A Visual Interactive Environment for Engineering Knowledge Modelling

Ewa Grabska, Barbara Strug, and Grażyna Ślusarczyk(✉)

Jagiellonian University, Kraków, Poland
{ewa.grabska,grazyna.slusarczyk}@uj.edu.pl

Abstract. Today, the field of modelling of engineering knowledge needs facilitating the specification and programming tasks associated with the modelling of complex systems. In this paper a graphical formalism is suggested as a way to tackle the system modelling problems more quickly. Composition graph (CP-graph) rewriting is suggested as a basis for visual modelling tools. Nodes of a CP-graph have explicit connection elements called bonds to which edges are attached. The proposed Visual MOdelling with GRaphs (VIZMOGR) system offers a simple graphical interface for conceptual designing artifacts, in which outline solutions are determined by spatial and structural relationship of the principal components and major functions. The system allows the user to create structure components of the model in the form of CP-graph nodes labelled by icons representing artifact's components. Creating edges between bonds of nodes is controlled by means of label-dependent rules. To modify created CP-graphs VIZMOGR system provides schemes of rules, which are most often applied to develop graphs. Moreover VIZMOGR system supports attributed CP-graphs by enabling the user to propagate semantic information and to capture parametric modeling knowledge. The benefits of the proposed approach and usefulness of VIZMOGR system are shown using examples of designing bridges.

Keywords: Knowledge modelling · Composition graph · Graph rewriting

1 Introduction

Today, the modelling of knowledge is made very complex by the use of multiple discipline-specific models. Integration of these models is difficult because different scientific bases are applied [1]. We are at a point in history when there exists an obvious need for facilitating the specification and programming tasks associated with the modelling of complex systems.

We consider here the process of modelling corresponding to early stages of design, called conceptual design, where primary decisions are to be made. The product of these stages is meant as an outline solution to a design problem in which spatial and structural relationships of the principal components are fixed, and major functions are determined. The outline solution can be easily represented by a graph [2].

In this paper a graph formalism is suggested as a way to tackle the system modelling problems more quickly. The consequence of using this formalism is stimulation of visual

I. F. C. Smith and B. Domer (Eds.): EG-ICE 2018, LNCS 10863, pp. 219–230, 2018.
https://doi.org/10.1007/978-3-319-91635-4_12

thinking that means taking advantage of our innate ability to see, discovering ideas that are otherwise invisible, developing these ideas quickly and intuitively, and then sharing them with others. *Composition graph (CP-graph)* rewriting is proposed as a basis for visual modelling tools. Intuitively, a CP-graph is a graph where nodes have explicit connection elements called *bonds* [3]. Edges are attached to bonds which express the properties of the connections (see Fig. 1). Graph rewriting consists of replacing a subgraph in the graph being processed by another graph, thus giving the possibility of creating a new graph out of the original graph algorithmically. We focus on CP-graph rewriting systems that have been used to synthesize models in a wide variety of domains such as architectural design, computer games, computational grids and Finite Element Method [4, 5].

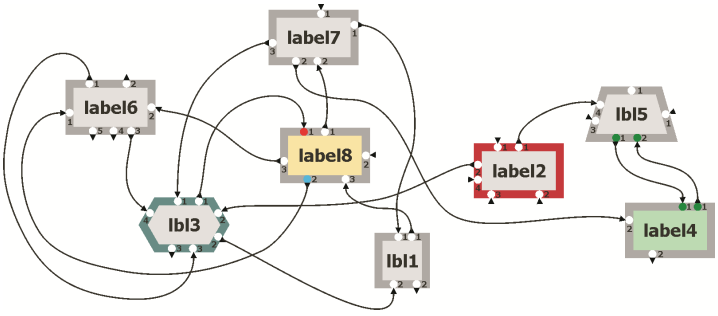

Fig. 1. An example of a CP-graph structure.

There exist a number of tools for generating graphs such as PROGRESS, AGG, Fujaba, GROOVE, but they are usually specialized for experts of a particular subject and focus on different areas. An interesting tool for graph rewriting is PORGY, which uses port graphs [6]. Ports, like bonds, are connection elements between edges and nodes defined on the syntactic level. However, the difference between ports and bonds is fundamental, since bonds are also interpreted on the semantic level. This interpretation plays an essential role in building a layer of meaning during the initial phase of modelling, called conceptualization.

