Graph Widget: A Tool for Automatic Data Visualization*

Paulo Quaresma† pq@fct.unl.pt
José G. P. Lopes gpl@fct.unl.pt

Centro de Inteligência Artificial, UNINOVA
Quinta da Torre
2825 Monte da Caparica
Portugal
phone: +351 1 2954464 ext: 1360
July 27, 1991

Abstract

A new X-Windows object — *Graph widget*[1]— will be presented. It handles graphs using one of the three graph layout functions implemented: trees, hierarchical graphs and general graphs. The widget interface to Prolog allowing its utilization in logical programming environments and some of the possible widget applications will also be described.

1 Introduction

Graphs have been used to model reality and to represent relationships between objects in many fields of computer science. In the last years the need for displaying automatically graphs has increased with the development of man-machine interfaces and with the utilization of environments for software engineering.

A new object — *Graph widget* — was defined as a X subclass implemented in C using the X Toolkit Intrinsics and build as general and domain-free as possible allowing its future utilization in different fields. An interface to X-Prolog, a Prolog environment enabling the access to the X-Windows functionalities, was implemented as well as a graph editor in Prolog.

2 Graph Widget

In order to achieve the functionalities desired the widget class *GraphWidget-Class* was created as a subclass of *Constraint*, allowing the objects to have children (i.e. nodes) with constraints.

*This work was funded by JNICT, through project "Communication in Portuguese with Computers", contract n^o PMCT/MIC/87439, and project "Robust Understanding of Portuguese", under contract n^o PMCT/P/TIT/167/90; INIC through project CALIPSO; FCT/UNL and Gabinete de Filosofia do Conhecimento

†Owns a scolarship from JNICT, reference n^o PMCT/BIC/152/90

As main features the user is allowed to: choose and define the graph layout function, toggle between automatic and manual mode, define different styles for each edge, select the graph layout orientation and define the default edge distance between nodes.

2.1 Layout functions

Three graph layout algorithms were implemented: trees, hierarchical graphs and general graphs.

2.1.1 Trees

The Reingold and Tilford tree layout algorithm was implemented ([6]) and it was generalized in order to handle both n-ary trees and general graphs. On the other hand, as a dynamic algorithm was desired, it was also changed according to the work by Moen ([4]). It allows nodes of any shape and allows insertion/deletion of nodes.

2.1.2 Hierarchical Graphs

The algorithm to represent hierarchical graphs by Sugiyama was chosen ([8]). As the original algorithm doesn't layout cycles, Rowe's suggestion ([7]) was used to temporarily invert edges. To minimize the number of edge crossings in each level an heuristics based on a combination of Gansner's ([2]) and Rowe's work was used.

2.1.3 General Graphs

We have chosen to implement Kamada's algorithm for drawing general graphs ([3]). This algorithm treats the graphical display of general graphs as a virtual dynamic system in which every two nodes are connected by a spring with some length. The algorithm tries to achieve a state in which the total spring energy is minimal.

During the implementation phase some problems were detected and solved, namely the assignment of values to the constants needed by the algorithm and the definition of stronger end conditions.

3 Interface to Prolog

Our initial motivation for this work was to build an interface to Prolog allowing the graph widget to be used as a new object in the logical programming environment.

In order to support the new widget we used Quintus Prolog in the framework of the ALPES environment for logic programming (developed as a consequence of the ESPRIT project 973, see [1]), and we implemented a graph editor with some basic functions to demonstrate the final result.

4 Conclusions and Future work

The graph widget was already used in two kinds of applications: natural language understanding (it enabled the graphical display of conceptual graphs, see [9]) and in the ESPRIT project CIM-PLATO (task $n^o 2202$) as a tool for developing interfaces ([5]). Besides, this widget can be the basic tool for the development of an integrated logical programming environment as it can be easily used to implement several development and debugging tools (browsers, program call graphs, graph editors, etc).

However it has still some open problems: definition of user constraints, dynamic graphs, node sizes. In order to solve these problems it will be probably necessary to improve the layout algorithms and to build a powerful constraint manager.

References

[1] Salvador Pinto Abreu. *ALPES X-Prolog Programming Manual.* CRIA/UNINOVA, 1989.

[2] E. R. Gansner, S. C. North, and K. P. Vo. DAG: A program that draws directed graphs. *Software - Practice and Experience*, 18(11), November 1988.

[3] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[4] Sven Moen. Drawing dynamic trees. *IEEE Software*, pages 21–28, July 1990.

[5] J. Moura Pires, H. Pinheiro Pita, and L. Camarinha Matos. Specification of a platform for the development of graphical user interfaces. Technical report, DI/UNL, July 1991.

[6] Edward M. Reingold and John S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, March 1981.

[7] Lawrence A. Rowe, Michael Davis, Eli Messinger, Carl Meyer, Charles Spirakis, and Allen Tuan. A browser for directed graphs. *Software - Practice and Experience*, 17(1):61–76, January 1987.

[8] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(2):109–125, February 1981.

[9] Michel Wermelinger. *GET: Graph Editor and Tools—The Incomplete Reference.* CRIA/UNINOVA, January 1991.

[1] Widget $\leftrightarrow$ Window Gadget, the X Toolkit's abstraction for graphical objects