*Adv Syst Sci Appl 2020; **01**; 50-65*
Published online at http://ijassa.ipu.ru/index.php/ijassa/article/view/781

Leveraging Category Theory in Model Based Enterprise

Serge P. Kovalyov[*]

V. A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences
Moscow, Russia
E-mail: kovalyov@sibnet.ru

Abstract: A mathematical framework based on category theory is proposed to formally describe and explore procedures of modeling engineering products and processes that comprise operation of a model-based enterprise. The framework is intended to provide interoperability across a variety of engineering modeling languages and tools, supplying them with a common abstract foundation capable to represent, generate, and verify diverse design and production knowledge. The framework is leveraged via algebraic representation of product configurations as diagrams in categories with models as objects and descriptions of actions involved into products assembly as morphisms. Relevance of the framework is justified by appealing to systems engineering standards such as IEC 81346. Category theoretical methods for solving direct assembly problems that consist in constructing a product model from a given configuration are presented. Specifically, solutions are obtained via the universal construction called a colimit of a diagram. Much attention is then paid to stating and solving inverse assembly problems that consist in recovery and subsequent optimization of the configuration from the product model and assembly actions. Inverse problem solving is in demand for generative design, viz. an emerging fully automatic product development and manufacturing technology. Example solutions to direct and inverse problems are described in categories that represent two major areas of model-based enterprise operation: solid body geometric modeling of mechanical products and discrete-event simulation of production processes.

Keywords: model-based enterprise, model-based systems engineering, assembly planning, generative design, category theory, colimit

1. INTRODUCTION

A model-based enterprise is a type of industrial organization in which all product life cycle processes are arranged around a digital 3D-model of the product annotated with detailed product manufacturing information (PMI) [1]. Techniques and tools to develop and leverage such a model, with applicability ranging from requirements elicitation and analysis through design to manufacturing, maintenance, and disposal, are provided by model-based systems engineering (MBSE). More than a decade has passed since the introduction of the model-based enterprise concept, during which it gradually matured to an industrial level of readiness for implementation, with supporting standards, neutral formats, techniques, and software tools for maintaining PMI. Recently, in the emerging Industry 4.0 context, new challenges arise, associated with upgrading a product model to the level of a digital twin, viz. a virtual copy of the product that reliably reproduces and imposes the structure, state, and behavior of the physical original in real time [2]. The degree of design and production automation is increased steadily, up to complete elimination of human intervention in the so-called generative design cycle [3].

At the same time, the core problem of specifying the product model in a neutral form that guarantees complete undistorted access to it for all life cycle participants and tools cannot be

[*] Corresponding author: kovalyov@sibnet.ru

considered solved [4]. There are many specific modeling languages, techniques, and tools that are poorly compatible with each other. A natural, albeit "difficult", approach to make them interoperable consists in supplying them with a common abstract formal framework of reference that is capable to represent, generate, and verify diverse design and production knowledge. The highest level of formality, generality, and verification power is achievable by leveraging appropriate mathematics. In this paper, following a recent trend [5, 6, 7], category theory is employed as such a framework. Models are treated as objects of suitable categories, in which morphisms describe actions involved into assembling complex products. Category theoretical constructions that represent MBSE techniques on abstract conceptual level are defined and studied. Considerable amount of experience in such a study has been gathered previously in model-based software engineering [8] and is now being transferred to systems engineering in general. Specifically, a product configuration is established to be represented as the diagram in a category of models, and the assembly procedure is formally described by a universal construction called the colimit of this diagram [9]. In the context of generative design, the inverse assembly problems are stated, solving of which amounts to reconstructing a diagram from a fragment of its colimit [10].

Some approaches to "categorify" the engineering domain are presented in the literature. For example, the framework for ontology and database design is proposed, where objects of the categories represent information entities that specify various aspects (facets) of products; and morphisms describe functional relationships like "has" or "contains" that connect entities to meaningful databases [11]. In [12], a symmetric monoidal category is constructed whose associated graphical language is capable to represent signal-flow diagrams of control theory: control system state spaces, comprised by finite vectors of signals, are objects of the category, and signal processing devices are represented by morphisms. Our approach differs from these and others in strong adhering to the model-based paradigm, where models and manipulations with them are taken for primary entities and thus represented by objects and morphisms, respectively.

The paper is composed as follows. Section 2 is devoted to overview of problems pertaining to assembling products in model-based setting. In Sections 3 and 4, category theoretical means to solve direct and inverse assembly problems are presented, respectively. Some conclusions and directions for further research are outlined in Section 5.

2. DIRECT AND INVERSE PROBLEMS OF ASSEMBLING PRODUCTS

In MBSE, products, parts, and equipment are represented by formalized models of various types: digital images of geometric shapes and bodies, numerical approximations of differential equations, labeled graphs, databases, etc. Herewith, the internal structure of models is not so much necessary to be known at decomposing the product into assembly units and planning the production process. The range of model capabilities to connect with other models is more important in order to create a hierarchy of assembly unit models. In other words, the models are usually considered as "black boxes" with known behavior with respect to other models, and are combined into structural schematics of complex products and processes, as shown in Fig. 1 [13]. Hence, model-based production planning procedures and software tools operate over a virtual catalog of models and descriptions of possible actions involved into composing assembly units.

This point of view is substantiated by systems engineering standards such as IEC 81346 "Industrial systems, installations and equipment and industrial products—Structuring principles and reference designations". This standard prescribes to represent the complex product structure as a directed graph of the hierarchy of units that are denoted by reference designations. The structure and related labeled graphs of various other types routinely occur while performing various MBSE procedures [10]. A graph that consists of models of constituents of a certain unit, which are related by the unit assembly actions, is called the unit configuration.

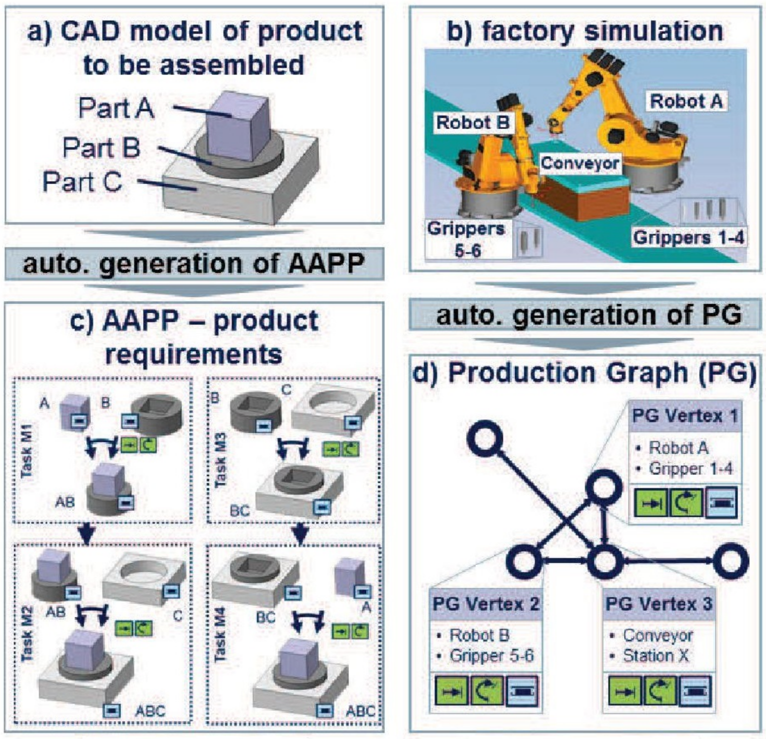

Fig. 1. Assembly planning [13]

As an example that is particularly relevant to mechanical design, consider a mechanical product consisting of rigid solid parts (see the left half of Fig. 1). For such a product, models of parts and assembly units are geometric bodies that can be represented for computer processing by various means: Constructive Solid Geometry (CSG), finite array of small spatial cells (voxels), boundary representation (BRep) [14]. Representations of particular product parts or assembly units are obtained from such models by affine isometries and stretches. For instance, the model of a barbell (heavy sporting equipment) consists of several cylinders of different sizes. In the barbell structure graph, the cylinders are represented by different vertices, although all of them are generated by the same geometric model. So the solid body model catalog contains a cylinder, with which several different actions of inclusion into the barbell are associated.

