A Demonstration of Interactive Graph Based Visual Language Applications

Jeffrey D. McWhirter

Department of Computer Science, University of Colorado, Boulder, CO

Abstract. Escalante is a development environment for applications that use the pictorial representations of graphs to facilitate human-computer communication. Escalante supports a wide range of graph representations and layouts through a localized spatial constraint mechanism as well as external graph layout tools.

1 Introduction

Escalante [2] is a development environment that supports the rapid construction of interactive applications for a wide range of graph based visual languages. The domain of Escalante is characterized as graphs based on the underlying content of the visual languages within the domain, not on any particular pictorial representation of those languages. Graph constructs are generalized as *entities* and *relations*. A further generalization is made by regarding these constructs as the *elements* of the graph. An entity represents a thing within a graph (e.g., node, graph, subgraph, aggregation, port). A relation concretely defines some relationship (e.g., edge, member of graph, containment) between two elements, termed the *tail* and *head*.

Escalante provides a localized spatial constraint mechanism that is useful in declaratively defining layouts for regularly structured graphs. A relation in a graph may contain any number of spatial constraints. A spatial constraint defines that a point on the target of the constraint is equal to, less than or greater than a point on the source of the constraint. The target and source pair can be any of the six pairs formed by the relation, its tail or head (e.g, {relation,tail}, {tail,head}). For example, the containment of the head of a relation by its tail is defined by constraining the upper left corner of the tail to be above and left of the upper left corner of the head (and likewise constraining the lower right corner).

Escalante supports the use of external layout tools such as *dot*[1] to perform more generalized graph layout than what can be defined with the localized spatial constraint mechanism. As shown in Fig.1, the user of an application can interactively invoke commands that write out the graph as a whole or an arbitrary subset of the graph. The external tool is invoked on this file with the result graph written to a file that Escalante then reads to layout the graph. All applications constructed with Escalante provide this facility. Currently, only the dot layout tool is used but there are no barriers within Escalante to incorporating other layout tools.

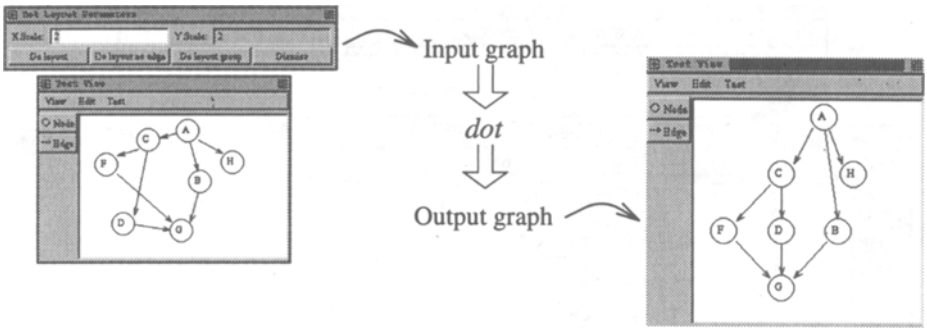

Fig. 1. Using External Layout Tools

Using Escalante, one can construct applications that consist of two or more distinct, yet related, graphs. Graphs are related to one another through an underlying non-visual graph. Each element (i.e, entity or relation) within a graph may be related to an element in the underlying graph. Each element in the underlying graph may be related to elements in one or more of the displayed graphs. Changes to the displayed graphs are propagated to the underlying graph which in turn propagates the change to the other displayed graphs. This functionality allows for the simultaneous creation, manipulation and display of graphs that share the same basic structure but may vary greatly in their representation. It is thought that the ability of a user to understand and manipulate graphs can be augmented by this multiple graph capability.

2 Example Applications

2.1 The MView Application

MView, shown in Fig.2, consists of six distinct, yet related graphs. Changes to one graph (e.g., element or relation addition) are reflected in each of the other graphs. The user can interact with any of the graphs, creating new windows or copying a graph. Much of the layout of the graphs was defined using the localized spatial constraint mechanism.

2.2 Nested Table

The NestedTable application, shown in Fig.3, allows for the interactive creation of tables. The user can dynamically add, copy, delete, resize and reposition columns, rows and table entries. In this application the underlying graph (e.g, column element, row element, table entry, next column relation) and the localized constraint mechanisms are used as a construction mechanism. This differs from the tabular display of graph based information (e.g., the MView example).

