Layout Algorithms of Graph-Like Diagrams for GRADE Windows Graphic Editors

Paulis ĶIKUSTS and Pēteris RUČEVSKIS

Institute of Mathematics and Computer Science
University of Latvia, 29 Rainis blvd., Riga, Latvia
paulis@cclu.lv, rpeteris@cclu.lv

Abstract. We propose a set of layout operations ensuring flexible and convenient interactive editing of communication diagrams and nested entity-relationship models having textual labels on connections. The set includes several procedures for incremental diagram layout. Tools for fully automatic layout and for direct manual painting of graphic primitives are also integrated in a single system. In this way we have filled to some extent the gap between both extremal levels of editing.

1 Introduction

GRADE Windows is a CASE tool, based on specification language GRAPES/4GL [1]. There are communication diagrams (CD) and entity-relationship models (ER) among system models built by GRADE Windows (Fig. 1, 2). Such diagrams are graph-like structures and consist of elements of two principal kinds: *blocks* and *connections*. The graphical notations used for ER models are those introduced by J.Martin. An additional graphic element is a notation for nested entities shown by placing descendent entities inside parent entities. Thus, we allow more general graph-like diagrams in which blocks are arbitrary nested without combinatorial restrictions for connections. Moreover, blocks and connections may be supplied with text areas called *superscriptions* which are the third principal kind of diagram elements. In such a way our ER diagrams generalize K.Sugiyama's and K.Misue's compound digraphs [2].

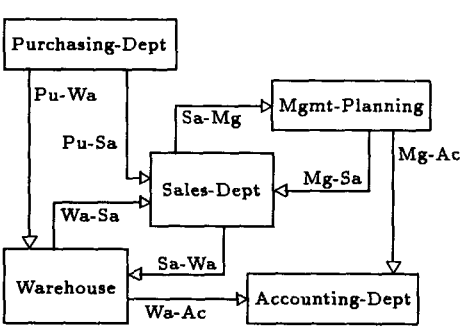

Fig. 1

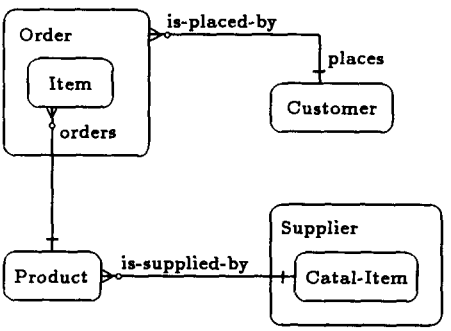

Fig. 2

Creating a complex diagram using only simple painting tools is very complicated process because adding or displacing even a single block may cause a need to change the placement of all diagram elements. Fully automatic layout

of graph-like structures is a well-studied problem and there are a great amount of layout algorithms for entire diagrams [3]. However, since creating a complex diagram is a gradual interactive process during which only fragments of diagram are presented, fully automatic tool may produced fragment layouts that are non-adequate for desired final layout of the whole diagram. Thus, the gap between the both extremal levels of editing deserves attention, but only few papers [4, 5] have focused on this topic.

We propose a set of layout operations ensuring flexible and convenient interactive editing of CD and ER diagrams. The set of our operations includes several procedures oriented on processing of intermediate stages of layouts. Tools for fully automatic layout and for direct manual painting of graphic primitives are also integrated in a single system. In this way we have filled to some extent the gap between the both extremal levels of editing.

2 Principles of Algorithms and Data Structures

All our editing procedures are grouped into several kinds of *editing modes*, the most important of which are AUTOMATIC and MANUAL. AUTOMATIC mode has the strongest set of geometric constraints maintained by the editors: contours of the block figures do not cross each other; block figures of non-nested blocks do not overlap; connection lines have no common segments and do not intersect CD block figures; superscriptions do neither overlap each other, nor block figures, nor connection lines. MANUAL mode has only one constraint: contours of the nested block figures do not cross each other. Fast procedures switching among the modes have also been implemented, and they are important additional elements for filling the gap between the extremal levels of editing.

In CD and ER editors geometry calculations are performed with a simplified representation of the actual graphical data: block figures and text areas are represented by their smallest enclosing rectangles, connections lines are modified accordingly. The obtained rectangles and line segments form the primary geometric data (PGD) structure (see Fig. 3 for our ER example). All newly interactively or automatically created diagram elements are inserted into the PGD-structure maintaining the constraints.

While in MANUAL mode CD and ER editors work directly with the PGD-structure, there are two secondary geometric data structures in the case of AUTOMATIC mode: H-structure and V-structure. H(V)-structure consists of rectangles of three sorts: (1) slim rectangles of width 2δ that are obtained by spreading each horizontal (vertical) segment of block rectangles and connection lines from PGD-structure by constant width δ in all four axes-parallel directions; (2) the block superscription rectangles; (3) the connection superscription rectangles pushed aside by distance δ from adjacent connection line segments. Fig. 4 shows H-structure of our ER example (dashed lines copy Fig. 3). Fast access to rectangles from H, V-structures are ensured by means of well known regular cell structure associated with them.

Our basic procedure that acts with rectangles from H-structure (V-structure) is the shifting procedure $Shift$. The procedure takes care of freeing a place of the

necessary size among diagram elements: when an axes-parallel vector is given, *Shift* recursively translates all the rectangles from an adjacent rectangular area by appropriate shorter vectors, and then translates the given rectangle by the given vector.

