

FIG. 1. *Interactions of GDR with its environment.*

is the other endpoint of  $e$ ).

- (iv) The graphical interface for GDR is written using simple X11 calls (no widgets or toolkits). In essence, GDR provides an additional X11 resource, a graph, that can be manipulated like other resources.

While GDR does not yet offer some of the sophisticated features of other graph editing systems, the advantages of those features may be gleaned by either using GDR with other tools or by developing custom enhancements to GDR for particular applications.

## 2. Design of GDR

The current implementation of GDR is a prototype for testing ideas that may be used in future implementations. Nonetheless it is quite usable even in its crude present form. GDR was originally intended as a simple tool to generate input for implementations of graph algorithms, but it has evolved substantially.

One way to view GDR is as an extendible graph editor. The interactions shown to the left of the dotted line in Figure 1 are analogous to those of a simple text editor with its environment: a file, created by GDR











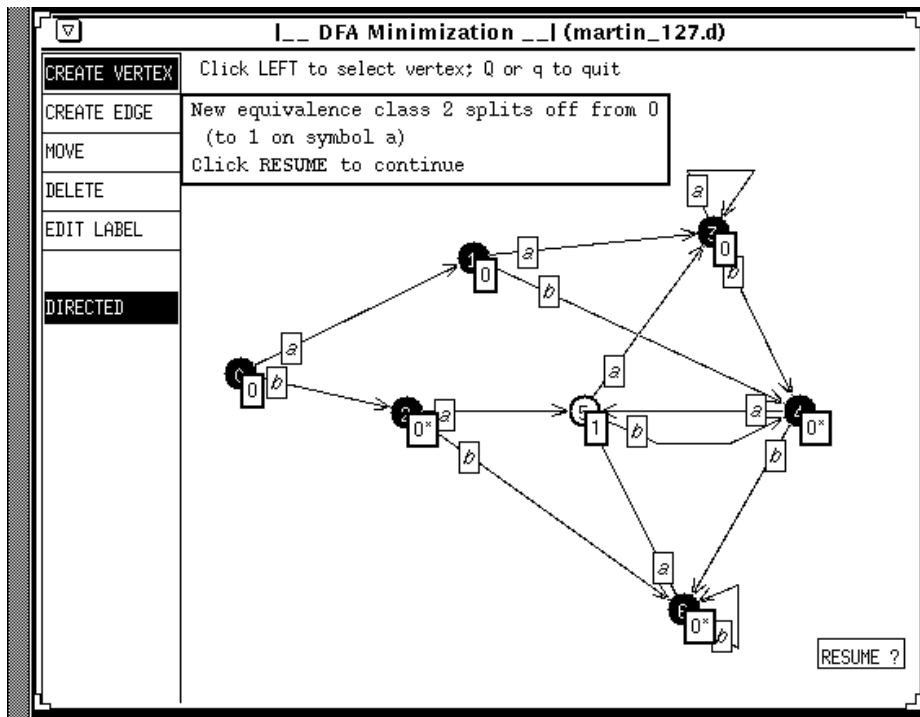


FIG. 3. DFA minimization in progress.

