with only **MatMsc** (or **Msc** or **Mscm**), no meaning (related to a domain of interest or business domain) is assumed for an interface, its ports and, if applicable, its internal links.

3.2 Interpretations of the structure

The present section proposes an approach that seeks to bridge the gap between a possible semantics (interpretation of the structure) and its implementation in practice. A semantics corresponding to an interpretation of the structure may be formally defined in a variety of ways (denotational, axiomatic, operational, etc.) and with numerous possibilities of implementation. This paper is not concerned with defining or implementing a particular semantics, but sets out principles for formally connecting the structure, semantics and their implementation. As mentioned above, the architecture of a system is not about the effective (or final) implementation of the system.

Using the structure of models, it is possible to specify an interpretation of boxes such that the associated semantics can be recovered. An actual interpretation could therefore be independently developed by using the structure of models. Ideally, the specification of the interpretation is given by a functor F : **Msc** $\rightarrow$ **Msin**, **Msin** being the category in which the structure is interpreted. The caveat 'ideally' is necessary, because it may not always be straightforward to define a functorial semantics.

Examples

(i) Consider once again a model corresponding to the box shown in Figure 1. A domain-specific meaning of this model can only be known once the definition of an interpretation of its structure is given. Let us suppose that the domain of interest is classical physics and that we wish to study some physical properties of boxes. In this case, F can be defined as F : **Msc** $\rightarrow$ **Msin**. Let **Msin** :=**Set**. Along with objects, set of real numbers, and arrows, n-ary functions define the relationship between a physical property of a box and the physical properties of its n constituent boxes. F is a map of multicategories and an algebra for **Msc** (See [12, Definition 2.1.9, Definition 2.1.12]). Here, let us take *mass* to be the physical property under consideration.

Let F be $F_0 : \{X_1, X_2, Y\} \rightarrow \mathbb{R}_{>0}$ given by $X \mapsto ma$. The image of $X_1, X_2 \rightarrow Y$ is the function $F_0(X_1) \times F_0(X_2) \rightarrow F_0(Y)$ given by $(ma_1, ma_2) \mapsto ma$.

Depending on the meaning of links, ma can be equal either to (*) $ma_1 + ma_2$ or to (**) $ma_1 + ma_2 + m_L$, $m_L \in \mathbb{R}_+$. In the first case (*), the mass of a composite box will depend solely on the masses of its constituent boxes. In the second case (**), the mass of a box will additionally depend on m_L, which is an additional mass emerging from internal connections.

Similar interpretations can be made for other physical properties.

(ii) Consider the arrow in Figure 6. Let each of the boxes $X_i, i = 1..4$ be interpreted as a *two-digit adder* ($d_0 + d_1 = cs$) which returns the sum (s) and the carry (c) via the top and bottom output ports respectively, given d_0 and d_1 obtained from top and bottom input ports. Here, the containing box Y can

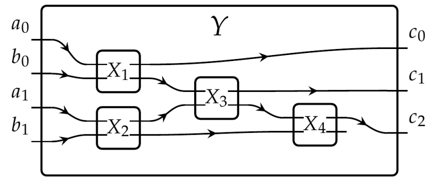

Figure 6: $X_1,X_2,X_3,X_4 \to Y$

be interpreted as *two-digit numbers adder* ($a_1a_0+b_1b_0=c_2c_1c_0$). There is a category (**Msin**) with five objects and one arrow.

The concept of mass in (i), and the concepts of two-digit adder and two-digit numbers adder in (ii), must be exposed by F, irrespective of how these concepts are defined in **Msin** and irrespective of the implementation. These data must be machine-accessible. It is worth noting that the functorial semantics of wiring diagrams (i.e. special cases of **Msc**) defined in [16], [19], [24] and [20] are concerned with the interconnections of discrete-time processes, databases and plug-and-play circuits, differential equations of open dynamical systems, and mode-dependent networks respectively. In each case, a particular algebra is devised which may be implemented as definitions of interpretations of the structure of models.