Procedure *Shift*, together with four converting procedures *MakeHstructure, MakeVstructure, MakePGDfromH, MakePGD fromV*, are the basic tools for performing geometric modifications of diagrams of both kinds without destroying their topological structure and maintaining all of the constraints in AUTOMATIC mode at the same time.

Overlappings of non-nested block rectangles are eliminated by means of a special procedure *NormalizeBlockDistances* which is based on *Shift* and works with arbitrarily placed block rectangles. As the procedure is useful also for adjusting specially calculated allocations of blocks, we can represent the given diagram by an undirected graph and use its drawing as a sketch for the layout.

We have implemented two graph drawing algorithms: one for 2-connected graphs and another for graphs with cutting vertices. In the case of a 2-connected graph, first, we uniformly lay out its vertices, and, second, improve their positions by well known barycentric method.

Graphs with cutting vertices are processed in a more complicated way as follows. First, we build a special cutting vertex-bicomponent tree (CVB-tree). The vertex set of a CVB-tree consists of the cutting vertices of the given graph, and vertices associated with its bicomponents that are not single edges. The set of edges of a CVB-tree consists of all pairs (c, b), where c is a cutting vertex and b is a vertex associated with the bicomponent containing c. In addition, each vertex of the CVB-tree is associated with a disk. The area of the disk is taken proportional to the total area of all block rectangles represented by the same CVB-vertex.

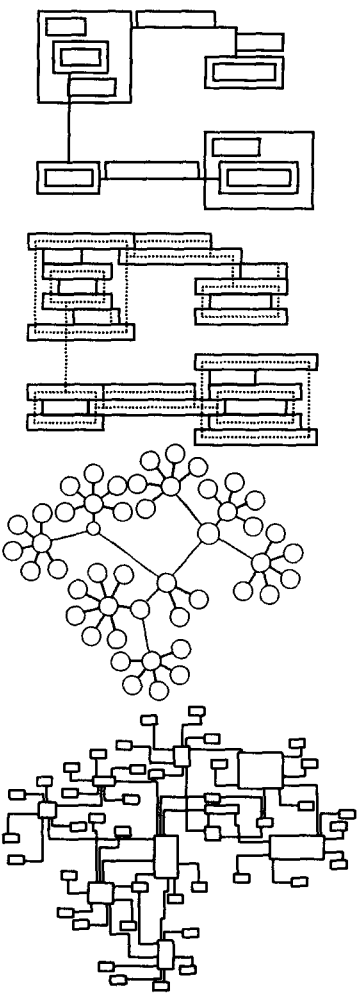

Fig. 3, 4, 5, 6

The next stage is to lay out the disks in non-overlapping way in order to get some initial planar representation of the CVB-tree. After this is done we compact iteratively the initial layout by translating both subtrees of each edge of the CVB-tree each towards the other. The disks are not allowed to intersect

any region between any two adjacent disks bounded by their common outer tangents.

Finally, we place the vertices of the given graph into corresponding disks and adjust locations of the non-cutting vertices by barycentric method. Examples of drawings of both a CVB-tree and a corresponding PGD-structure with raw connection lines are given in Fig. 5, 6.

3 Conclusions

In this paper we have sketched the most important details of our data structures and algorithms on which various layout procedures for graph-like diagrams are based. H and V-structures represent diagram blocks, connections and superscriptions as a homogeneous set of rectangles. This ensures automatic transfiguration of layouts without destroying the diagram readability for relatively simple CD diagrams as well as arbitrarily nested ER diagrams.

K.Miriyala, S.W.Hornick and R.Tamassia [4] ask "what do we do if there is no space to place the arc label?" and "what happens if there is no space for the expanded version of the node?" Our approach gives a natural answer to these and similar questions: the procedures *Shift* and *NormalizeBlockDistances* are the very algoritmic tools which via the H and V-structures take care of freeing place of necessary size for the new object and eliminate overlappings of the other objects performing sufficiently small number of automatic moving of diagram elements.

Acknowledgements

The authors would like to thank the GRADE development group at Institute of Mathematics and Computer Science of University of Latvia for enabling them to participate in an interesting collaborative project, and particularly Ilona Etmane and Kārlis Podnieks. We would also like to thank Alvis Brāzma for carefully reading and discussing the manuscript.

The work was supported by the grant 93.601 of Latvian Council of Science, by Software House Riga, and by Infologistik GmbH, Munich.

References

1. GRADE V1.0: Modeling and Development Environment for GRAPES-86 and GRAPES/ 4GL, User Guide. – Siemens Nixdorf, 1993.
2. K.Sugiyama, K.Misue. Visualization of structural information: automatic drawing of compound digraphs. – *IEEE Trans. Syst. Man, Cybern.*, vol. 21, no. 4, 1991, pp. 876–892.
3. G.Di Batista, P.Eades, R.Tamassia, I.G.Tollis. Algorithms for drawing graphs: an annotated bibliography – *Computational Geometry: Theory and Applications*, vol. 4, no. 5, 1994, pp. 235–282.
4. K.Miriyala, S.W.Hornick, R.Tamassia. An incremental approach to aesthetic graph layout, – *Proc. Int. Workshop on Computer-Aided Software Engineering (CASE '93)*, 1993.
5. A.Papakostas, I.G.Tollis. Issues in interactive orthogonal graph drawing, – *Proc. Graph Drawing '95*, in *Lecture Notes in Computer Science*, this volume.