The existing systems can be considered in the framework of Computational Design Synthesis (CDS). CDS aims to support conceptual design through formalization and computer aided process of finding solutions for knowledge-based design tasks. The lack of popularity of CDS in industry is evident. This lack of acceptance is caused by a great effort required to acquire knowledge and skills necessary to formalize tasks, limited range of applications, lack of reuse of existing design knowledge, lack of modeling standards and tool integration [7]. For instance, CDC is often discussed within the framework of local graph transformations, specified by graph rewriting rules for transforming subgraphs of limited sizes. The graph approach is fundamental because it allows to consider both the product modelling and the process modelling. There exist many formal graph approaches to the application of graph rewriting but their use most often requires skills to formalize tasks. The main purpose of this paper is to facilitate the specification of graph rewriting rules by presenting steps of applying them on the visual level.

The proposed VIZMOGR (Visual MOdelling with GRaphs) system is a visual environment that allows users to define CP-graphs and rewriting rules, and to apply these rules in an interactive way or via the use of a strategy with memory, where successive rewriting steps depend on the history of the previous ones [8]. VIZMOGR system offers a simple graphical interface that allows users to visualize and analyse the dynamics of finding solutions for knowledge-based design tasks. The graph rewriting in the system is based on the GraphTool engine [4, 5]. In the first step of modelling, when the user analyses relevant entities and organizes them into concepts and relations, he can decide on shapes and colours of elements creating a graph structure of the model. It turns out that visual perception of graph structures in graphic design is based on the same neural machinery that is used to interpret the everyday environment [9]. A kind of natural semantics, which is fundamental in reasoning, is built on the basis of user's patterns.

VIZMOGR system offers the support for attributed CP-graphs. Within each project there is a file that is used for creating declaration of attributes that can be later assigned to any object defined in the project. Each rewriting rule can be equipped with a predicate of applicability specifying conditions under which the rule can be used. These predicates are in the form of expressions with parameters being attributes assigned to CP-graph nodes and edges and therefore describe semantic properties of CP-graphs to which rules can be applied.

The benefits of the proposed approach and usefulness of VIZMOGR system are shown using examples of designing bridges.

2 CP-Graph Representation of Designs

In the proposed VIZMOGR system design objects are internally represented by means of CP-graphs. A CP-graph is a labelled and attributed graph, where nodes represent components of artefacts and are labelled by names of components they represent. Moreover, attributes specifying properties of components are assigned to nodes representing them. To each node a number of bonds expressing potential connections between components is assigned. Edges of a CP-graph represent relations between components. Non-symmetrical relations are represented in a directed CP-graph, where nodes are equipped with two types of bonds: in-bonds and out-bonds and directed edges are drawn from out-bonds to in-bonds. Figure 1 shows an example of a generic CP-graph structure with directed edges, which represents a designed model. In this CP-graph, for instance the node "*label7*" and the node "*lbl3*" are connected by the directed edge which is drawn from the out-bond of the former node to the in-bond of the latter one.

Bonds of CP-graph nodes, which are not engaged are called free and represent places on components where other components can be attached. In Fig. 1 the node *"label7"* has one free in-bond, whereas the node "*lbl3*" has one free out-bond. Each free bond of a node signals a potential connection of a node with other nodes [3]. Undirected CP-graphs have only one type of bonds. In Fig. 2b a CP-graph representing the structure of the bridge shown in Fig. 2a is undirected, i.e., its edges represent symmetrical relations. This structure is a solution internal representation obtained as a result of the modeling process controlled by the user and described in the next section. Nodes of this CP-graph

represent abutments, pylons, beams and two types of cables, which constitute structural components of the bridge, and are labeled as *ab*, *bm*, *pl*, *cb1* and *cb2*, respectively. Bonds assigned to nodes are connected by undirected edges representing relations between bridge components. The edge labels *fx* and *hn* denote fixed and hinged connections between components of the bridge. The edge label *jn* denotes the join relation.