Another example of modeling technique, widely used in production planning, is discrete-event simulation (see the right half of Fig. 1). Its widespread support is regarded among the most significant achievements of MBSE. Here, a model represents an operational scenario, viz. a fragment of the imagined history of the simulated object behavior described as a discrete sequence of events of various types. Some events may trigger others or prevent others from occurring, inducing causal relations between some events within a scenario. Descriptions of actions used to build scenarios of the complex systems behavior reflect the contribution of the components. Specifically, the operational scenario of the model-based enterprise plant is composed from operational scenarios of plant equipment units that interact with each other in the course of the production process. So, in discrete-event simulation, a structure graph represents the process hierarchy.

The direct product assembly problem is stated as follows. Given a product configuration, the model of the product as a whole needs to be constructed along with actions involved into assembling it, to complete the hierarchical product structure graph. The principle of constructing the target product model is easy to see from the structure representation: the model should be located at precisely one hierarchical level above the level of largest constituent units. In other words, the product model should include all constituents' models respecting their relations within the configuration, and nothing more in a sense that it should be included into any model that include all constituents' models respecting the configuration relations.

This principle is easy to explain on a simple example. Suppose that a product should be assembled from two given parts *P* and *S*, and a manufacturing engineer has decided to join them by means of a certain glue, viz. the intermediate part *G* that can be directly applied on both *P* and *S*. Action of the glue is described by the following configuration: parts *G* and *P* comprise an intermediate unit P_G produced as a result of applying the glue on *P*, and similarly, parts *G* and *S* comprise a unit S_G. The model of the target product *R* assembled by gluing *P* and *S* with *G* is selected among candidate models that contain units P_G and S_G and respect the glue *G* in a sense that tracing the glue inclusion through either one of the two units amounts to the same action. The selection obeys the following structural criterion: *R* should be unambiguously identifiable within any candidate model *T*. This criterion is shown schematically in Fig. 2 [15], where models included into the IEC 81346 compliant product structure graph are colored in yellow and tagged by reference designations, and the glue is appended as a manufacturing information annotation.

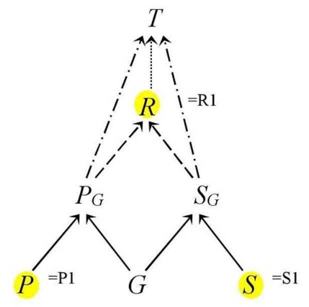

Fig. 2. Schematics of gluing [15]

Provided that such a model *R* exists, it clearly represents the product that is assembled by gluing two parts without any extra changes or augmentations. Moreover, it is easy to see that such model is unique in a sense that any two such models, by definition, contain each other. On the contrary, if the model *R* doesn't exist (cannot be found in the model catalog), then the wrong glue was chosen: *G* is incapable to connect *P* with *S*.

Notice that the gluing configuration is significantly simplified in case when influence of the glue on parts is negligible, so we can assume that $P_G = P$ and $S_G = S$. In this case, the configuration takes a shape of a span, viz. a graph consisting of two edges with the same source: $P \leftarrow G \rightarrow S$. We will refer to this kind of gluing as non-intrusive. In real world, it routinely occurs in solid body modeling: applying thin layer of a glue leaves the shape of a part unchanged. For example, non-intrusive gluing is intended to fix parts together at assembly tasks M1-M4 shown in Fig. 1.

Let's return to the general setting. If models contain values of some parameters, and descriptions of assembly actions contain rules for transforming the values, then the structure graph allows to calculate values of the parameters for the target product, including geometry, topology, dimensions, tolerances, etc. Emergent parameters that can't be attributed to any particular unit can also be calculated. Examples of calculations of this kind are known in the field of composite materials design [16]. The averaged (effective) physical properties of a composite, such as a Young's modulus and a Poisson's ratio, depend in a complex way on properties of components and methods of manufacturing the composite from them. Using elasticity theory methods, these dependencies are specified as linearized matrix relations that label arrows of the composite material structure graph. Then it becomes possible to predict the properties of the material virtually by computation over a database of components shown in Fig. 3, avoiding costly physical experiments.

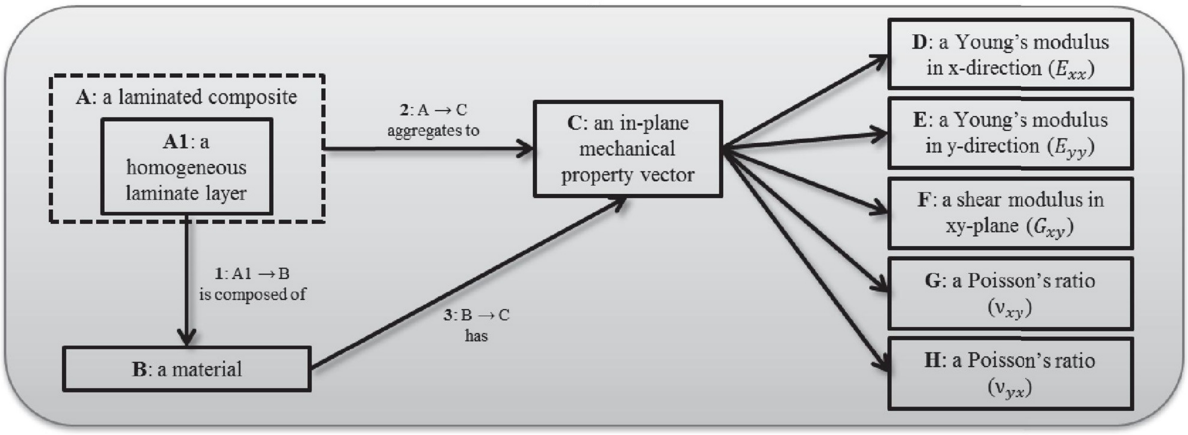
A: a laminated composite
A1: a homogeneous laminate layer
2: A → C aggregates to
C: an in-plane mechanical property vector
1: A1 → B is composed of
3: B → C has
B: a material
D: a Young's modulus in x-direction (E_{xx})
E: a Young's modulus in y-direction (E_{yy})
F: a shear modulus in xy-plane (G_{xy})
G: a Poisson's ratio (ν_{xy})
H: a Poisson's ratio (ν_{yx})

Fig. 3. Materials database [16]

Although direct calculation of the product model by its configuration is so much important, it plays only a secondary, instrumental role in preparing a large complex product for manufacturing. According to the standards like IEC 81346 and manufacturing practices, the assembly process is routinely designed in a top-down fashion, from the product as a whole to progressively smaller units. Therefore, a typical assembly planning problem has inverse rather than direct character: given the unit model, it is needed to determine (recover) the configuration from which the unit can be effectively assembled. The assortment of theoretically possible configurations is unimaginably huge, but in practice, it is severely constrained by plant capabilities, stakeholders' requirements, suppliers' offers, engineers' preferences, and other factors. So, the primary assembly planning procedure consists in identification of the class of all configurations that produce the given unit and satisfy all constraints imposed by practical factors. Once this class is identified, search for the final unit configuration is performed across it aiming to optimize conventional process objective functions, such as cost, duration, and complexity [17]. For this reason, we will call this class the assembly design space. Developing algorithms and tools to explore the space on a solid mathematical basis powered by computer algebra and artificial intelligence would allow to automate the design for manufacturability (DfM) practices. The ultimate target is the end-to-end generative design cycle of complex products in model-based enterprise environment.

