Representing designs by composition graphs

Representing Designs by Composition Graphs

Représentation de projets par graphes de composition

Die Wiedergabe von Entwürfen durch CP-Graphen

Adam BORKOWSKI
Prof.
IFTR
Warsaw, Poland

Adam Borkowski, born 1942, received his civil eng. degree at Techn. Univ. of Kaunas, Lithuania. He leads the Lab. of Adaptive Systems at the Inst. of Fundamental Technol. Res. (IFTR), Warsaw, Poland. His interests include applying AI-techniques to CAD and robotics.

Ewa GRABSKA
Assist. Prof.
Jagiellonian University
Cracow, Poland

Ewa Grabska, received her Ph.D. degree in math. from Jagiellonian Univ. in 1982. Since 1976 she has been with the Inst. of Computer Science of the same university. Her current research interests include graph transformations, computer graphics, CAD and computer aesthetics.

SUMMARY
The linguistic approach to the representation of knowledge about the shape and dimensions of the designed object is considered. A two-stage description is proposed: the composition graphs for representing topological features and the realisation schemes for generating instances of objects. Several examples of the conceptual designs of bridges demonstrate the ability of the proposed approach to support browsing through alternatives and adjusting geometrical properties.

RÉSUMÉ
Une approche linguistique est proposée pour la représentation de la connaissance de la forme et des dimensions d'un objet. Un graphe de composition représente la topologie et un schéma de réalisation produit les objets spécifiques. L'approche visuelle fournit des alternatives de représentation graphique pour chaque conception. Des exemples illustrent les différentes conceptions et démontrent le formalisme par le biais de propriétés géométriques ajustées.

ZUSAMMENFASSUNG
Der Artikel ist einer sprachorientierter Repräsentation des Wissens über Form und Abmessungen eines Entwurfsobjektes gewidmet. Eine zweistufige Beschreibung wiird vorgeschlagen: Die Mischdiagramme dienen der Festlegung von topologischen Eigenschaften des Objektes und die Entwurfsrepräsentationen generieren spezifische Austragungen. Ein paar Beispiele aus dem Gebiet des Brückenentwurfs zeigen, dass der vorgeschlagene Formalismus in der Lage ist, die Suche nach alternativen Lösungen und die Anpassung der geometrischen Eigenschaften zu unterstützen.

1. INTRODUCTION

Computer Aided Design systems were dedicated at the beginning of their history to pure geometrical modelling. Nowadays we expect much more from such tools: they evolve towards hollistic design environments that should facilitate an integration of the various aspects and phases of engineering design process [2, 11, 12]. Nevertheless, the geometric modelling remains the basis upon which all other elements of thinking about the designed object are built.

A possibility of applying the methodology of formal linguistic to the description of design objects was recognized early enough [10, 15]. An exhaustive discussion of that topic can be found in [5, 6], whereas the most recent contribution is probably the PhD-thesis [14].

Geometrical modelling can be discussed within the framework of graph transformations which constitute a relatively young discipline of computer science. Graph transformations are applied in many fields of computer science, because they are useful for modelling of a wide range of non-numerical computation [13]. The most interesting case in graph transformations arises when the effect of transformation is local, i.e., it can be specified by rules for transforming subgraphs of limited size. Allowing only a finite set of such rules leads to a syntactic model called a *graph grammar*. The process of transforming graphs by the iterative application of these rules is called *derivation* or *generating process* of the graph grammar.

A conventional approach to the shape grammar is to treat it as a generative system allowing the user to compose objects from geometrical primitives. The present paper develops further linguistic approach to the knowledge representation in engineering design. In the approach, the notion of graph grammar describing topological properties of the searched for objects is complemented by the concepts of *realization scheme* and *control diagram*. The realization scheme defines a specific instance of the generic object. Hence, it deals with the coordinates, transformations, dimensions and constraints. The control diagram selects specific productions from the grammar and defines the order of their application.

Given the limited length of the paper, we present in Section 2. only a brief summary of the proposed methodology. An exhaustive description of it can be found in [8]. Section 3. contains the discussion of the requirements that must be met by a system representing knowledge about the shape of the object to be designed. Several examples showing how the proposed formalism works in the design of bridges are given in Section 4..

2. CP-GRAPHS AND REALIZATION SCHEMES

In this Section the notions of *composition graph (CP-graph)* and *realization scheme* are presented. CP-graphs are directed labelled graphs which allow us to represent topological features of objects. Realization schemes are mappings which assign object instances to CP-graphs representing object topology [8].

Let us consider the bridge in Fig. 1a and its representation by means of the CP-graph in Fig. 1b. To describe all necessary relations between components of the bridge we would like to have a chance of describing on graph level the relations not only between the whole components (transformed primitives) but also between fragments of these components. To achieve this we equip graph nodes with two types of labels.