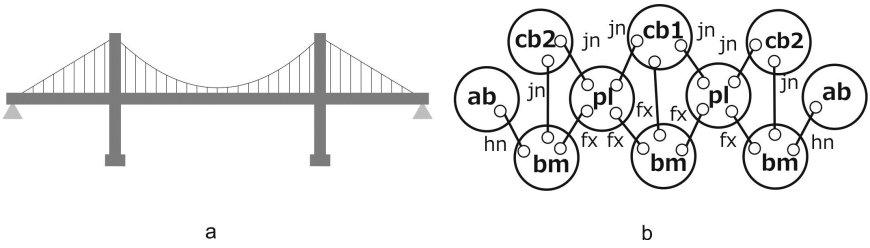

Fig. 2. A bridge and a CP-graph representing it.

3 VIZMOGR System Characteristics

The proposed system offers a simple graphical interface that allows users to visualize and analyse the dynamics of solutions for knowledge-based design tasks. The system enables the user to create CP-graphs corresponding to design drawings being early solutions.

At first, in the conceptualization phase, the user specifies icons representing object structural components the design drawings are to be generated of together with their labels. In this way a set Σ_V of labels which correspond to the icons is determined. Then, a set of graph nodes labelled by icons is created.

Exemplary icons, which are used in case of designing bridges to represent structural components of models, together with their labels are presented in Fig. 3a. Therefore set Σ_V contains labels *ab*, *pl, bm*, *sp*, *ar1, ar2, cb1, cb2* and *cb3*, which represent abutments, pylons, beams, supports, two types of arches, and three types of cables.

When we see CP-graph nodes with icons we are relying on the same neural machinery that is used to interpret the everyday environment. This natural semantics permeates the visual thinking. Abstract phrases such as *connected* or *contained within* are not considered metaphoric. In our approach the interplay between CP-graph nodes with icons is considered. Two CP-graph nodes connected by edges represent related components. The places of connections are represented by small circles in nodes called *bonds*. CP-graph nodes can contain bonds which are not connected by edges. They are called free and signal potential connections of a node with other nodes. CP-graph nodes with labels in the form of icons and with bonds assigned to them are shown in Fig. 3b.

Visual relations which can occur between icons are also specified and a set Σ_E of labels representing types of possible relations between object components, is determined. For bridges, the set Σ_E contains labels *fx, hn* and *jn*, which represent relations corresponding to fixed and hinged connections between components of bridges, and the join relation, respectively. When the user analyses relevant entities and organizes them

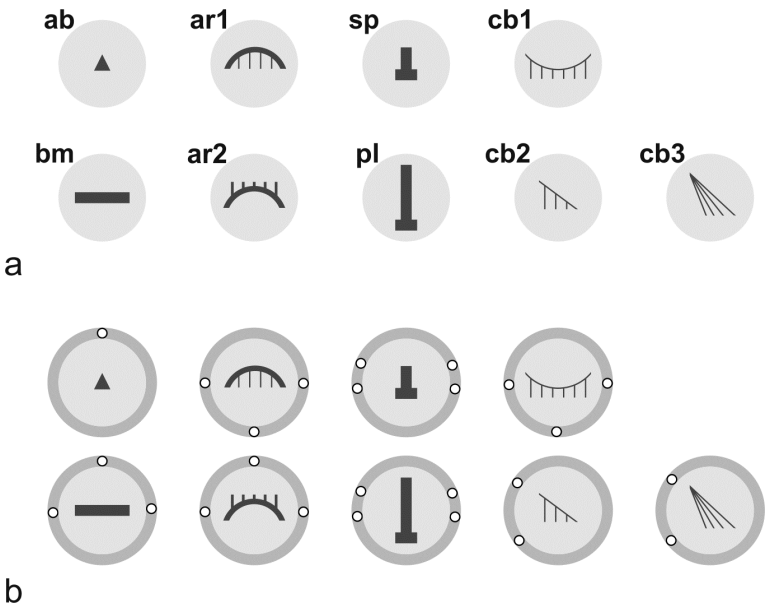

Fig. 3. (a) Icons representing components of bridges, (b) CP-graph nodes representing these components.

into concepts and relations, he can decide on shapes and colours of elements representing graph nodes. Moreover, for each node a set of attributes specifying properties of a corresponding component can be specified.