For example, the inverse problem of gluing is stated as follows. In accordance with the IEC 81346 standard, the structure of the target product R is specified as a graph consisting of three models P, S, R and two actions $P \to R$, $S \to R$. An admissible gluing configuration is any combination of five models and four actions that comprise the "zigzag" shape highlighted by solid arrows in Fig. 2. Among all such configurations, those are of interest for which solving the direct problem produces specified actions $P \to R$ and $S \to R$ as two-step inclusions of the two outer models into the target product. In particular, outer models themselves are required to be precisely P and S, and the target product model is required to be R.

3. CATEGORY THEORY IN PRODUCT ASSEMBLY MODELING

As stated in the Introduction, the natural source of mathematical means for representation, generation, and verification of product structure graphs is category theory, viz. a branch of higher algebra that "starts with the observation that many properties of mathematical systems can be unified and simplified by a presentation with diagrams of arrows" ([18, p. 1]). Recall that a category is a class of abstract objects pairwise related by morphisms (arrows). The precise definition takes just a few lines: a category C consists of a class of objects $\mathsf{Ob}\ C$ and a class of morphisms $\mathsf{Mor}\ C$ endowed with the following operations. Firstly, two objects are associated with each morphism f: a domain (arrow source) $\mathsf{dom}\, f$ and a codomain (target) $\mathsf{codom}\, f$, so that the pair of equalities $\mathsf{dom}\, f = A$ and $\mathsf{codom}\, f = B$ is depicted as an arrow $f : A \to B$. Secondly, for any pair of morphisms f, g that satisfies the condition $\mathsf{codom}\, f = \mathsf{dom}\ g$, the composition of morphisms $g \circ f : \mathsf{dom}\ f \to \mathsf{codom}\ g$ is defined, and it is associative: for any three morphisms f, g, h such that $\mathsf{codom}\, f = \mathsf{dom}\ g$ and $\mathsf{codom}\ g = \mathsf{dom}\ h$, the equality $h \circ (g \circ f) = (h \circ g) \circ f$ holds. Thirdly, any object A has an associated identity morphism $1_A : A \to A$ such that $f \circ 1_A = 1_B \circ f = f$ for any morphism $f : A \to B$. A classic example of a category is **Set** that consists of all sets and all maps between sets: the composition law for maps is defined by the standard substitution, and the identity map of a set sends each element to itself.

The axioms of a category are inherently symmetrical with regard to swapping dom with codom along with reversing the order of composing morphisms: the category obtained this way from C is called dual of C and denoted by C^{op}. The axioms also tolerate restriction to subclasses, so the notion of subcategory is defined: a subcategory of C is a pair consisting of a subclass of $\mathsf{Ob}\ C$ and a subclass of $\mathsf{Mor}\ C$ that are closed with respect to operations inherited from C. A subcategory of C is called full if it contains any C-morphism whose domain and codomain are contained in it.

Together with category, the concept of functor is routinely introduced, which is a structure-preserving map between categories. Specifically, a functor $fun : C \to D$ from category C to D is a pair of maps $fun : \mathsf{Ob}\ C \to \mathsf{Ob}\ D$ (an object function) and $fun : \mathsf{Mor}\ C \to \mathsf{Mor}\ D$ (a morphism function) that satisfies the following conditions for any C-morphisms f, g and C-object A: (i) $fun(\mathsf{dom}\, f) = \mathsf{dom}\, fun(f)$, $fun(\mathsf{codom}\, f) = \mathsf{codom}\, fun(f)$; (ii) $fun(g \circ f) = fun(g) \circ fun(f)$ whenever the composition $g \circ f$ is defined; (iii) $fun(1_{\mathrm{A}}) = 1_{fun(A)}$. Obviously, all categories and all functors comprise a (formal) category denoted by **CAT**. To explore relationships between functors, the following construction is introduced: the natural transformation ε of a functor $fun : C \to D$ to a functor $fun' : C \to D$ is a function that assigns to each C-object A a D-morphism $\varepsilon_A : fun(A) \to fun'(A)$ in such a way that $\varepsilon_B \circ fun(f) = fun'(f) \circ \varepsilon_A$ for any C-morphism $f : A \to B$.

$$\begin{tikzcd}
fun(A) \arrow[r, "\varepsilon_A"] \arrow[d, "fun(f)"'] & fun'(A) \arrow[d, "fun'(f)"] \\
fun(B) \arrow[r, "\varepsilon_B"] & fun'(B)
\end{tikzcd}$$

Categories are routinely constructed from graphs: any directed multigraph induces a category with vertices as objects and finite paths as morphisms. Indeed, the domain and the codomain of a path are its source and target, respectively; composition of morphisms acts as concatenation of paths; the identity morphism of a vertex is an empty "path" from the vertex to itself. This construction leads to the fundamental notion of a diagram in C, viz. a functor of the kind $\Delta : X \to C$ where X is a category generated by some graph called the shape of the diagram. A diagram is visually depicted as a graph that generates X, endowed with labels from C: each vertex $\boldsymbol{a}$ is labeled by the C-object $\Delta(\boldsymbol{a})$ and each edge k is labeled by the C-morphism $\Delta(k)$. All diagrams in C comprise a category denoted by $\mathbf{D}C$ (a covariant modification of a "super-comma" category introduced in [18, Exercise V.2.5]), in which a morphism from a diagram $\Delta : X \to C$ to $\Xi : Y \to C$ is a pair $\langle \gamma, fd \rangle$ consisting of a functor $fd : X \to Y$ and a natural transformation $\gamma : \Delta \to \Xi \circ fd$; the composition law for morphisms is specified as follows: $\langle\gamma, fd\rangle \circ \langle\varphi, gd\rangle = \langle\gamma_{gd(-)} \circ \varphi, fd \circ gd\rangle$. Category theory provides a number of algebraic techniques for construction and analysis of diagrams.

The effectiveness of employing category theory as a mathematical framework for model-based enterprise stems from the observation that a catalog of models and actions introduced in Section 2 is nothing but a category. Indeed, the catalog has composite morphisms (sequential execution of actions) and identity morphisms (idle "doing nothing" action with any model). For example, in solid body modeling of mechanical products, a model is algebraically represented as a subset of $\mathbb{R}^3$ which is bounded, regular (i.e., it coincides with the closure of its interior wrt the standard topology), and semi-analytical (i.e., it admits a representation as a finite Boolean combination of sets of the kind $\{(x, y, z) \mid F_1(x, y, z) \leq 0, F_2(x, y, z) \leq 0, \ldots\}$ where $F_i : \mathbb{R}^3 \to \mathbb{R}$ is a real analytic function for any index i) [14]. In order to support specifications of assembly procedures such as gluing bodies together by surface fragments, all bounded regular semi-analytical subsets of $\mathbb{R}^n$, $0 \leq n \leq 3$, are considered as valid solid body models. Then, the quotient is formed: any two such sets that can be turned into one another by a composition of affine isometries and stretches are identified with one another. Morphisms of such equivalence classes, which describe assembly actions of complex mechanical products, are generated by isometric embeddings and stretches. This way, the subcategory of **Set** emerges, which we will denote by **SBM** (from Solid Body Modeling).