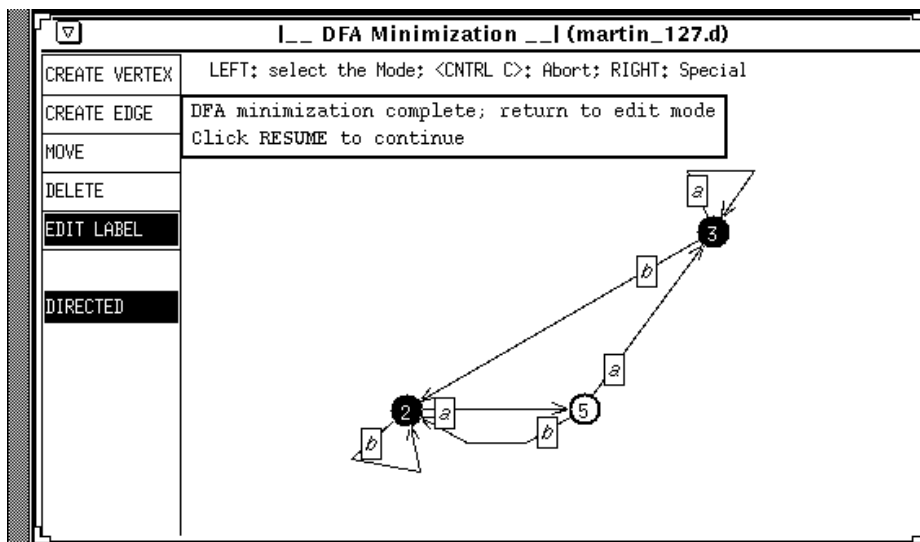


FIG. 4. DFA minimization completed.

use of GDR as a graph editor to create pictures of graphs for articles and reports. Possible enhancements fall into three categories (in ease-of-implementation order starting with easiest): new editing capabilities, new graph attributes, and automated drawing features.

Examples of new editing features are the ability to align vertices, options on size of vertices and thickness of edges (with defaults taken from a configuration file), use of splines for edges that are not straight lines, and the ability to copy and move user-specified subgraphs. New attributes include the use of multiple colors (or shadings) for vertices, edges, and labels and the use of more than two knots. Automated drawing features include such issues as better heuristics for the placement of labels and knots (the current implementation occasionally places labels on top of each other or creates a line segment for a new edge on top of an existing one), and automatic placement of new vertices. Tools that create aesthetically pleasing drawings from logical descriptions of graphs (see Di Battista et al. [5] for a survey) could interact with GDR via the GDR file mechanism described in the next subsection.

**4.3. GDR files.** GDR communicates with other programs via a simple ASCII file format that records all attributes of a graph. Procedures that access or modify GDR files can be written using the same object-oriented approach that is used for the internal representation of graphs (GDR itself, without its interactive part, could be used to write filter programs to modify GDR files).

Further development of the GDR file format and tools that manipulate GDR files is motivated by three considerations. First, GDR files can be used to capture a sequence of stills from the execution of an animation. The stills can either be viewed later or printed (although a GDR-to-PostScript translator has not yet been developed). Second, GDR can be used as an editor of graph drawings for documents. This would probably require the recording of additional information in GDR files as well as the development of “style files” to allow customization of such attributes as thickness of edges. Finally GDR files can serve as a convenient mechanism for compact storage of graphs and their drawings. This could spur the development of tools to modify the logical or physical structure of graphs stored in such files (analogous to the variety of Unix tools for text files).

**4.4. Object-oriented approach.** While tools written in languages other than C can interact with GDR via GDR files, direct interaction with the internal representation of a graph in GDR is limited to C programs compiled and linked with GDR. This choice was prompted by a need to develop a working prototype quickly. The original vision, GDR as an

independent process that interacts with other programs via messages, requires further study. The advantages of such a “pure” object-oriented approach are clear: animations could be coded in any language as long as the appropriate message protocol was used, GDR could interact with several programs simultaneously, or one program could interact with several GDR displays (for example, to animate actions on a graph and an auxiliary data structure at the same time). The primary disadvantage is the difficulty of implementing a communication scheme among independent processes (while still retaining the portability of GDR).

**4.5. Portability.** The current implementation of GDR, because of its use of ordinary C and plain X-Windows, can easily be ported to most Unix-based workstations (so far we have only tested this for DEC and Sun workstations). A feature of GDR that suggests portability to graphical user interfaces other than X-Windows is the simplicity of the routines that access and modify the display of the graph. It is likely that these access routines could be rewritten for adaptation to, for example, a Macintosh environment. Some rearrangement of the current code would be desirable to isolate the parts that interact directly with X-Windows. This would entail defining a higher-level portable user interface for simple graphical interactions.