The CP-graph is created by adding selected nodes and connecting their bonds by edges representing relations between components corresponding to nodes. It is worth noting that starting with designing a fragment of the bridge structure in the form of a configuration of nodes with icons, the location of individual nodes can be easily determined by the relations (for instance *below*, *on the left*) among graphical primitives in the bridge drawing (see Fig. 4).

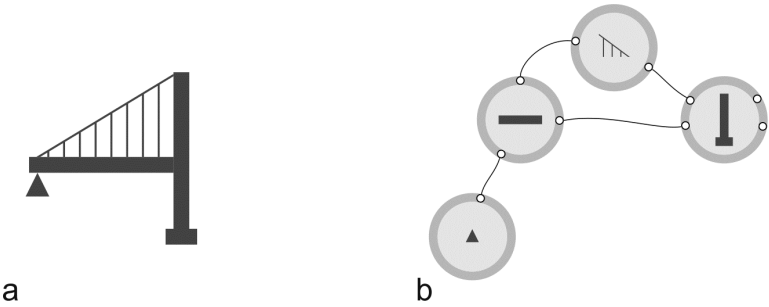

Fig. 4. (a) A fragment of a bridge (b) a CP-graph corresponding this fragment.

Creating edges between bonds of nodes is controlled by means of label-dependent rules. If the created edge is directed, its source and target bonds are automatically converted into out-bonds and in-bonds, respectively. The system gives also the possibility to label edges, remove nodes and edges, and change labels of both nodes and edges.

In Fig. 5 the first screen of GUI of VIZMOGR system is presented. In the left-hand side window the available icons with their labels are shown. In the right-hand side window a CP-graph representing a preliminary design can be created.

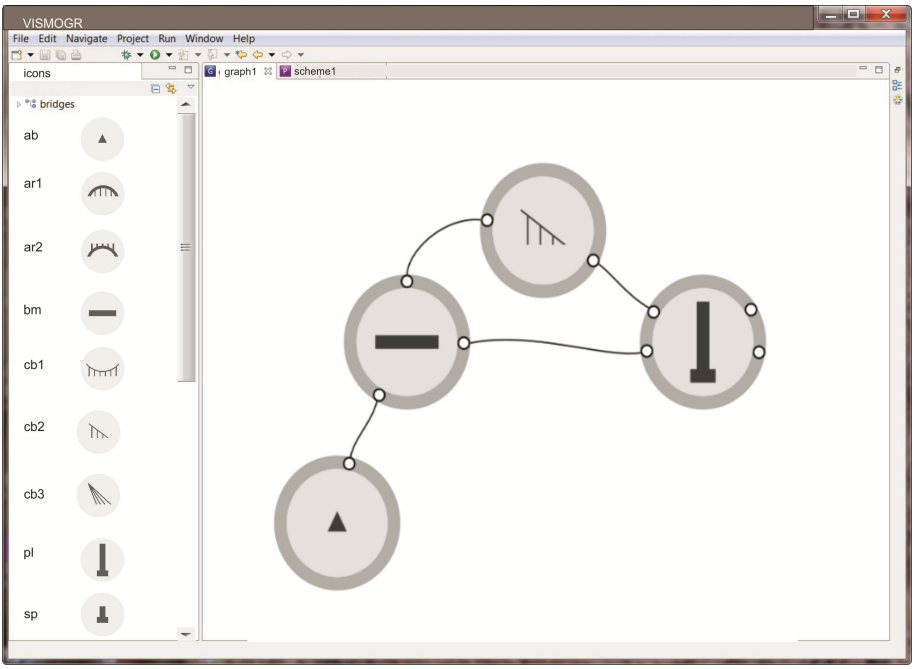

Fig. 5. A screenshot of the first part of VIZMOGR GUI, where CP-graphs can be created.

CP-graphs representing design drawings can be further modified by automatically applying sequences of CP-graph transformation rules selected by the user. The graph rewriting is based on the GraphTool engine [4, 5].

A CP-graph rewrite rule consists of two CP-graphs L and R (see Fig. 6a). The former (L) describes a subgraph of a derived CP-graph that after application of the rewriting operation is replaced by the latter (R), which is embedded in the rest CP-graph. Bonds of CP-graph nodes, which are not connected with edges are called free. They play an important role in the rewriting operation. Graphs L and R with the same number of free bonds can easily replace each other with the constant embedding in the rest part of the CP-graph.