The first type of labels is the name of a primitive. The following prototypical parts: an abutment (ab), a beam (bm), a pylon (pl), a cable1 ($cb1$), and a cable2 ($cb2$) are distinguished for the bridge in Fig. 1a. The abbreviations of primitives ab, bm, pl, $cb1$, and $cb2$ are node-labels of the

CP-graph shown in Fig. 1b.

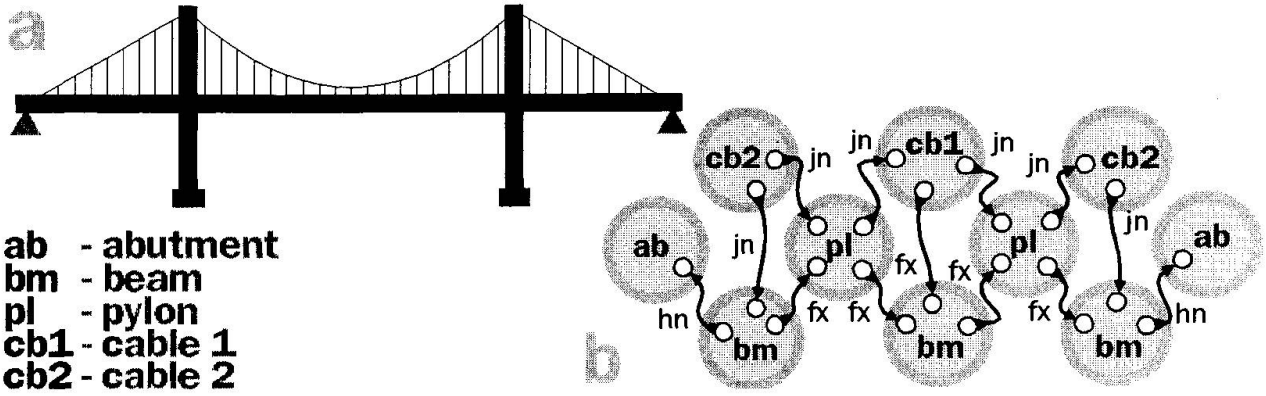

Fig. 1 The bridge and its CP-graph.

The second type of labels consists of node bonds. For a given node, in-bonds and out-bonds are defined. They correspond to fragments of the component representing by the node and express incoming and outcoming connections to and from the node.

We draw a picture of the CP-graph representing in-bonds and out-bonds as graphic symbols $\eth$ and $\hat{\eth}$, respectively, and edges as lines going from out-bonds to in-bonds.

Apart from labelling nodes of CP-graphs it is also useful to label their edges. Different edge labels can define different types of connections between the components of the designed object. In our example, the edge labels f_x and h_n denote fixed and hinged connections between components of the bridge. The edge label j_n denotes the join relation.

We show details of the bonding model for the part (Fig. 2a) of the bridge in Fig. 1a. Fig. 2b

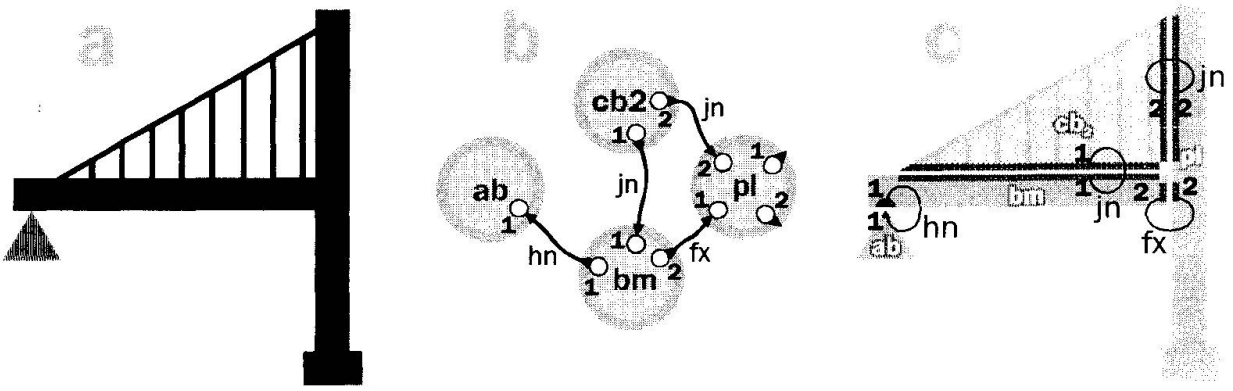

Fig. 2 The bonding model for the part of the bridge in Fig. 1.

presents the subgraph whose nodes with labels *ab*, *bm*, *cb2* and *pl* correspond to the abutment, beam, cable2 and pylon of this part, respectively, and the bonds of the nodes represent the fragments of the components (Fig. 2c). To each pair of bonds connected by the edge in Fig. 2b corresponds the pair of fragments of two components, which are marked by a circle in Fig. 2c.