## 5. Conclusion

We have presented an object-oriented graph editing and animation tool and illustrated both its present capabilities and future possibilities. A variety of algorithm animations have been implemented for classroom use, and GDR has been used as a research tool. GDR’s interfaces with the user, the programmer, and with other tools have proven to be flexible and easy to use.

Throughout this overview we have emphasized GDR’s properties as a tool rather than a system. The overall vision we wish to articulate is analogous to that of the software tools for text processing developed by Kernighan and Plauger [8]. As a general-purpose text editor is central to any collection of text-processing tools, so the graph editing capabilities of GDR allow it to play an important role in the development of graph-processing tools. In graph processing, there is the additional need to visualize operations, and GDR’s animation capabilities serve as a testing, debugging, and interactive viewing facility for other tools.

## REFERENCES

- [1] J. Abello, S. Sudarsky, J. Waller, and T. Veatch. AGE: An animated graph environment. In *Proc. DIMACS Workshop on Computational Support for Discrete Mathematics*. American Mathematical Society, 1993.
- [2] B. Birgisson and G. E. Shannon. GraphView: An extensible interactive platform for manipulating and displaying graphs. Technical Report 295, Computer Science Department, Indiana University, December 1989.
- [3] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics-based tool for the verification of finite-state systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.
- [4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [5] G. Di Battista, P. Eades, and R. Tamassia. Algorithms for drawing graphs: An annotated bibliography. Available by anonymous ftp from `wilma.cs.brown.edu`: files `/pub/gdbiblio.tex.Z` and `/pub/gdbiblio.ps.Z`, March 1993.
- [6] J. Ebert. A versatile data structure for edge-oriented graph algorithms. *Communications of the ACM*, 30(6):513 – 519, 1987.
- [7] J. E. Hopcroft. An  $n \log n$  algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.
- [8] B. W. Kernighan and P. J. Plauger. *Software Tools*. Addison-Wesley, 1976.
- [9] M. S. Krishnamoorthy, A. Suess, M. Ongheana, F. Oxaal, and T. Spencer. Improvements to GraphPack: A system to manipulate graphs and digraphs. In *Proc. DIMACS Workshop on Computational Support for Discrete Mathematics*. American Mathematical Society, 1993.
- [10] V. J. Leung, M. B. Dillencourt, and A. L. Bliss. GraphTool: A tool for interactive design and manipulation of graphs and graph algorithms. In *Proc. DIMACS Workshop on Computational Support for Discrete Mathematics*. American Mathematical Society, 1993.
- [11] J. Malhotra, S.A. Smolka, A. Giacalone, and R. Shapiro. Winston: A tool for hierarchical design and simulation of concurrent systems. In *Proceedings of the Workshop on Specification and Verification of Concurrent Systems*, Stirling, Scotland, 1988.
- [12] J. C. Martin. *Introduction to Languages and the Theory of Computation*. McGraw-Hill, Inc., 1991.
- [13] V. Roy and R. de Simone. AUTO/Autograph. In *Computer-Aided Verification '90*, pages 477–491, 1990.
- [14] M. Stallmann, R. Cleaveland, and P. Hebbbar. GDR: A visualization tool for graph algorithms. Technical Report 91-27, Department of Computer Science, North Carolina State University, Raleigh NC 27695-8206, October 1991.
- [15] Vikas Trehan. VTIIEW: A graphical editor for hierarchical networks of finite-state processes. Master's thesis, Dept. of Computer Science, North Carolina State University, 1992.

DEPARTMENT OF COMPUTER SCIENCE, NORTH CAROLINA STATE UNIVERSITY, RALEIGH,  
NORTH CAROLINA 27695-8206

*E-mail address:* `matt@euler.csc.ncsu.edu` or `Matt.Stallmann@ncsu.edu`