Figure 6b shows a visualization of the rewriting rule with constant embedding for L and R. The rhombus, in which to each red solid edge one blue dotted edge is assigned, indicates a bond of R which replaces a bond of L in the derived CP-graph. VIZMOGR system allows users to define rewriting rules with other embeddings by drawing red

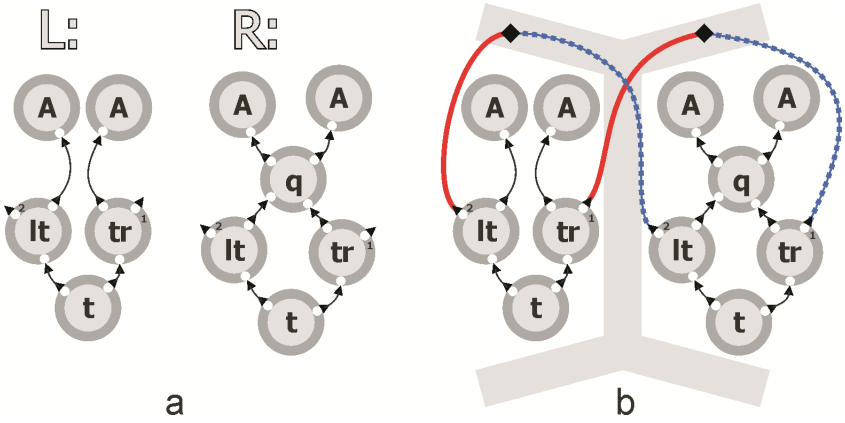

Fig. 6. (a) Two CP-graphs *L* and *R*, (b) a visualization of embedding. (Color figure online)

solid edges with rhombuses for all free bonds of the CP-graph *L* and drawing blue dotted edges connecting free bonds of the CP-graph *R* with appropriate rhombuses.

In order to modify created CP-graphs, firstly CP-graph transformation rules are specified by the user. VIZMOGR system offers five schemes of rules, which are presented in Fig. 7. These schemes correspond to rules which are most often applied in order to develop graphs. The rule presented in Fig. 7a allows for adding a new node and connecting its bond by an edge with a bond of the existing node. The rule from Fig. 7b allows for adding a new node and connecting its bonds by edges with bonds of the two already existing nodes. The rules shown in Figs. 7c–e allow for adding two new nodes with bonds connected by an edge and connecting bonds of them with bonds of the existing node. The difference in rules from Fig. 7c–e lies in location of free bonds in the nodes of the rule right-hand sides. The embedding of the scheme rules is indicated using rhombuses. The user has also the possibility to define his own rule schemes.

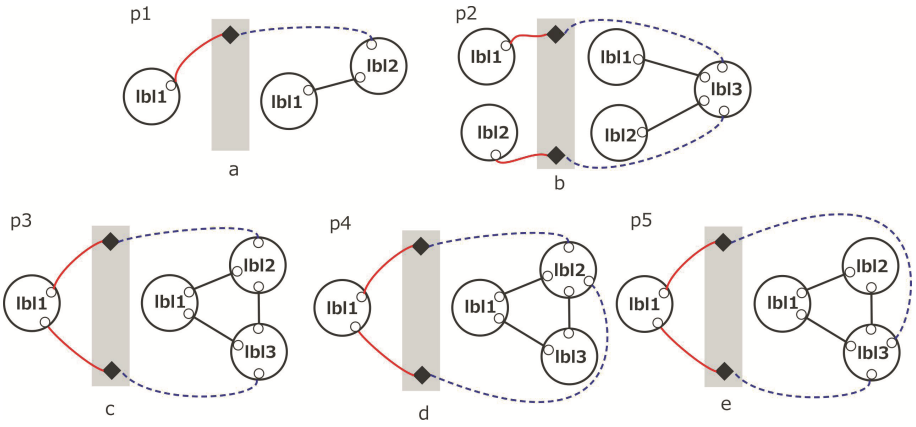

Fig. 7. Five CP-graph transformation rule schemes.