Nodes of CP-graphs play not only a static role as building components but also a dynamic role as potential places to locate convenient CP-graphs. In Section 4. we shall use context-free CP-graph

grammar as a finite mechanism for producing CP-graphs which will represent relationships among components of bridges. A CP-graph grammar is similar to an ordinary context-free grammar. In the former, nodes of CP-graphs are rewritten by CP-graphs. In the latter, symbols of strings are replaced by strings. When using a grammar, rules of replacing nodes by other CP-graphs may be repeated and as a result of this similar parts of designs may appear [9].

Our approach to the knowledge representation in design makes it also possible to generate drawings of designs (graphical models). Ways of realizations of graphical models are defined by means of the so called realization schemes determined for CP-graphs.
A *realization scheme* for given set of CP-graphs consists of

1. a family of predicates which describe design criteria,
2. a set of admissible transformations in a Euclidean space,
3. a set of primitives in a Euclidean space,
4. a mapping, called the *prototype assignment*, which assigns primitives to the nodes of CP-graphs,
5. a mapping, called the *fragment assignment*, which assigns fragments of primitives to bond nodes of CP-graphs,
6. a function, called the *fitting function*, whose values are admissible transformations allowing to determine transformations which the primitives must undergo to transform them into design components.

Given a CP-graph and a realization scheme, a graphical model is obtained by a union of the transformed primitives, where the primitives directly and the transformations indirectly are specified by the realization scheme [9]. For the CP-graph shown in Fig. 3a three drawings of

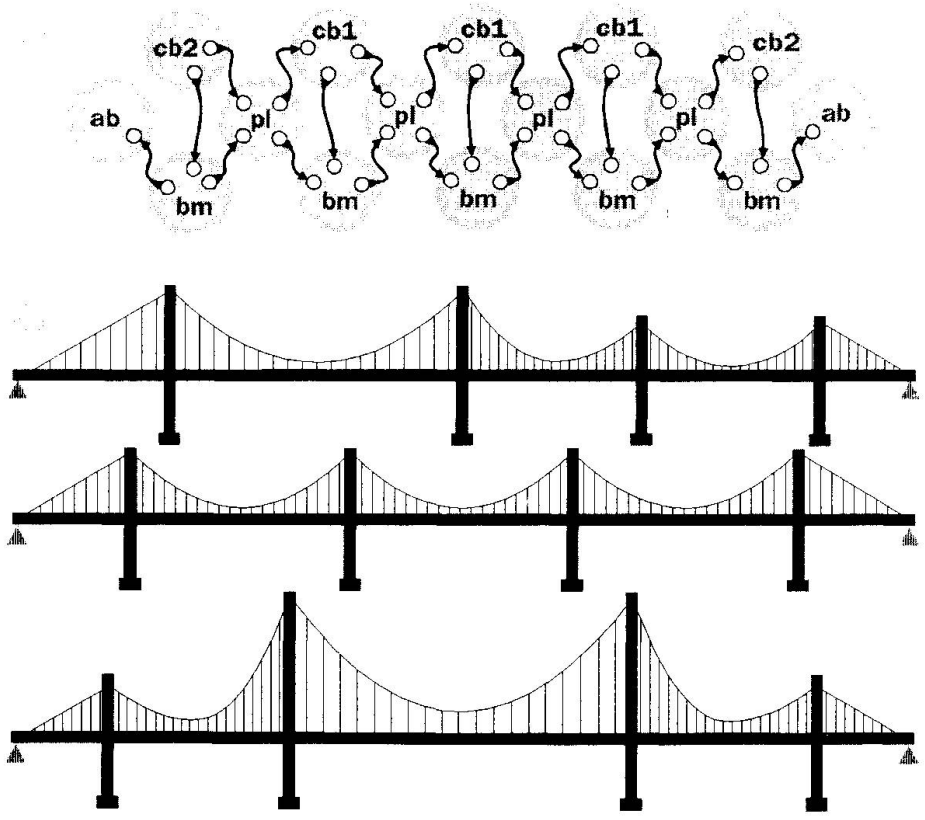

Fig. 3 A two stage description: a) the CP-graph representing topological features of potential bridges, b) three bridges specified by three different realization schemes for that CP-graph.

bridges presented in Fig. 3b are determined by three different realization schemes with the same set of primitives but different sets of admissible transformations and different fitting functions.

The fact that the structure of the object described by the CP-graph is independent of its realization may be useful in the process of designing. Due to it we can design an object applying one of two strategies: one that consists in changing of connections between nodes of the CP-graph, while primitives remain unchanged, or the other changing primitives or transformations, while the CP-graph remains unchanged.