3.3 Usages of the structure

If the structure of a model is to be automatically exchangeable, then in the same way that the definition of an interpretation of the structure needs to be machine-accessible, the related data must also be machine-accessible.

The data relating to an actual (or candidate) system modelled is defined by a Set-valued functor $I : \mathbf{Msc} \to \mathbf{Set}$. This links an object $a \in \mathbf{Msc}_0$ to the set of its usages $I_0(a)$, and any arrow $a_1,...,a_n \xrightarrow{\theta} a \in C(a_1,...,a_n;a)$ to the function $I_0(a_1)\times ... \times I_0(a_n) \xrightarrow{I(\theta)} I_0(a)$. Taking the composition and the identity in the same way as in **Set**, I is a map of multicategories [12, Definition 2.1.9] and an algebra for **Msc** [12, Definition 2.1.12]. The usefulness of I stems from the fact that where the structure (**Msc**) of a model, together with at least one interpretation (F) of this structure, has been established, this is still not quite enough to allow the model to be used. The actual business data corresponding to the components of the structure are also required. For instance, consider the box shown in Figure 1 and the interpretation given in example (i) above. How do we communicate the fact that the masses of boxes X_1 and X_2 and Y are given by actual values? No assumptions are made concerning the relations between the different masses. A parallel can be drawn with databases: it is good to have a database schema and to know what tables correspond to in the real world, but in order to use a database we also need to know the contents of the tables (the actual data entries). The purpose of I is to answer this kind of question, by providing instances of the components of a structure of a model.

Let us recapitulate the purpose of the present section. We have provided an informal and a formal understanding of the proposed architecture FA. To be able to read a model (data), it is necessary to have at least the specification of $I : \mathbf{Msc} \to \mathbf{Set}$, while to be able to utilize a model (semantics), it is necessary to have at least the specification of $F : \mathbf{Msc} \to \mathbf{Msin}$.
I and F are intended to facilitate the transfer of the modularity of **Msc** in different interpretations of **Msc**. Without functoriality, the modularity and the generality offered by the category-theoretic framework are lost (even though this functoriality is not always easily perceived, and may in some cases be unnecessary).

For exploiting models through a given automated routine, it is necessary to specify

$$\mathbf{Msin} \xleftarrow{F} \mathbf{Msc} \xrightarrow{I} \mathbf{Set} \tag{6}$$

and make the definition of interpretation machine-accessible. Nonetheless, wherever the usage of a structure is linked to a particular interpretation of that structure, it can be argued that the effective data related to **Msc** should instead be given by an amalgamated sum (a pushout) over (6).

4 Related research

FMI[2] (Functional Mockup Interface), the standard which attempts to solve the problem stated in Section 1, is discussed in Section 4.1. Then other relevant related works that can be correlated to FA are discussed in Section 4.2.

4.1 FMI

FMI is a tool-independent interface that allows models to be exchanged and inter-operated in a co-simulation. An FMU (Functional Mock-up Unit), one of the components of FMI, must implement the FMI API via XML files and C-code. In FMI a model is associated with a particular FMU, which is generated using different tools. The kinds of models covered by FMI are (co) simulation models [1]. This paper is not primarily concerned with how models co-operate. Nevertheless, the way a model is defined (such as via an FMU) will influence how it can be reused.

An FMU separates the description of interface data (via an XML – Extensible Markup Language – file) from its mode of operation (via a program written in the C programming language). The interface data describes the static elements of the model such as type definitions (Real, Integer etc.), model variables, model structure (an ordered list of outputs termed *Outputs*, states termed *Derivatives*), and some initially unknown data termed *InitialUnknowns*), along with units etc.
The C-code and header files define the functions to be implemented by an FMU. These functions are used to simulate a model described according to a particular FMU. Generally speaking, the functions are for purposes such as initializing the FMU, assigning values to an input variable, retrieving the value of an output variable, changing the state of the FMU, etc. More information about the evolving specification of the FMI standard and supported tools can be found on the FMI website.