After selecting a given rule scheme the user gives labels of Σ_V to nodes and labels of Σ_E to edges, specifies numbers of bonds for nodes, and direction of edges. Schemes adapted for modelling bridges are presented in Fig. 8. The first and third scheme (*p1* and *p3*) have been adapted twice, while schemes *p2* and *p5* only once. It should be noted that after schemas adaptation the right and left-hand sides of rules can have different number of free bonds. This is admissible, as the embedding transformation is defined by rhombuses. We assume that the nodes of the left-hand side of a rule are found in a derived CP-graph if they have at least the same number of free bonds in this CP-graph. The other bonds, either engaged or free, assigned to the matched nodes remain unchanged during the rule application.

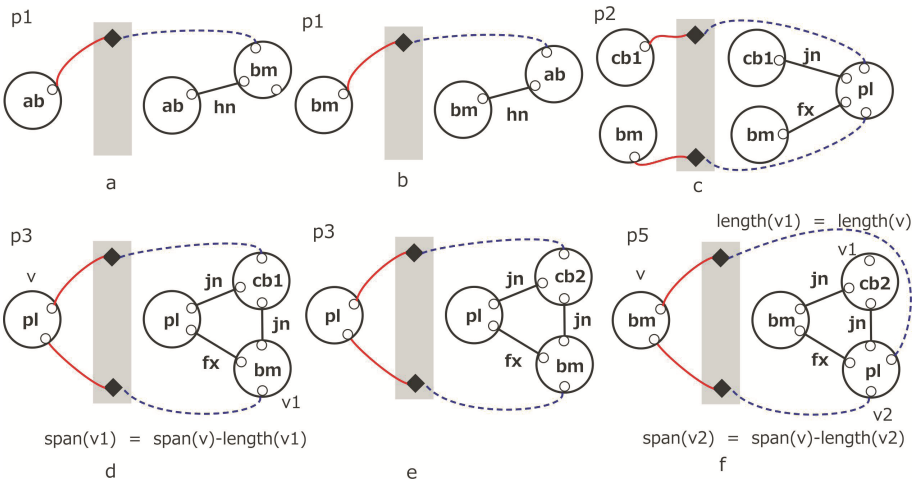

Fig. 8. CP-graph transformation rule schemes adapted for bridge modelling.

Applying CP-graph transformation rules leads to generation of different CP-graphs representing possible design object structures with geometry and material properties specified by graph attributes. There often exists a need to preserve or even to propagate some information throughout the modelling process. VIZMOGR system offers the support for attributed CP-graphs. Within each project attributes can be declared and assigned to CP-graph elements. The user defines the way of transferring the values of attributes from the left to the right hand side. The possibility of relating attributes of right-hand sides of CP-graph rules to attributes of their left-hand sides enables us to propagate semantic information and also to capture parametric modeling knowledge.

In case of bridge design to nodes of both sides of rules the attributes *length* and *span,* which specify the length of the corresponding bridge fragments and the remaining span of the bridge, respectively, are assigned. The initial value of the *span* attribute is decreased after adding nodes representing new bridge elements. For example in the rule presented in Fig. 8d the value of the attribute *span* of the node *v1* labelled *bm* on the rule right-hand side is equal to the value of the attribute *span* assign to the node *v* of the left-hand side decreased by the length of the beam represented by the node *v1* of the right

hand side. In the rule presented in Fig. 8f the value of the attribute *length* of the node *v1* labelled *cb2* on the rule right-hand side has to be the same as the value of the same attribute assigned to node *v* of the rule left-hand side.

Each rewriting rule can be equipped with a predicate of applicability specifying conditions under which the rule can be used. These predicates are in the form of expressions with parameters being attributes assigned to CP-graph nodes and edges and therefore describe semantic properties of CP-graphs to which rules can be applied. For example the rule, which corresponds to elongating the bridge by adding a new fragment (see Fig. 8d), can be applied under the condition that the remaining span of the bridge is longer than 100 m, i.e., $span(v) > 100$. Moreover, the rewriting rules can access and/or modify so called memory which is a common set of variables. Then successive rewriting steps can depend on the history of the previous ones.