3. REPRESENTING KNOWLEDGE ABOUT DESIGN

It is commonly agreed to distinguish two aspects of computer-aided design: the product modelling and the process modelling [12]. The former deals with more or less static representation of the knowledge about a specific class of artefacts. The latter should reflect the dynamic character of search for alternative designs. Our methodology conforms with such a dual description. The CP-graph generated by the grammar corresponds to the product model: it defines topological properties of the structure. The realization scheme and the control diagram describe the process model allowing the user to explore the space of admissible solutions.

Each engineering object can be seen as an assembly of parts occupying specific portions of a Euclidean space. Such a decomposition can be performed recursively resulting in a hierarchical tree. The root of this tree corresponds to the entire object, intermediate nodes represent its subsystems and the leaves correspond to primitives.

Typically the design process follows the top-down pattern: at first the layout of the main subsystems is found, then each of them is considered separately [11]. If at a certain level of the decomposition tree an unsolvable situation is detected, then the process is turned back to the higher level in order to find an alternative solution. Thus engineering design can be seen as a repetitive transversal of the spiral path from the general idea of the artefact towards its detailed project. The trial-and-error nature of the search of optimum solution, inevitable due to complexity of real world problems, shows where assistance of the computer is most efficient. It should enable the human designer to explore the space of plausible solutions in a comfortable and efficient manner. It is also of great importance that modifications introduced at a particular level of the decomposed problem are properly propagated throughout the database in order to maintain it in a consistent state.

The old good rule says: never submerse into details before considering general goal to be accomplished. Thus, designing an engineering object should begin with a conceptual phase, where a notion of plausible solution is worked out. In our formalism this means establishing graph-production rules which express possible topological relations between the principal parts of the desired object. This can be done using a special purpose graph-editor.

After the user has defined the rules, the system knows the universe of plausible solutions. Upon request these rules can be browsed and edited by the user. Next the realization scheme is to be described. Here the user must enter data related to the coordinates of the designed object with respect to a specific reference frame, define characteristic dimensions of the object and write the predicates responsible for geometrical constraints.

Finally, the control diagram, defining which production rules in which order are to be applied, is to be specified. This completes the description of the object and the visualization module can be invoked in order to evaluate the result and to decide upon possible changes. The usage of the proposed system is easiest to explain on examples.