Although the FMI standard provides an interface to be implemented by FMUs, one concern is that there can be semantic gaps (studied in [21] [4]) between different model semantics (e.g, discrete events, dataflow models) or modelling languages and the target interface [22]. The semantics of the target interface has been considered as a timed Mealy machine [21]. Another concern is the extent to which the interface can be used to capture the semantics of different kinds of models [22]. Attempts to address this problem have resulted in various (proposed) adaptations of FMI [4] [3]. Therefore, the specification and definition of the common interface are crucial. The FMI standard is mainly concerned with the exchange of models and their simulation as black boxes.

In this paper we adhere to the FMI standard in separating structure and function. But, unlike in FMI, we give a formal definition of FA. The structures of models are considered as composite components defined within a category-theoretic framework, and a predefined interface for the implementation of functions is not imposed. Any implementation has to be derived from an interpretation of the structure

[2] https://www.fmi-standard.org/

of models. The (software) code corresponding to an implementation must expose the functions whose inputs and outputs are correlated to an interpretation of the structure. The data corresponding to the instantiations (i.e. actual serialized models) of the structure are elements that should be persisted as a file.

4.2 Other works

The kind of descriptive model that we are talking about is close, in terms of model structure (or abstract syntax), to the component-based, modular, and even composite-like models that are habitually encountered in system and software modelling, simulation, and more generally, analysis. See for instance [9] [23] for approaches and tools that deal with this kind of model structure. Although these approaches address modelling and simulation, their ultimate aim is not to solve the problem that is our concern here. They generally introduce their own approach (mainly on modelling formalisms) to system modelling and analysis. Metamodels for this kind of structure have been presented by [11] [9]. Their approaches advocate hierarchical and component-based design in systems modelling and engineering.

From a practical (yet formal) point of view, a box (a component of FA) is analogous to an *actor* in [23]. In [23], boxes are encoded using the logic of the modelling tool Ptolemy and the Java programming language. This may hamper the reuse of models, albeit some of these models may be imported using FMI.

In the present paper, unlike in [23], a box does not correspond to a particular semantics (i.e. the target of an interpretation of the structure). For [23], boxes or actors are seen as extended timed state machines, since the authors' focus is on behavioural models. The corresponding semantics is also called the actor interface, defined in a way that includes both the structure and the semantics of the box. However, the actor interface can also be considered as the target of an interpretation of the structure of a box, since the input and output ports of a box are explicit in this interface. In [23] composite actors are basically considered as a set of actors, whereas for us a composite box is an arrow in **Msc** (see Section 3). It will be noted that [23] is mainly concerned with a unified description of the behavioural semantics of a composite that comprises more than one actor.

It might be suggested that graphs are a way of formalizing the structure of models. Although graphs are very useful in representing interconnected objects, they lack the sort of intrinsic feature for dealing with the hierarchical composite structure of vertices (in the present context vertices equate to boxes) that we require. In order to represent this kind of structure, the graph would need to include additional data, which might complicate the formalization.

In fact, hierarchical graphs have also been formalized in category theory to model hierarchical structures [15] (where coupling of hyper-graphs and forests yields bigraphs) [2] (graphs inside graphs). Although hierarchical graphs encapsulate the hierarchy of nodes (here, boxes) and the communication links between nodes, the nature and structure of nodes and node containment (nesting) and links between nodes do not have the same meaning as described in Section 2. In the present work a node is given by input and output ports. The source and target of a link are ports at the same level of hierarchy of a box. We do not use two separate structures to define boxes (objects) and as a basis for composite boxes (arrows), and in this our approach contrasts with the two-structure approaches of [15] (place graph and linking graph) and [2] (graph layer and hierarchical graph).