In Fig. 9 the screen of GUI of VIZMOGR system, where CP-graph transformation rule schemes can be adapted to a given application domain, is presented. In the left-hand side window the available rule schemes are shown. The selected one is marked by the black frame. The same scheme during the adaptation process is shown in the central window. In the right-hand side window icons with their labels, which can be used in the adaptation, are presented. In the bottom window the values of attributes assigned to CP-graph nodes can be specified.

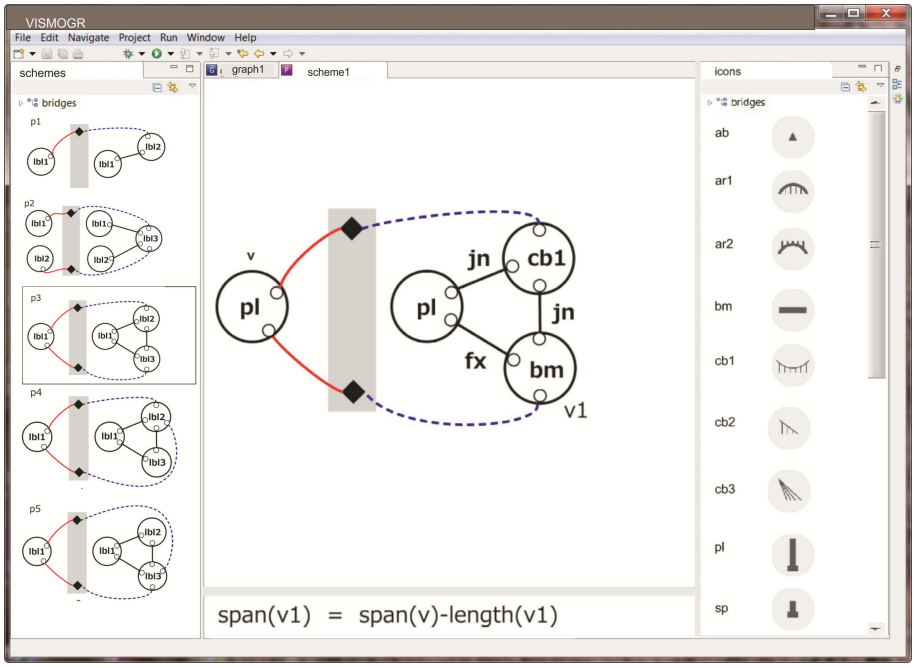

Fig. 9. A screenshot of the second part of VIZMOGR GUI, where CP-graph transformation rules can be adapted.

CP-graph transformation rules applied to initial CP-graph representations of drawings generate structures representing new designs. In VIZMOGR system the user can easily define the strategy of controlling CP-graph rewriting process, by drawing a graph, called control diagram, which specifies the possible order of applying CP-graph rules. The ordered sequences of rules, which can be obtained in this diagram, reflect the course of the design process.

In Fig. 10 a control diagram, which specifies the possible order of applying CP-graph transformation rules shown in Fig. 8, is presented. Rule numbers *p1* to *p6* correspond to rules illustrated in Figs. 8a to f, respectively. Starting from the node labelled *ab* and applying rules in the sequences *p1,p2*; *p1,p6,p5,p2*; *p1,p6,p4,p3,p5,p2* and *p1,p6,p4,p3,p4,p3,p4,p3,p5,p2* results in CP-graphs representing bridges from Figs. 11a–d, respectively.

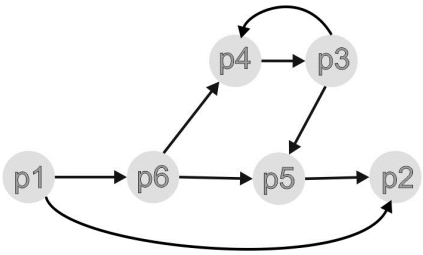

Fig. 10. A control diagram for bridge design.