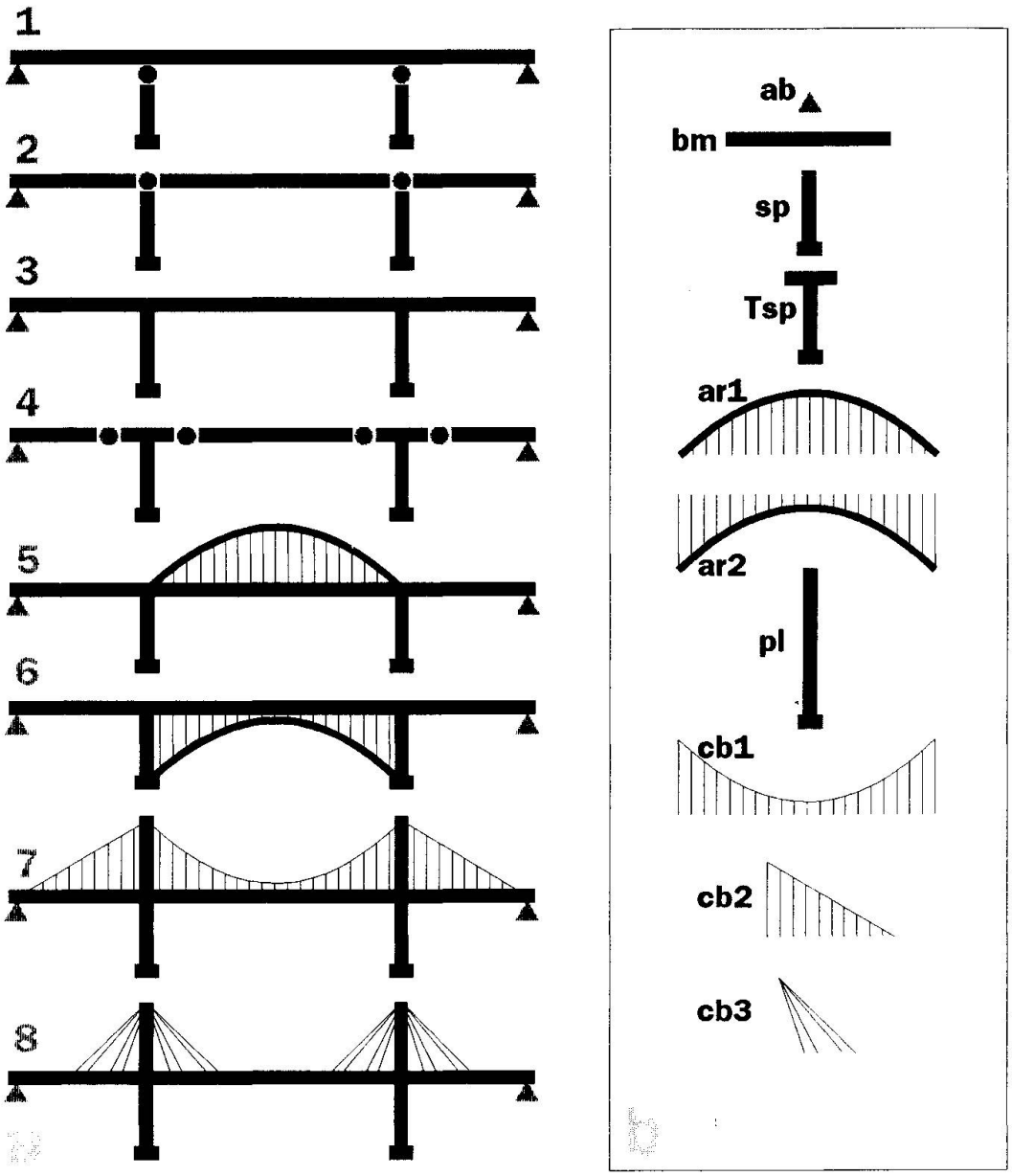

Fig. 4 Different types of bridges with their primitives: a) eight types of bridges, b) the set of labelled primitives for the bridges.

4. EXAMPLES OF APPLICATION

In this Section we give several examples to show how the proposed formalism works in the design of bridges. We define explicit syntaxes and semantics for computer modelling of bridges.

Fig. 4a presents different types of bridges which can be generated by CP-graph grammar. Fig. 4b shows the set of primitives with the abbreviations of their names. In Fig. 5 we present 6 syntactic rules $p_1 \div p_6$ with their typical interpretations $i1 \div i2$, which allow to generate the bridge numbered 7 depicted in Fig. 4a, and its modifications. The grammar for generation of all bridges shown in Fig. 4a contains 33 rules. As we can see, each rule is composed of two parts - its left-hand side (a CP-graph node) and its right-hand side (a CP-graph). The symbol $\Longrightarrow$ serves as a replacement operator which informs that the left-hand side of a rule is to