Our inspiration for the category structure considered in this paper is, rather, to be found in [19] [16], [24] and [13]. [13] considers symmetric monoidal and compact closed categories, with associated logics and proof theories as a possible formal foundation for systems engineering, given that it is compatible with current modelling languages and tools and encompasses existing formalisms. The mathematical object presented in this paper for encapsulating the structure of models, a symmetric multicategory (see Section 3), is closely related to the Wiring Diagrams (WD) structure developed in a series of papers: [16], [19], [24] etc. The two big differences between the WD structure and our structure are the following:

(i) Our structure is a more general construction supported by the "wiring" or connection pattern, i.e. almost all possible ways of connecting sub-components (or constituents) of a composite component are allowed, unlike WD (see Section 3.1 below for details of what WD does not allow). A WD is basically given by

$$(X^{in}+X^{out}) \to (Y^{in}+X^{out}) \leftarrow (Y^{in}+Y^{out}) \tag{7}$$

which can be decomposed into two functions

$$\begin{aligned} \phi^{in} &: \quad X^{in} \to Y^{in}+X^{out} \\ \phi^{out} &: \quad Y^{out} \to X^{out} \end{aligned} \tag{8}$$

To obtain (1) – our definition – from (8), the elements of ϕ^{in} and ϕ^{out} must be associated to the abstract sets of links or wires L^{in} and L^{out}. To get (8) from (1) it is necessary to have θ^{in} and θ^{out} of (1) that are injective. The elements of L^{in} and L^{out} can be seen as the labels of wires.

(ii) We are not concerned with a particular interpretation (e.g. an algebra) of the structure of models.

5 Summary and perspective

Our aim was to provide a federated architecture for descriptive models in order to facilitate their development, sharing and utilization. Though clear and precise definitions of models are essential, how such definitions are conveyed in practice, particularly in distributed, collaborative and computer-aided environments, is also essential, and crucial to some extent. In order to bridge the gap between theoretical and practical considerations, a federated architecture (FA) was presented via the specification: $\mathbf{Msin} \xleftarrow{F} \mathbf{Msc} \xrightarrow{I} \mathbf{Set}$ (6). Categorical settings are useful for abstracting away from details, and for differentiating and correlating structures of models and their usages. Since there are well-known basic data structures, algorithms and software libraries related to matrices, their use may foster calculations with FA. Setting up this framework can make it possible to communicate and utilize different descriptive models, transparently or independently of the tool used to build them and without sacrificing the definitions of models.

One question not addressed in this paper is the way in which models change and evolve over time. This aspect might be manageable canonically using the structure of models. It would be possible to include within a structure an ‘off’ object, the ‘black hole’ (to model a destroyed box) yielding a new structure K. A functor from a category K_t at time t to the same category $K_{t'}$ at time $t' > t$ could then be defined. By taking the time instants as objects in a category T featuring some order, and defining a functor from T to K, we would then be in a position to address change and evolution in models with the structure K[7].

References

[1] Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel et al. (2012): *Functional mockup interface 2.0: The standard for tool independent exchange of simulation models.* In: *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, 076, Linköping University Electronic Press, pp. 173–184.

[2] Roberto Bruni, Andrea Corradini, Fabio Gadducci, Alberto Lluch-Lafuente & Ugo Montanari (2011): *Hierarchical Models for Service-Oriented Systems*. In: *Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing*, pp. 349–368, doi:10.1007/978-3-642-20401-2_16. Available at `https://doi.org/10.1007/978-3-642-20401-2_16`.

[3] Fabio Cremona, Marten Lohstroh, David Broman, Edward A Lee, Michael Masin & Stavros Tripakis (2019): *Hybrid co-simulation: it's about time*. *Software & Systems Modeling* 18(3), pp. 1655–1679.

[4] Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks & Edward A Lee (2016): *FIDE: an FMI integrated development environment*. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ACM, pp. 1759–1766.