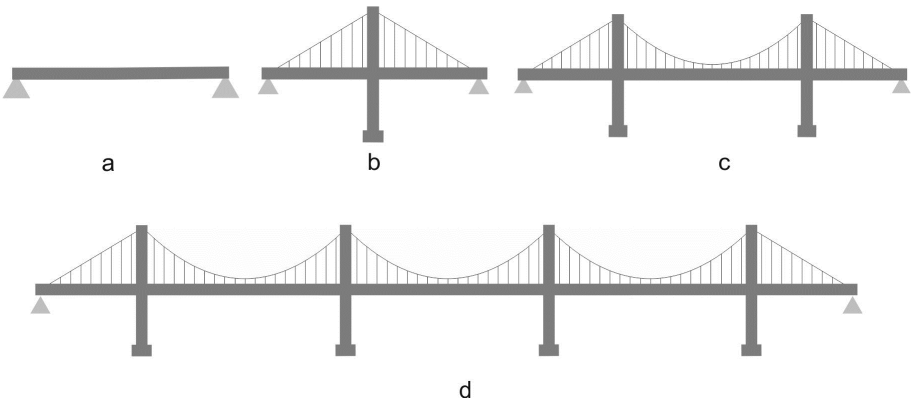

Fig. 11. Bridges represented by CP-graphs obtained using different sequences of rules.

The proposed VIZMOGR system is easy to adapt to a wide range of applications, as CP-graph elements and rule schemes are defined in a generic way. Sets of labels, nodes with bonds and transformation rules can be defined for multiple applications in different domains. The system is easy to use as the structure of available schemes of CP-graph transformation rules can be intuitively understood. Moreover, new schemes can be added in a simple way. Therefore, the proposed approach efficiently supports the task

of encoding various types of knowledge. The system also facilitates the specification of CP-graph rewriting rules by presenting steps of applying them in the visual way.

4 Conclusion

In this paper the prototype interactive system which supports modelling of engineering knowledge has been described. In the unified graphical environment the user can select icons representing object structural components and determine relations between them in order to obtain preliminary designs. On the basis of these designs their internal representations in the form of CP-graphs are obtained automatically. The conceptual phase of the object design process is also supported by providing schemes of CP-graph transformation rules in a visual editor. These schemes are adapted for a given application domain and used for modifying graph structures by applying sequences of rules selected by the user. In this way different CP-graphs representing possible design object structures with geometry and material properties specified by graph attributes are obtained. The presented examples show the way in which the system supports encoding knowledge needed to create bridge designs.

In the next steps of our research the catalogues of icons for various design domains will be developed. The memory of graph rules will be used to ensure the presence of a predefined number of selected components within a designed object and to preserve some required characteristics of the design. Moreover the system should enable the user to specify design requirements, and then by checking obtained CP-graph structures verify satisfaction of these requirements by the created design solutions.

References

1. McMahon, C.: Open issues in design informatics. In: Proceedings of the International Conference on Methods & Tools for CAE – Concepts and Applications, pp. 7–12. Bielsko-Biała (2017)
2. Helmes, B.: Object-Oriented Graph Grammars for Computational Design Synthesis (Dissertation), Institution Technische Universität München (2013)
3. Grabska, E.: Graphs and designing. In: Schneider, H.J., Ehrig, H. (eds.) Graph Transformations in Computer Science. LNCS, vol. 776, pp. 188–202. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57787-4_12
4. Palacz, W., Paszyńska, A., Świderska, I., Ślusarczyk, G., Strug, B., Grabska, E.: GraphTool: a visual support for generating graph models of artefacts. In: Proceedings of the International Conference on Methods & Tools for CAE – Concepts and Applications, pp. 81–86. Bielsko-Biała (2017)
5. Palacz, W., Paszyńska, A., Świderska, I., Ślusarczyk, G., Strug, B., Grabska, E.: GraphTool: case studies. In: Proceedings of the International Conference on Methods & Tools for CAE – Concepts and Applications, pp. 75–80. Bielsko-Biała (2017)
6. Fernandez, M., Kirchner, H., Pinaud, B.: Strategic Graph Rewriting: an Interactive Modelling Framework, [Research Report]. Inria, LaBRI, King's College London (2017)

7. French, M.J.: Conceptual Design for Engineers, 3rd edn. Springer, London (1999)
8. Strug, B., Paszyńska, A., Paszyński, M., Grabska, E.: Using a graph grammar system in the finite element method. Int. J. Appl. Math. Comput. Sci. **23**, 839–853 (2013)
9. Ware, C.: Visual Thinking for Design. Morgan Kaufmann, Elsevier (2008)