For many well-known MBSE techniques, formal description of the model catalog produces a category of sets endowed with some structure, e.g., algebraic systems, topological spaces, graphs, etc. Morphisms in such categories are maps of sets compatible with the structure. A canonical functor to **Set** acts on such a category by "forgetting" the structure. An example is the following algebraic representation of discrete-event simulation. A scenario is represented by a set of events partially ordered by causal dependencies and labeled by event types. Since neither events nor dependencies, nor labels could be "lost" when composing a complex system behavior scenario from components behavior scenarios, actions involved into assembling scenarios are described as maps that preserve the order and the labeling [19]. All scenarios and all such actions form a category called **Pomset** and equipped with a functor to **Set** that forgets the order and the labeling.

Let C be a category that represents some model catalog. A product structure and related graphs are diagrams in C, so the category $\mathbf{D}C$ represents the respective catalog of such graphs and actions involved into product structure development. We show how to rigorously solve various problems of assembling products by means of the category theoretical framework provided by $\mathbf{D}C$. For instance, the solution to the direct problem is represented by the following universal construction called the colimit of the diagram. Denote by $\mathbf{1}$ the singleton category that consists of one object $\boldsymbol{0}$ and one morphism $1_{\boldsymbol{0}}$. For any category X, there exists a unique functor from X to $\mathbf{1}$, denoted by $!_X : X \to \mathbf{1}$, which sends any X-object to $\boldsymbol{0}$ (in other words, $\mathbf{1}$ is a terminal **CAT**-object). There is a full embedding (i.e., an injective functor onto a full subcategory of its codomain) $\ulcorner - \urcorner : C \hookrightarrow \mathbf{D}C$ that takes a C-object Q to the singleton diagram $\ulcorner Q \urcorner : \mathbf{1} \to C : \boldsymbol{0} \mapsto Q$. A $\mathbf{D}C$-morphism with a singleton diagram as the codomain is called a cocone. This name is justified by visual representation: a cocone $\langle\sigma, !_X\rangle : \Delta \to \ulcorner Q \urcorner$ over a diagram $\Delta : X \to C$ can be depicted by adding an extra vertex Q and associated edges $\sigma_{\boldsymbol{a}} : \Delta(\boldsymbol{a}) \to Q$, $\boldsymbol{a} \in \mathsf{Ob}\, X$, to the graph of the base diagram Δ. If the extra edges were directed reversely (dually), i.e., from Q to vertices of Δ, then category theorists would call this picture by the familiar word "cone", hence they apply the prefix "co-" to indicate the duality.

A colimit (named as a dual to a limit) of Δ is a cocone $\mathsf{colim}\ \Delta : \Delta \to \ulcorner R \urcorner$ that is universal in the following sense: for any C-object T and any cocone $\delta : \Delta \to \ulcorner T \urcorner$ there exists a unique C-morphism $w : R \to T$ that satisfies the equality $\delta = \ulcorner w \urcorner \circ \mathsf{colim}\ \Delta$. It is easy to see that this universality condition is precisely the structural criterion from Section 2 which determines the model R of the target product built from the configuration Δ. Thus, given a product configuration represented as a diagram, rigorous analysis and verification of the product assembly process amounts to checking that the diagram has a colimit and calculating its vertex and associated edges.

For example, a vertex of a colimit of a discrete diagram is known as a sum of objects that label the diagram's vertices, and its edges are called injections. Another useful kind of a colimit is that of a diagram consisting of a parallel pair of morphisms $\bullet \rightrightarrows \bullet$. The colimit edge that starts from the right vertex of the diagram is called a coequalizer of the pair; it generalizes the set-theoretical concept of a quotient. A morphism is called a regular epimorphism whenever there exists a parallel pair that it coequalizes.

Yet another fundamental type of a colimit is constructed as follows. Let V be a span-shaped two-arrow graph $\bullet \leftarrow \bullet \rightarrow \bullet$ and $\Delta : V \rightarrow C$ be a V-shaped diagram populated by a pair of C-morphisms $f : P \leftarrow G \rightarrow S : g$. The colimit of Δ, called a pushout (or a universal square) of the pair, is specified by the vertex R and the pair of edges $p : P \rightarrow R \leftarrow S : s$ that satisfy the following two conditions: the equality $p \circ f = s \circ g$ holds (it establishes naturality of the pushout and determines the colimit edge directed from G to R), and, for any object T and a pair of morphisms $u : P \rightarrow T \leftarrow S : v$ such that $u \circ f = v \circ g$, there exists a unique morphism $w : R \rightarrow T$ that satisfies the equalities $w \circ p = u$ and $w \circ s = v$ (this is the universality condition).

$$\begin{tikzcd}
 & A \amalg B & \\
A \arrow[ur, dashed, "in_A"] & & B \arrow[ul, dashed, "in_B"']
\end{tikzcd}
\qquad
\begin{tikzcd}
M \arrow[r, shift left] \arrow[r, shift right] & N \arrow[r, dashed] & {}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & T & \\
 & R \arrow[u, dotted, "w"] & \\
P \arrow[uur, dashed, "u"] \arrow[ur, dashed, "p"] & & S \arrow[uul, dashed, "v"'] \arrow[ul, dashed, "s"'] \\
 & G \arrow[ul, "f"] \arrow[ur, "g"'] &
\end{tikzcd}$$

Clearly, a pushout provides a solution to the direct non-intrusive gluing problem stated in Section 2. A pushout can be constructed via a sum and a coequalizer provided that they exist [18, Exercise III.3.2]. Let $p = q \circ in_P$ and $s = q \circ in_S$, where $in_P : P \rightarrow P \amalg S \leftarrow S : in_S$ are injections of the components into the sum and q is the coequalizer of the following parallel pair of morphisms:

$$in_P \circ f,\ in_S \circ g : G \rightrightarrows P \amalg S.$$

For the pushout constructed in such a way, the naturality condition $(q \circ in_P) \circ f = (q \circ in_S) \circ g$ and the universality condition directly follow from the definition of a coequalizer. In the category **Set**, the sum is the disjoint union of sets P and S, and the coequalizer is its quotient map modulo the equivalence relation generated by the set of pairs $\{(f(x), g(x)) \mid x \in G\}$. So the edge map p takes each element of the set P to the element's equivalence class, and the map s acts similarly on S. If maps f and g are injective, then they establish that G is the intersection of the sets P and S, and the pushout produces precisely the union of the sets. It often occurs at modeling a mechanical product assembly by means of the category **SBM**. Illustrative examples are shown in Fig. 1 as assembly tasks M1-M4. For instance, the task M3 consists in actions depicted by dashed arrows of the commutative square shown in Fig. 4 (drawn upside-down in accordance with IEC 81346). It is easy to verify directly that this square is a pushout in **SBM**. Observe that the glue is represented by means of **SBM** as a parts' contact 2D surface on which thin layer of a red-colored chemical adhesive is applied to fix parts together.

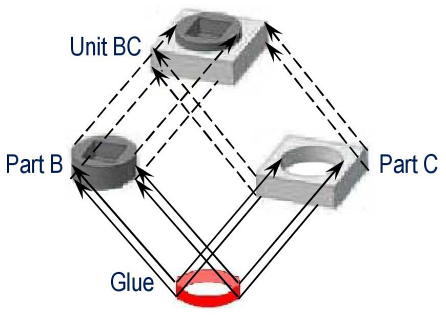

Fig. 4. Non-intrusive gluing as a pushout

Non-intrusive gluing is also widely used as a method of producing laminated composite materials. The database used in digital design and verification of the composite properties is outlined schematically in Fig. 3 above. An interested reader with some background in materials science may construct example pushouts and evaluate properties of composites over them following [16], as an exercise.