[5] Zinovy Diskin & Tom Maibaum (2014): *Category theory and model-driven engineering: from formal semantics to design patterns and beyond*. *Model-Driven Engineering of Information Systems: Principles, Techniques, and Practice*, p. 173.

[6] Manfred Droste & Werner Kuich (2009): *Semirings and Formal Power Series*, pp. 3–28. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-642-01492-5_1.

[7] Andrée C Ehresmann & J-P Vanbremeersch (1987): *Hierarchical Evolutive Systems: A mathematical model for complex systems*. *Bulletin of Mathematical Biology* 49(1), pp. 13–50.

[8] Rik Eshuis (2009): *Reconciling statechart semantics*. *Science of Computer Programming* 74(3), pp. 65–99.

[9] Cécile Hardebolle & Frédéric Boulanger (2008): *ModHel'X: A component-oriented approach to multi-formalism modeling*. In: *Models in Software Engineering*, Springer, pp. 247–258.

[10] David Harel (1987): *Statecharts: A visual formalism for complex systems*. *Science of computer programming* 8(3), pp. 231–274.

[11] Edward A Lee (2010): *Disciplined heterogeneous modeling*. In: *Model Driven Engineering Languages and Systems*, Springer, pp. 273–287.

[12] Tom Leinster (2003): *Higher Operads, Higher Categories*. *arXiv preprint math 0305049*.

[13] Dominique Luzeaux (2015): *A Formal Foundation of Systems Engineering*. In: *Complex Systems Design& Management*, Springer, pp. 133–148.

[14] Saunders Mac Lane (1998): *Categories for the working mathematician. 2nd ed.*, 2nd ed edition. New York, NY: Springer.

[15] Robin Milner (2008): *Bigraphs and Their Algebra*. *Electr. Notes Theor. Comput. Sci.* 209, pp. 5–19, doi:10.1016/j.entcs.2008.04.002. Available at `https://doi.org/10.1016/j.entcs.2008.04.002`.

[16] Dylan Rupel & David I Spivak (2013): *The operad of temporal wiring diagrams: formalizing a graphical language for discrete-time processes*. *arXiv preprint arXiv:1307.6894*.

[17] Peter Selinger (2010): *A survey of graphical languages for monoidal categories*. In: *New structures for physics*, Springer, pp. 289–355.

[18] Sanjit A Seshia, Natasha Sharygina & Stavros Tripakis (2014): *Modeling for verification*. *Handbook of Model Checking, EM Clarke, T. Henzinger, and H. Veith, Eds. Springer*.

[19] David I Spivak (2013): *The operad of wiring diagrams: Formalizing a graphical language for databases, recursion, and plug-and-play circuits*. *arXiv preprint arXiv:1305.0297*.

[20] David I Spivak (2015): *Nesting of dynamic systems and mode-dependent networks*. *arXiv preprint arXiv:1502.07380*.

[21] Stavros Tripakis (2015): *Bridging the semantic gap between heterogeneous modeling formalisms and FMI*. In: *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 International Conference on*, IEEE, pp. 60–69.

[22] Stavros Tripakis (2016): *Compositionality in the Science of System Design*. *Proceedings of the IEEE* 104(5), pp. 960–972.

[23] Stavros Tripakis, Christos Stergiou, Chris Shaver & Edward A Lee (2013): *A modular formal semantics for Ptolemy*. Mathematical Structures in Computer Science 23(04), pp. 834–881.

[24] Dmitry Vagner, David I Spivak & Eugene Lerman (2015): *Algebras of open dynamical systems on the operad of wiring diagrams*. Theory and Applications of Categories 30(51), pp. 1793–1822.

6 Appendix

Connection between morphisms of Mscm of and certain matrices over a semiring

The connection assumed in Definition 3.1 is established and proved in three steps as follows. (1) an equivalence between certain functions and matrices over a semiring is proved. Then (2) certain functions are shown to be equivalent to the morphisms of **Mscm**, by rewriting the signatures of morphisms' component functions (f^{in} and f^{out}). Finally, it is deduced (3) that a morphism of **Mscm** yields a matrix being one of these certain matrices and the converse. Computations relating to morphisms are then made via matrices. Finally, matrix multiplication is used to obtain the composition formula of **Mscm**.