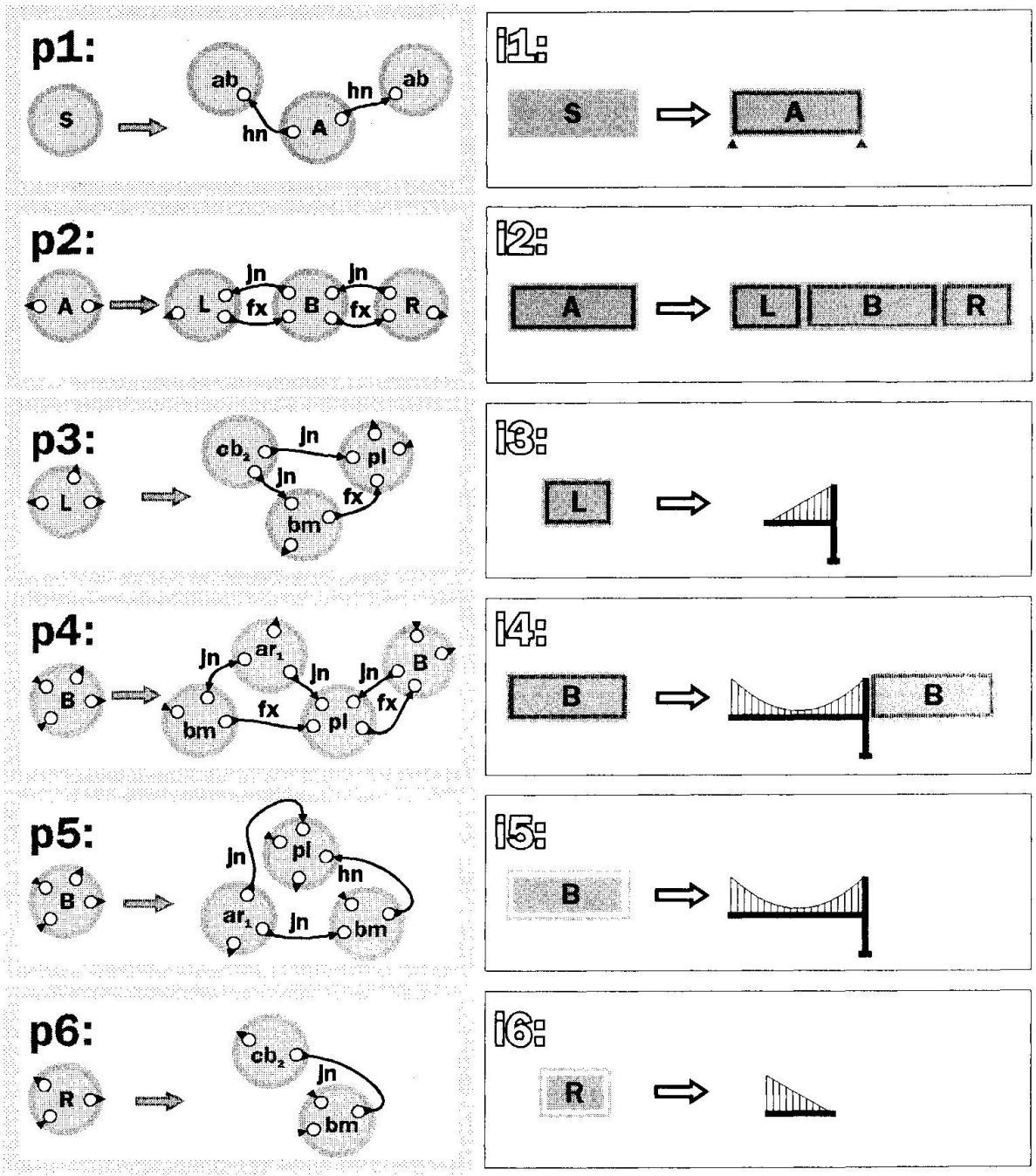

Fig. 5 Rules and their interpretations for the bridge numbered 7 in Fig. 4a and its modifacations.

be replaced by the right-hand side of the rule. The nodes of CP-graphs in rules, with labels being the abbreviations of names of primitives (Fig. 3b), correspond to these primitives and are terminal. The remaining CP-graph nodes are nonterminal. The edge labels f_x, h_n denote fixed and hinged connections between components. The edge-label j_n denotes the joint relation between components.

CAD requires knowledge about plans or sequences of rules rather than simply about individual rules. Therefore some means of representing and using control knowledge for CP-graph grammars are needed. The approach proposed in the following was originally introduced in [7]. We use a graphical notation for the control algorithm, which seems to be more illustrative [4].

A *control diagram* for a finite set P of productions is a connected directed labelled graph with the alphabet $P \cup \{I, F\}$ of node labels and satisfying the following conditions:

1. there exists exactly one initial node labelled with I,
2. there exists exactly one final node labelled with F,
3. there exists no edge terminating in the initial node, and
4. there exists no edge leaving the final node.

An example of a control diagram is shown in Fig. 6. With exception of the initial and the final

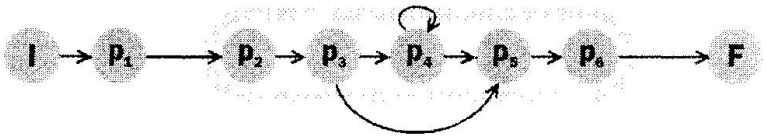

Fig. 6 The control diagram for the grammar rules in Fig. 5.

node, all other nodes are labelled with productions. Applying a derivation process according to the order stated in the control diagram, we start with a production which is the label of a direct successor of the initial node. The derivation process stops when the final node is reached. Each path from the initial node to the final node allows us to generate the CP-graph belonging to the CP-graph language under consideration. The control diagram shown in Fig. 6 allows to generate such CP-graphs which describe the structure of the bridges numbered 7 in Fig. 4a, and its modifications.

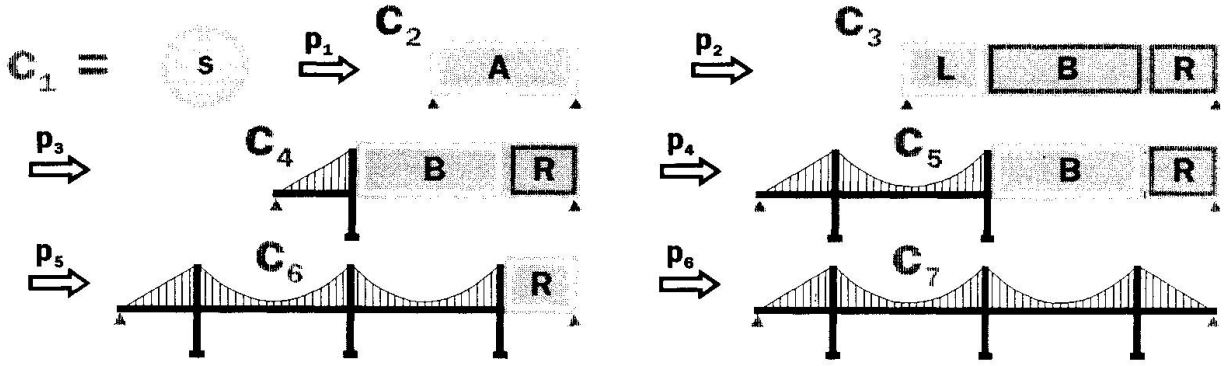

Fig. 7 Generation of a modification of the bridge 7.