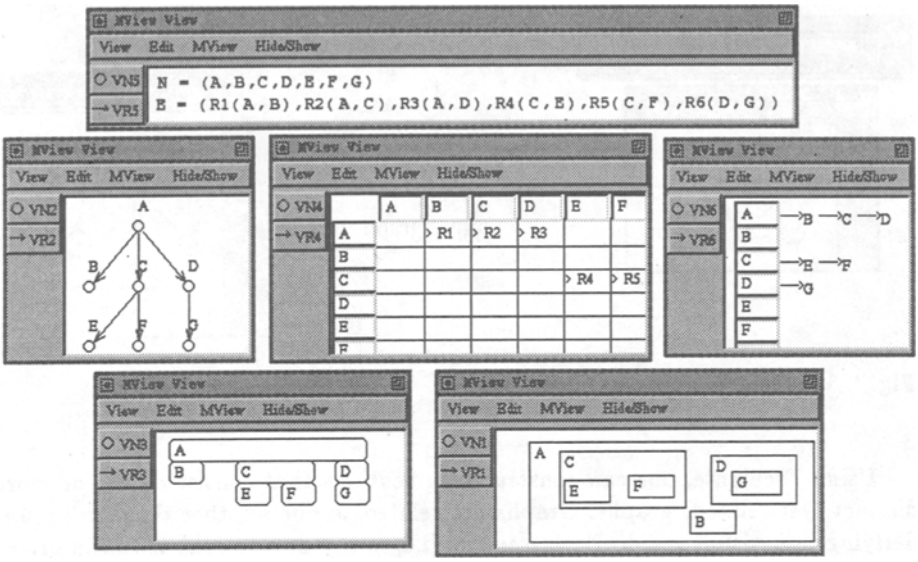

Fig. 2. The MView Application

2.3 MessageView

MessageView, shown in Fig.4, is a trace driven visualization tool that displays the message passing behavior in a parallel computation. The underlying graph consists of a set of *Process* elements (horizontal lines) laid out successively vertically. The *Header* element, shown at the top of the figure, displays the set of time intervals. Message passing between processes is represented with *Call* relations between the Process elements. Various spatial constraints are used to layout the Process elements, Call relations, and to display the vertical time intervals.

3 Conclusion

The applications constructed using Escalante exhibit a wide range of pictorial representations of graphs, one is not limited to simple "circles and arrows" representations. Escalante demonstrates that the possible representations of graph based information is quite broad, including traditional node/edge representations, topological arrangements, tabular displays and textual lists. Given regular structure within a graph the localized spatial constraint mechanism within Escalante can be used to declaratively define a wide range of graph layouts.

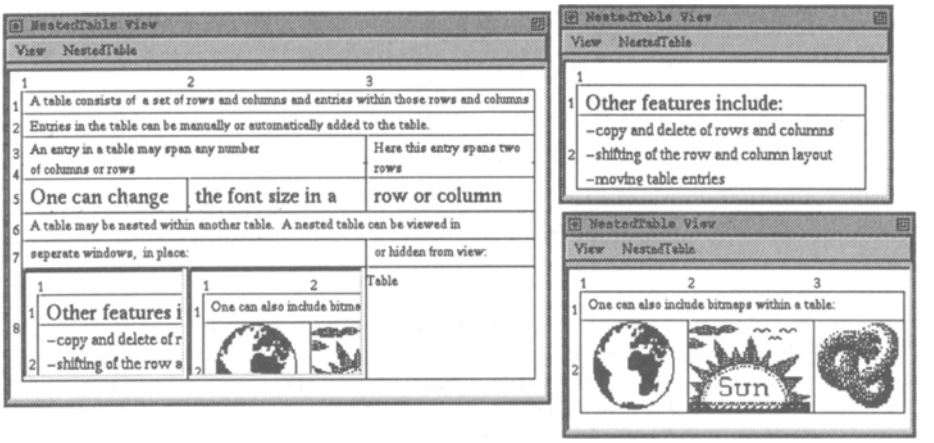

Fig. 3. The NestedTable Application

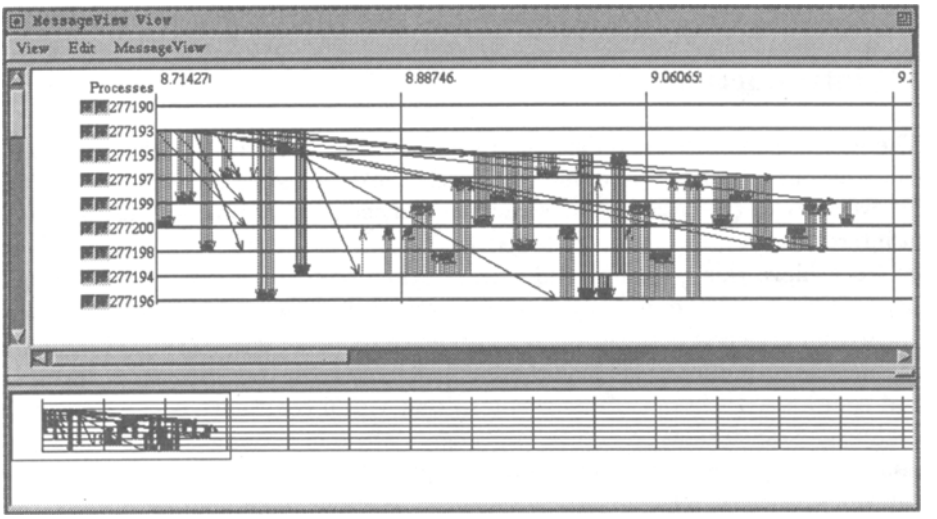

Fig. 4. MessageView

References

1. Eleftherios Koutsafias and Stephedn J. North. *Drawing Graphs with dot.* AT&T Bell Laboratories.

2. Jeffrey D. McWhirter and Gary J. Nutt. Escalante: An environment for the rapid construction of visual language applications. In *1994 IEEE/CS Symposium on Visual Languages (VL'94)*, 1994. To Appear.