Let K be the power set of the set of strings of finite lengths. $(K,\cup,\cdot,\emptyset,\{\varepsilon\})$ is a semiring [6] where $\times := \cdot$ is the product induced by the string concatenation operator, $+ := \cup$ is the addition given by union of sets of strings of finite lengths, $0 := \emptyset$ is the zero given by the empty set, and $1 := \{\varepsilon\}$ is the unit given by the singleton set containing the empty string. Let also I and J be the finite sets with n and m elements respectively.

Proof of (1). Let $f : I \times J \to K$ be a function. Then $M(n,m)$ is a matrix, given by $M = a_{ij}, a_{ij} \in K, i = 1..n, j = 1..m$. Conversely, given a matrix $M = a_{ij}, a_{ij} \in K, i = 1..n, j = 1..m$, the corresponding function is $f : I \times J \to K$ given by $(i,j) \mapsto a_{ij}$. This completes the proof of (1). □

Proof of (2). The data (1) of an arrow can be rewritten as follows. We define two functions f^{out}, f^{in}

$$\begin{array}{rrcl} f^{out}: & Y^{in} \times (Y^{out} \cup Z^{in}) & \to & 2^{L^{in}} \\ f^{in}: & Z^{out} \times Y^{out} & \to & 2^{L^{out}} \end{array} \tag{9}$$

where $Y^{in} = in(b)$, $Y^{out} = out(b)$, $Z^{in} = in(a)$ and $Z^{out} = out(a)$. 2^L is the power set. Like the functions θ^{in} and θ^{out} that are total functions, f^{out} and f^{in} are also total functions given by associating with each element α of their domain the set of elements that are each the preimage of α in θ^{in} and θ^{out} respectively. From (1), we have $2^{L^{in}}, 2^{L^{out}} \subset K$. Hence, the functions θ^{in} and θ^{out} are equivalent via (f^{out} and f^{in}) to certain functions $f : I \times J \to K$. This proves (2). □

Proof of (3). (3) follows from (1) and (2). □

Henceforth, matrix multiplication is used for the composition formula of **Mscm**.

Example Let us consider $f_A : I_1 \times S \to K$, and $f_B : S \times I_2 \to K$ where $I_1 = \{a,b\}$, $S = \{c,d\}$ and $I_2 = \{e,f\}$.
Suppose given
$f_A := \{(a,c) \mapsto \{w_1\}, (a,d) \mapsto \emptyset, (b,c) \mapsto \{w_2,w_3\}, (b,d) \mapsto \emptyset\}$
$f_B := \{(c,e) \mapsto \{w_5\}, (c,f) \mapsto \emptyset, (d,e) \mapsto \emptyset, (d,f) \mapsto \emptyset\}$

The matrices A and B corresponding to the functions f_A and f_B are as follows.

$$A = \begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{cc} c & d \end{array} \\ \begin{pmatrix} \{w_1\} & \emptyset \\ \{w_2, w_3\} & \emptyset \end{pmatrix} \end{array} \quad B = \begin{array}{c} \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cc} e & f \end{array} \\ \begin{pmatrix} \{w_5\} & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \end{array}$$

The matrix A means in practice that there is a link or wire (w_1) coming from the port c and supplying the port a. There are also two links (w_2 and w_3) coming from c and supplying b. The product $A \times B$ given by

$$A \times B = \begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{cc} e & f \end{array} \\ \begin{pmatrix} \{w_1 w_5\} & \emptyset \\ \{w_2 w_5, w_3 w_5\} & \emptyset \end{pmatrix} \end{array}$$

shows in practice that there is also a link $(w_1 w_5)$ coming from e and supplying a. Typically, a port will be either an input or output port of a box.