The construction of a pushout in **Set** admits generalization to many familiar categories of sets with structure. For example, any *V*-shaped diagram in the category of discrete-event simulation **Pomset** has a pushout. However, if it were constructed literally as in **Set**, then the order on the vertex inherited from base components might turn into a preorder, so an extra quotient may apply to restore the partial order on the vertex in a standard way. The real-world example that includes a pushout in **Pomset** is shown in Fig. 5 below.

A colimit of any diagram in any category can be constructed via a sequence of sums and pushouts provided that it exists (for example, the whole process shown in Fig. 1 amounts to assembling the product from parts A, B, and C via a sequence of pushouts in **SBM**). Furthermore, in practical applications simplifying the shape of a diagram helps to calculate colimits. We present the following result to pursue this direction.

Theorem 1:

Let $\Delta : X \to C$ be a diagram, Y be a full reflective subcategory of X, and $isc : Y \hookrightarrow X$ be an embedding. For each cocone $\theta : \Delta \circ isc \to \ulcorner S \urcorner$, there exists a unique cocone $\theta' : \Delta \to \ulcorner S \urcorner$ such that $\theta = \theta' \circ \langle 1_{\Delta \circ isc}, isc \rangle$. Moreover, the cocone θ is a colimit of the diagram $\Delta \circ isc$ if and only if the cocone θ' is a colimit of the diagram Δ.

Proof. Recall that a subcategory Y of X is called reflective if the embedding $isc : Y \hookrightarrow X$ has a left adjoint functor $isc^* : X \to Y$ called a reflector [18, § IV.3]. Specifically, there exists a natural transformation $\eta : 1_X \to isc \circ isc^*$, called the reflection unit, that consists of universal arrows. The unit induces a **D***C*-morphism $\langle \Delta(\eta), isc^* \rangle : \Delta \to \Delta \circ isc$. By definition of the cocone, for each X-object $\boldsymbol{i}$ the edge $f : \Delta(\boldsymbol{i}) \to S$ of the desired cocone θ' satisfies the condition $f = h \circ \Delta(\eta_{\boldsymbol{i}})$ where $h : \Delta(isc(isc^*(\boldsymbol{i}))) \to S$ is an edge of the cocone θ, so $\theta' = \theta \circ \langle \Delta(\eta), isc^* \rangle$.

Assume that θ is a colimit, and consider an arbitrary cocone $\rho : \Delta \to \ulcorner T \urcorner$. Let $w : S \to T$ be a canonical C-morphism that satisfies the condition $\rho \circ \langle 1_{\Delta \circ isc}, isc \rangle = \ulcorner w \urcorner \circ \theta$. Hence, $\ulcorner w \urcorner \circ \theta' = \rho \circ \langle 1_{\Delta \circ isc}, isc \rangle \circ \langle \Delta(\eta), isc^* \rangle = \rho$ and, in addition, if some C-morphism $\tilde{w} : S \to T$ satisfies the condition $\rho = \ulcorner \tilde{w} \urcorner \circ \theta'$, then $\rho \circ \langle 1_{\Delta \circ isc}, isc \rangle = \ulcorner \tilde{w} \urcorner \circ \theta' \circ \langle 1_{\Delta \circ isc}, isc \rangle = \ulcorner \tilde{w} \urcorner \circ \theta$, so $\tilde{w} = w$. We conclude that θ' is a colimit of the diagram Δ.

Conversely, assume that θ' is a colimit, and consider an arbitrary cocone $\gamma : \Delta \circ isc \to \ulcorner U \urcorner$. Let $u : S \to U$ be a canonical C-morphism that satisfies the condition $\gamma' =$

$\ulcorner u \urcorner \circ \theta'$ where $\gamma' = \gamma \circ \langle \Delta(\eta), isc^* \rangle$. Hence, $\ulcorner u \urcorner \circ \theta = \gamma \circ \langle \Delta(\eta), isc^* \rangle \circ \langle 1_{\Delta \circ isc}, isc \rangle = \gamma$ and, in addition, if some C-morphism $\tilde{u} : S \to U$ satisfies the condition $\gamma = \ulcorner \tilde{u} \urcorner \circ \theta$, then $\gamma' = \ulcorner \tilde{u} \urcorner \circ \theta \circ \langle \Delta(\eta), isc^* \rangle = \ulcorner \tilde{u} \urcorner \circ \theta'$, so $\tilde{u} = u$. We conclude that θ is a colimit of the diagram $\Delta \circ isc$. □

As an application of Theorem 1, consider the direct general gluing problem. Its solution amounts to constructing a colimit of a diagram with the following shape that we will denote by ***M***.

$$\begin{tikzcd}
 & \boldsymbol{b} & & \boldsymbol{b'} & \\
\boldsymbol{a} \arrow[ur] & & \boldsymbol{c} \arrow[ul] \arrow[ur] & & \boldsymbol{a'} \arrow[ul]
\end{tikzcd}$$

There exists an embedding $iv : \boldsymbol{V} \hookrightarrow \boldsymbol{M}$ that maps $\boldsymbol{V}$ onto a full subcategory of $\boldsymbol{M}$ with a set of object $\{\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{b'}\}$. This subcategory is reflective, with the reflector iv^* taking $\boldsymbol{a}$ to $\boldsymbol{b}$, $\boldsymbol{a'}$ to $\boldsymbol{b'}$, and other vertices to themselves, so that two outer arrows $\boldsymbol{a} \to \boldsymbol{b}$ and $\boldsymbol{a'} \to \boldsymbol{b'}$ are the reflection unit components associated with objects $\boldsymbol{a}$ and $\boldsymbol{a'}$, respectively. By Theorem 1, the colimit of a diagram $\Delta : \boldsymbol{M} \to C$ is determined one-to-one by the pushout of the diagram $\Delta \circ iv : \boldsymbol{V} \to C$ comprised from two central arrows of $\boldsymbol{M}$. Real-world examples of colimits of this kind can be constructed in the category **SBM** by representing "intrusive" means of joining mechanical parts, such as riveting. Yet another example is inspired by the right half of Fig. 1: the production process simulation can be glued from simulations of robot A, robot B, and the conveyor. According to the production graph, the conveyor acts as a "process glue" between two robots, delivering parts and assembly task results from one to another. One possible production scenario consisting of tasks M3 and M4 performed by robots B and A, respectively, is represented in Fig. 5 by a colimit of an ***M***-shaped diagram in the category **Pomset**.