When equipping terminal nodes of the generated CP-graphs with attributes being transformations we create drawing of bridges. Drawings corresponding to such CP-graphs are the unions of primitives determined by labels of terminal nodes and transformed according to the attributes assigned to these nodes. The primitives directly and the transformations indirectly are specified by the realization scheme. Each drawing can be formed step by step simultaneously with the derivation of the CP-graph. Fig. 7 presents fragments of the drawing and non-terminal parts obtained during the generation of the CP-graph describing the bridge being a modification of the bridge 7 in Fig. 4a. In this generation we start with the CP-graph node c_1 with label S and generate the sequence of CP-graphs c_2, c_3, c_4, c_5, and c_6, by means of the sequence of the productions p_1, p_2, p_3, p_4, p_5, p_6 determining one of the pathes of the control diagram in Fig. 6.

A typical session with the computer program based upon the proposed methodology is the last example which will be considered in the paper. Fig. 8a shows the initial situation faced by the designer: a profile of a river valley that should be transversed by the bridge. Choosing from the

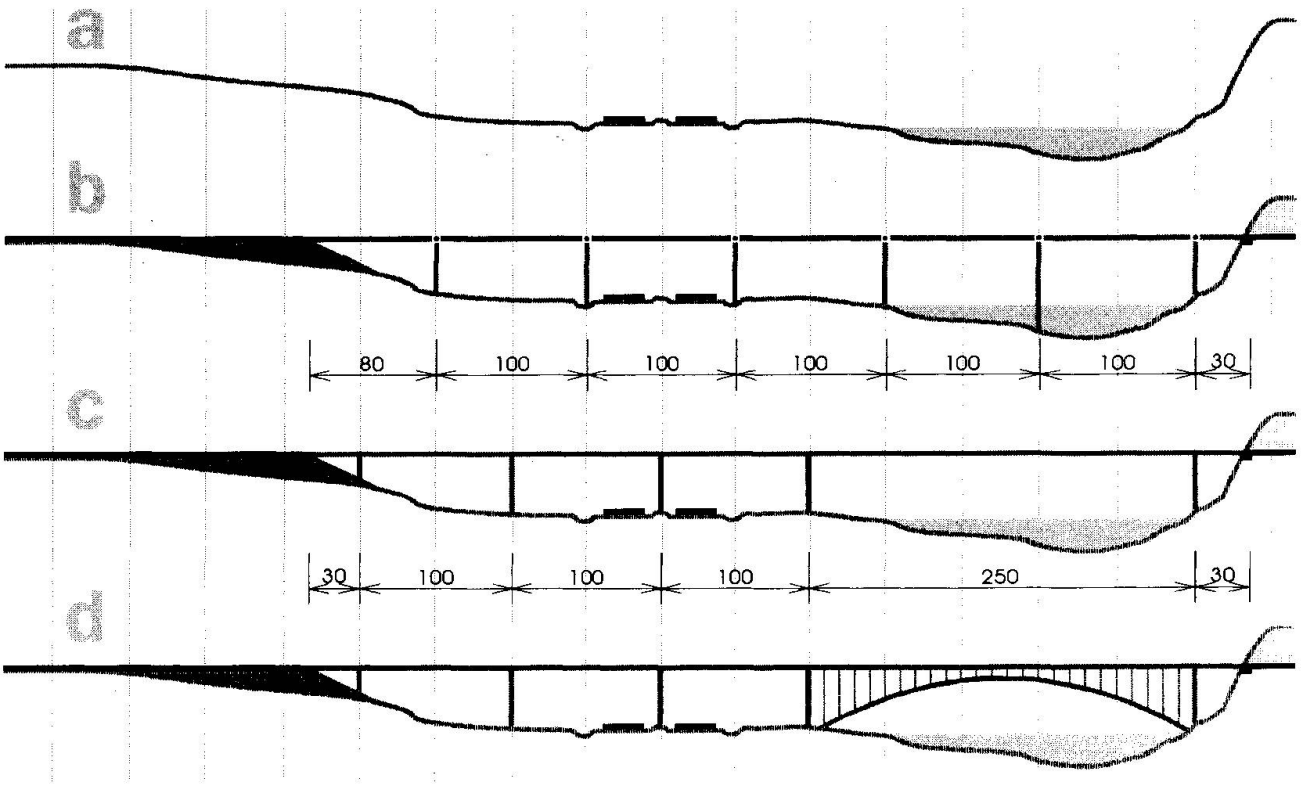

Fig. 8 Choosing the layout of a bridge: a) the cross-section of the valley, b) initial version - a bridge composed of the simple supported beams, c) a continuos beam solution - the support standing in the river removed, d) the arch added for aesthetical reasons.