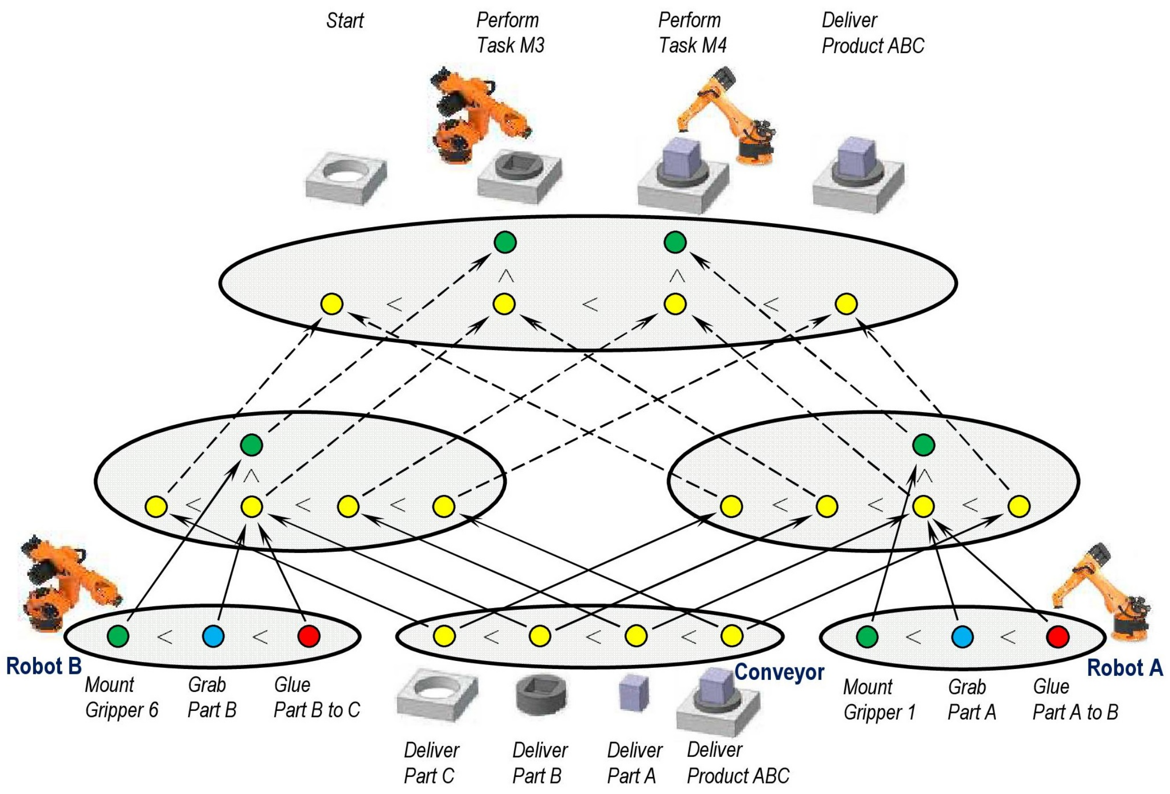

Fig. 5. Constructing a production process simulation via a colimit

We conclude this Section with the following remarks. A colimit of an arbitrary diagram Δ is unique up to an isomorphism provided that it exists: a cocone of the kind $\ulcorner i \urcorner \circ \mathsf{colim}\, \Delta$ is a colimit of Δ if and only if i is a C-isomorphism. Moreover, colimits allow to represent assembling products from configurations as a functor. Still further, the construction of the category of diagrams $\mathbf{D}C$ induces a monad in **CAT** whose associated category of algebras contains instances of the colimit functor and other functors that are relevant for mathematical framework of MBSE. Therefore, this monad is regarded as the MBSE metamodel [10].

4. INVERSE ASSEMBLY PROBLEMS AND SOLUTIONS

To explore inverse product assembly problems, the following technical concept of a subcocone is needed. A cocone δ is called a subcocone of a cocone β if there exists a $\mathbf{D}C$-morphism $\langle \mu, iy \rangle : \mathsf{dom}\ \delta \to \mathsf{dom}\ \beta$ such that the natural transformation μ consists of identity C-morphisms, the functor iy is injective, and $\beta \circ \langle \mu, iy \rangle = \delta$. The inverse problem is stated mathematically as follows. Given a cocone δ (that represents the known fragment of the target product structure) and a class Cd of diagrams (that represents the space of all practically feasible configurations), evaluate the subclass $Cds \subseteq Cd$ consisting of all diagrams from Cd with a colimit that contains δ as a subcocone.

The subclass Cds is precisely the assembly design space across which (sub-, Pareto-) optimal configurations are sought. Category theory may improve the search as follows. Let Q be a linearly ordered set in which some search objective function takes values (most often, Q is the set of all real or integer numbers). Obviously, Q can be regarded as a category, since any ordered set can be depicted as a directed graph. Of particular interest is the situation where the objective function acts as an object function of a functor that takes values in the category Q and arguments in a subcategory of $\mathbf{D}C$ (or of the dual category $(\mathbf{D}C)^{\mathrm{op}}$) which has Cds as class of objects and enough non-identity morphisms. In this case, optimization algorithms of gradient descent type can be employed, which navigate along morphisms of this subcategory, calculating the path by means of computer algebra. Performant computer algebra packages for such category theory calculations are available [20].

For example, consider the inverse gluing problem stated in Section 2. Here, the class Cd from the general configuration recovery problem statement consists of all $\boldsymbol{M}$-shaped diagrams with a colimit in C. The cocone δ can easily be reconstructed from the product structure presented in Fig. 2 above: the cocone base is the diagram $\Delta : \boldsymbol{a} \mapsto P, \boldsymbol{a}' \mapsto S$ with the two-vertex discrete shape $\{\boldsymbol{a}, \boldsymbol{a}'\} \subseteq \boldsymbol{M}$, and the cocone vertex is the target product R, so the cocone is specified as a pair of C-morphisms $p : P \to R \leftarrow S : s$. This problem is solved as follows.

Proposition 1:

For any C-objects P, S, R and any pair of C-morphisms $p : P \to R \leftarrow S : s$, there exists a diagram $\Delta : \boldsymbol{M} \to C$ with a colimit $\langle \kappa, !_{\boldsymbol{M}} \rangle : \Delta \to \ulcorner R \urcorner$ that satisfies the equalities $\kappa_{\boldsymbol{a}} = p$ and $\kappa_{\boldsymbol{a}'} = s$. This diagram is unique up to a $\mathbf{D}C$*-isomorphism if and only if the pair consists of isomorphisms and every epimorphism with the codomain R is an isomorphism.*

Proof. Construct a desired diagram Δ in two steps. First, choose an arbitrary factorization of the kind $p = o \circ q$ in order to label the outer arrow $\boldsymbol{a} \to \boldsymbol{b}$ of Δ by the morphism q, along with the similar factorization $s = o' \circ q'$. Second, having that, as established in Section 3 by Theorem 1, a colimit of an $\boldsymbol{M}$-shaped diagram amounts to a pushout, find a diagram $\Xi : \boldsymbol{V} \to C$ that has a pushout specified by edges o and o', and use it to label two central arrows of Δ (i.e., to satisfy the equality $\Delta \circ iv = \Xi$).

Consider trivial factorizations with $o = o' = 1_R$. A C-morphism e with the codomain R is an epimorphism if and only if the equality $1_R \circ e = 1_R \circ e$ determines a pushout [18, Exercise III.4.4]. Hence, we shall take the following $\boldsymbol{M}$-shaped diagram, which we will denote by Δ_e, as Δ provided that e is an epimorphism.

$$\begin{tikzcd}
 & & & R & & \\
 & & R \arrow[ur, dashed, "1_R"] & & R \arrow[ul, dashed, "1_R"'] & \\
\Delta & P \arrow[ur, "p"] & & \mathsf{dom}\, e \arrow[ul, "e"'] \arrow[ur, "e"] & & S \arrow[ul, "s"']
\end{tikzcd}$$

In particular, with $e = 1_R$ we obtain somewhat paradoxical formal solution to the glue finding problem: if the complex product R that includes parts P and S is already specified, then let it serve as a glue fixing these parts together!