alternatives proposed by the program the user takes the chain of 6 simply supported beams as the first alternative (Fig. 8b). It is easy to try different positions of the supports because this requires only the changes of few numerical parameters (the lengths of spans) in the realization scheme. Note that the knowledge about plausible span lengths is stored in the constraints of the realization scheme.

Assume that the user is not satisfied with the scheme shown in Fig. 8b. It presumes placing a support in the river itself which is expensive and might be opposed by the ship owners. An obvious alternative is the continuous beam bridge over the river shown in Fig. 8c. Removing a support demands removing an appropriate production in the production sequence of the control diagram and the changes of the lengths of spans in the realization scheme.

For aesthetical reasons one might consider putting an arch over the river (Fig. 8d). In order to generate such a bridge it is sufficient to add an appropriate production to the production sequence, which allows to add a new node (representing the arch primitive) to the CP-graph. Again the user can easily adjust the lengths or consider various support conditions for individual spans.

5. CONCLUSION

Preliminary experience gained so far indicates that the proposed format is quite convenient for storing and manipulating knowledge about the topology and geometry of the designed object. A clear cut separation of the topological backbone and of the specific realization allows the de-

signer to investigate alternative solutions at two levels. Generating alternative objects amounts simply to the transversal of the control graph, whereas scaling and adjusting the dimensions of the object is performed by changing the predicates and parameters of the realization scheme.

Our further aims include merging together composition graphs related to the different perspectives of the design (the architectural layout, the load carrying structure etc., compare [1]). This can be accomplished by introducing a meta-grammar and developing hierarchical structure of the CP-graphs.

REFERENCES

1. BAILEY S. F, SMITH I., Structural and Architectural Integration in Building Design. Proceedings of the EG-SEA-AI Workshop on Artificial Intelligence in Civil Engineering, Lausanne, pp.327-341, 1994.
2. BOULANGER S., SMITH I., Models of design processes. Proceedings of the EGSEA Workshop, Lausanne, 1994, pp. 132-145.
3. BORKOWSKI A., JÓŹWIAK S., FLEISCHMANN N., On procedural and declarative aspects of optimum structural design. Progress in Structural Engineering, eds. D. Grierson, A. Franchi and P. Riva, Kluwer Academic Publishers, Dortrecht, 1991, 279-292.
4. BUNKE H., Programmed Graph Grammars. Lecture Notes in Computer Science, 1977.
5. COYNE, R. D., ROSENMAN, M. A., RADFORD, A. D., BALACHANDRAN, M. and GERO, J. S., Knowledge-Based Design Systems. Addison-Wesley, New York, 1990.
6. FENVES S. J., BAKER N. C., Spatial and functional representation language for structural design. Expert System in Computer-Aided Design, J.S. Gero (ed.), North-Holland, Amsterdam, 1987, pp. 511-530.
7. GEORGEFF M. P., Procedural Control in Production Systems. Artificial Intelligence 18, 1982.
8. GRABSKA E., Graphs and designing. Lecture Notes in Computer Science: Graph Transformations in Computer Science, 776, Springer-Verlag, Berlin, 1994, pp. 188-203.
9. GRABSKA E., Theoretical Concepts of Graphical Modeling, Part two: CP-graph Grammars and Languages. Machine GRAPHICS and VISION, 2(2), 1993, pp. 149-178.
10. MARCH L., STINY G., Spatial systems in architecture and design: some history and logic. Environment and Planning B, 12, 1985, pp. 31-53.
11. SAUSE R., POWELL, G. H., A design process model for computer integrated structural engineering. Engineering and Computers, 7, 1991, pp. 145-160.
12. SCHERER R. J., Integrated product and process model for the design of reinforced concrete structures. Proceedings of the EGSEA Workshop, Lausanne, 1994, pp. 132-145.
13. SCHNEIDER H. J., EHRIG H., Graph Transformations in Computer Science. Lecture Notes in Computer Science, 776, Springer-Verlag, Berlin, 1994.
14. SHIH S. G., The use of string grammars in architectural design. PhD Dissertation, Dept. of Architecture, ETH Zurich, 1994.
15. STINY G., Introduction to shape and shape grammars, Environment and Planning B, 7, 1980, pp. 343-351.