If p is not an isomorphism, then another solution (non-isomorphic to the above) is obtained by changing the factorization of p to another trivial one with $q = 1_P$, so that the part P plays the role of a glue as follows.

$$\begin{tikzcd}
 & & & R & & \\
 & & P \arrow[ur, dashed, "p"] & & R \arrow[ul, dashed, "1_R"'] & \\
\Delta & P \arrow[ur, "1_P"] & & P \arrow[ul, "1_P"'] \arrow[ur, "p"] & & S \arrow[ul, "s"']
\end{tikzcd}$$

Other pairs of morphisms that have a pushout specified by edges p and 1_R lead to other solutions. “Symmetrical” solutions are obtained from factorizations with $o = 1_R$ and $q' = 1_S$ provided that s is not an isomorphism. Other factorizations may produce yet other solutions.

If both p and s are isomorphisms, then morphisms o and o' are epimorphisms with the codomain R [21, Proposition 7.42]. If any such epimorphism is an iso, then the target diagram Δ is essentially unique: it can be comprised solely from isomorphisms. Otherwise, at least two non-isomorphic diagrams of the kind Δ_e exist. □

For example, in the category **Set** the uniqueness criterion of the glue finding problem solution is equivalent to the condition that the set R is empty. If this is not the case, then any larger set can be mapped onto R, so the target class Cds contains at least infinitely many pairwise non-isomorphic $\boldsymbol{M}$-shaped diagrams of the kind Δ_e.

The stepwise glue finding procedure executed to prove Proposition 1 admits generalization for recovering configurations with more complicated shapes than $\boldsymbol{M}$. Such a general procedure essentially amounts to “unwinding” the construction of a colimit from sums and pushouts. Crucial steps of the procedure consist in recovering a pushout from its edges. This is the case for an inverse non-intrusive gluing problem. Indeed, the class Cd in this problem's statement consists of all $\boldsymbol{V}$-shaped diagrams with a colimit in C, while the cocone δ has a discrete base consisting of two outer vertices of $\boldsymbol{V}$ and is specified by a pair of C-morphisms with the same codomain. This pushout recovery problem is quite difficult to solve. We start with solving it in the category **Set** as follows.

Proposition 2:

Let P, S, R be sets and $p : P \to R \leftarrow S : s$ *be a pair of maps.*

(1) A pair of maps that has a pushout specified by edges p and s exists if and only if the following condition holds for each $r \in R$*:*

$$(|p^{-1}(r)| = 0 \Rightarrow |s^{-1}(r)| = 1) \wedge (|s^{-1}(r)| = 0 \Rightarrow |p^{-1}(r)| = 1).$$

(2) Let $pr_P : P \leftarrow P \times S \to S : pr_S$ *be a cartesian product of sets equipped with projections and* $[p, s] : P \coprod S \to R$ *be the canonical map that acts as p on P and as s on S. A pair of maps* $f : P \leftarrow G \to S : g$ *has a pushout specified by edges p and s if and only if the map* $[p, s]$ *is surjective and there exists a map* $h : G \to P \times S$ *such that* $f = pr_P \circ h$, $g = pr_S \circ h$*, and the set* $h(G)$*, considered as a binary relation on* $P \coprod S$*, generates the equivalence relation that coincides with the kernel equivalence relation of the map* $[p, s]$*.*

(3) There exists a unique pair of maps that has a pushout specified by edges p and s if and only if the map $[p, s]$ *is bijective.*

Proof.

(1) Assume that the maps p and s specify the pushout of some pair of maps $f : P \leftarrow G \to S : g$. From the construction of pushout in **Set** described explicitly in Section 3, it is easy to see that if the set $p^{-1}(r) \subseteq P$ is empty for some $r \in R$, then the element r has been brought into R from the set $S \setminus g(G)$: there exists a unique $y \in S$ such that $s(y) = r$. The case for empty set $s^{-1}(r) \subseteq S$ is considered similarly.

Conversely, assume that p and s satisfy the condition (1); in particular, $p(P) \cup s(S) = R$. Construct a pullback (a limit, viz. a universal construction dual to a pushout) of the $\mathbf{V}^{\mathrm{op}}$-shaped diagram in **Set** populated by the pair [18, § III.4]: let

$$G_0 = \{(x, y) \mid p(x) = s(y)\} \subseteq P \times S,$$

$$f_0 : G_0 \to P : (x, y) \mapsto x,\ g_0 : G_0 \to S : (x, y) \mapsto y.$$

The equality $p \circ f_0 = s \circ g_0$ holds and determines the pullback; we claim that it also determines a pushout of the pair of pullback edges $f_0 : P \leftarrow G_0 \to S : g_0$. Indeed, construct the pushout of this pair in a standard way, with $p' : P \to R' \leftarrow S : s'$ being its edges. Define the map $i : R \to R'$ as follows: for each $r \in R$, if $r \in p(P)$, then let $i(r) = p'(x)$ for an arbitrary $x \in p^{-1}(r)$ (obviously, the value $i(r)$ doesn't depend upon the choice of x); otherwise, let $i(r) = s'(y)$ where y is a unique element of the set S such that $s(y) = r$ (such an element exists by assumption). It is easy to see that i is a bijection, $p = i^{-1} \circ p'$, and $s = i^{-1} \circ s'$, so the maps p and s are edges of a pushout of the pair consisting of f_0 and g_0.

(2) For a map $h : G \to P \times S$, the requirement that the map $[p, s]$ is surjective and its kernel equivalence relation is generated by $h(G)$ is equivalent to the condition that $[p, s]$ is a coequalizer of the following parallel pair of maps:

$$in_P \circ pr_P \circ h,\ in_S \circ pr_S \circ h : G \rightrightarrows P \coprod S.$$

If h satisfies this condition, then, by the construction of a pushout via a coequalizer, the maps p and s specify the pushout of the pair

$$pr_P \circ h : P \leftarrow G \to S : pr_S \circ h.$$

Conversely, assume that p and s specify the pushout of some pair of maps $f : P \leftarrow G \to S : g$. Let

$$h = \langle f, g \rangle : G \to P \times S : x \mapsto (f(x), g(x)).$$

(3) If the map $[p, s]$ is bijective, than its kernel equivalence relation is discrete (coincides with the identity). Hence, a pair of maps, constructed from a map $h : G \to P \times S$ composed with projections as in the proof of (2) above, may have a pushout with such edges p and s only if the set $h(G)$ is empty, which implies that G itself is empty. Indeed, the pair of empty maps $P \leftarrow \varnothing \to S$ is the only one that has a pushout with such edges.

Conversely, assume that the map $[p, s]$ is not bijective. If the condition (1) doesn't hold, then no pushout exists with edges p and s. Otherwise, the set G_0 constructed above is non-empty, so any larger set can be mapped onto it. In this case, infinitely many pairwise non-isomorphic *V*-shaped diagrams exist that have a pushout specified by edges p and s: by (2), each of these diagrams is populated with a pair of maps of the kind

$$f_0 \circ e : P \leftarrow G \to S : g_0 \circ e,$$

where $e : G \to G_0$ is an epimorphism (surjection). □

If the maps p and s are injective, then the maps f_0 and g_0 constructed via the pullback in the proof of Proposition 2 are injective as well [21, Proposition 11.18]. Hence, the method of the proof in many cases allows to recover gluing configurations in the category of solid body modeling **SBM**. For instance, recovering the glue for the assembly task M3 from Fig. 1 by the pullback yields a pushout isomorphic to the one shown in Fig. 4.

Proposition 2 can also be adapted for many familiar categories of sets with structure. In such a category, the solution existence condition stated for a pair $p : P \to R \leftarrow S : s$ in Proposition 2(1) is augmented with a suitable structure preservation requirement. One possible form of such a requirement is the following regularity condition: the canonical morphism $[p, s]$ is a regular epimorphism. At least, by the construction of a pushout via a coequalizer, the regularity condition is necessary for morphisms p and s to be edges of some pushout in any category where the sum $P \coprod S$ exists. The glue can be constructed, as in **Set**, via the pullback of the pair: the statement dual of [21, Lemma 11.15(1)] implies that in any category, if a pair of morphisms with the same codomain admits both pushout recovery and a pullback, then the latter is the former, i.e., the pullback is at the same time a pushout, so it provides a solution to the pushout recovery problem. Other glues can be constructed similarly to Proposition 2(2) that, restated in terms of coequalizer as in its proof, holds in any category where both the product $P \times S$ and the sum $P \coprod S$ exist. Analogues of Proposition 2(3) can be obtained as well.

The above considerations apply for the category of discrete-event simulation **Pomset**. For a pair of maps $p : P \to R \leftarrow S : s$ that preserve the order and the labeling, the pushout recovery problem has a solution if and only if the pair satisfies both the condition of Proposition 2(1) and the regularity condition. The example glue is the set G_0 constructed as in the proof of Proposition 2 and endowed with the order and the labeling inherited from the product $P \times S$. In enterprise simulation, such a glue represents a plant control system that ensures coherent operation of manufacturing equipment units P and S jointly executing the target process R. The construction of the glue demonstrates how category theory can be leveraged to automate control system programming. This kind of automation is in demand for the emerging approach to production planning and operation on the basis of a digital twin [22]. For example, recovering a glue of two upper maps from Fig. 5 by the pullback yields a control program specification isomorphic to the conveyor simulation shown at the bottom.

5. CONCLUSION

Category theory demonstrates significant potential in improving model-based enterprise performance. In particular, the theory provides mathematically rigorous solutions to direct and inverse problems associated with assembling products. A number of real-world examples of such problems can be found in the cited literature and elsewhere. Yet, theorems and proofs are of little utility to engineers: in order to hand category theoretical techniques over to industry, they shall be implemented in appropriate software tools. The tools, given requirements to the target product (e.g., a 3D-model) and to the manufacturing plant (e.g., a factory simulation), would represent and evaluate them by means of category theory as described above, and then automatically synthesize the production plan that is optimal wrt specified objective functors. Provided that such synergy between computer algebra and optimization algorithms will be achieved, the tools will open new horizons in automatic production planning and generative design. Development and deployment of the tools is a major area for further theoretical and applied research.

ACKNOWLEDGEMENTS

This work was supported by the Russian Foundation for Basic Research (Grant 19-011-00799).

REFERENCES

1. Frechette S. (2011). Model Based Enterprise for manufacturing. NIST, [Online]. Available: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=908343
2. Madni A. M., Madni C. C. & Lucero S. D. (2019). Leveraging digital twin technology in model-based systems engineering, *Systems*, 7(1), 7, https://doi.org/10.3390/systems7010007
3. Kowalski J. (2016). *CAD is a lie: generative design to the rescue*. Autodesk, [Online]. Available: https://www.autodesk.com/redshift/generative-design/
4. Harris G. A., Abernathy D., Whittenburg R., Holden A. & Still A. (2018). Issues in implementing a Model Based Enterprise. *Proc. 9th Model-Based Enterprise Summit (MBE 2018)*. NIST, 34–38. https://doi.org/10.6028/NIST.AMS.100-22
5. Mabrok M. A. & Ryan M. J. (2017). Category theory as a formal mathematical foundation for model-based systems engineering, *Applied Math. and Information Sci.*, 11(1), 43–51. https://doi.org/10.18576/amis/110106
6. Luzeaux D. (2015). A formal foundation of systems engineering. In Boulanger F., Krob D., Morel G. & Roussel J. C. (Eds.), *Complex Systems Design & Management* (pp. 133–148). Springer, Cham. https://doi.org/10.1007/978-3-319-11617-4_10
7. Wisnesky R., Breiner S., Jones A., Spivak D. I. & Subrahmanian E. (2017). Using category theory to facilitate multiple manufacturing service database integration, *J. Computing and Information Sci. in Engineering*, 17(2), 021011. https://doi.org/10.1115/1.4034268
8. Kovalyov S. P. (2016). Category-theoretic approach to software systems design, *J. Math. Sci.*, 214(6), 814–853. https://doi.org/10.1007/s10958-016-2814-1
9. Ginali S. & Goguen J. (1978). A categorical approach to general systems. In Klir G. J. (Ed.), *Conf. (Intl.) on Applied General Systems Research Proc.* (pp. 257–270). NATO conference ser., 5, Plenum Press. https://doi.org/10.1007/978-1-4757-0555-3_18
10. Kovalyov S. P. (2018). Category theory as a mathematical pragmatics of model-based systems engineering, *Inform. Appl.*, 12(1), 95–104. https://doi.org/10.14357/19922264180112
11. Spivak D. & Kent R. (2012). Ologs: a categorical framework for knowledge representation, *PloS one*, 7, e24274. https://doi.org/10.1371/journal.pone.0024274
12. Baez J. C. & Erbele J. (2015). Categories in control, *Theory and Applications of Categories*, 30(24), 836–881. http://www.tac.mta.ca/tac/volumes/30/24/30-24.pdf

13. Michniewicza J., Reinhart G. & Boschert S. (2016). CAD-based automated assembly planning for variable products in modular production systems, *Procedia CIRP*, 44, 44–49. https://doi.org/10.1016/j.procir.2016.02.016
14. Requicha A. G. (1980). Representations for rigid solids: theory, methods, and systems, *J. ACM Computing Surveys*, 12(4), 437–464. https://doi.org/10.1145/356827.356833
15. Kovalyov S. P. (2017). Methods of category theory in model-based systems engineering, *Inform. Appl.*, 11(3), 42–50. https://doi.org/10.14357/19922264170305
16. Giesa T., Spivak D. I. & Buehler M. J. (2012). Category theory based solution for the building block replacement problem in materials design, *Advanced Engineering Materials*, 14(9), 810–817. https://dspace.mit.edu/handle/1721.1/77560
17. Bozhko A. N. & Karpenko A. P. (2018). Computer-aided subassembly generation. In *Proc. 5th Intl. Workshop IWCI 2018* (pp. 7–12). Advances in Intelligent Systems Research ser., 158, Atlantis Press. https://doi.org/10.2991/iwci-18.2018.2
18. Mac Lane S. (1998). *Categories for the working mathematician*. 2nd Ed. New York, NY: Springer.
19. Pratt V. R. (1986). Modeling concurrency with partial orders, *Int. J. Parallel Prog.*, 15(1), 33–71. https://doi.org/10.1007/BF01379149
20. Gross J., Chlipala A. & Spivak D. I. (2014). Experience implementing a performant category-theory library in Coq. In Klein G. & Gamboa R. (Eds.), *5th Conf. (Intl.) on Interactive Theorem Proving Proceedings* (pp. 275–291). Lecture notes in computer science ser., 8558, Springer. https://doi.org/10.1007/978-3-319-08970-6
21. Adámek J., Herrlich H. & Strecker G. E. (1990). *Abstract and concrete categories*. New York, NY: John Wiley.
22. Liu Q., Zhang H., Leng J. & Chen X. (2019). Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system, *Intl. J. Production Research*, 57:12, 3903–3919. https://doi.org/10.1080/00207543.2018.1471243