

SUI: A System Independent User Interface for an Integrated Scientific Computing Environment

Yaser Doleh and Paul S. Wang¹
Department of Mathematics and Computer Science
Kent State University
Kent, Ohio 44242-0001

ABSTRACT

The design and implementation of a Scientific User Interface is presented. Written in the C language, SUI is a window-menu-mouse oriented graphical user interface that is designed to provide a modern and integrated computing environment for scientific work. SUI can serve multiple client systems in parallel including symbolic, numeric, graphics and document formatting systems. SUI achieves hardware and operating system independence as well as network transparency by employing the X11 protocols and achieves client system independence by defining a client-SUI protocol that is simple and effective. Features of SUI includes input editing, history, 2-D mathematical expression display, interactive selection of subexpressions, interactive display and manipulation of 2-D and 3-D plots of mathematical functions, cut and paste with syntax translation, command templates, incremental 2-D display of mathematical input, and interactive configuration. A prototype system demonstration is planned for ISSAC'90

1 Introduction and Background

A well designed user interface not only makes a computing system more appealing and convenient to use but also increases the computation power provided to the user. The availability of modern workstations with high-resolution graphics displays has revolutionized user interface design for many computer systems. Scientific systems, whether numeric or symbolic in nature, are no exceptions. The *Matlab* system provides a user interface more convenient than most numerical packages. Research on better user interfaces for symbolic computation systems has been active as well. Work on the Maple [4] user interface [9]

implemented at University of Waterloo, GI/S [19] for Vaxima [12] at Kent State, and MATHSCRIBE [14] for Reduce [7] at Tektronix and Mathematica front end on the Macintosh are examples. These systems have introduced many important ideas and features to the design of user interfaces for scientific computation. Among them are the use of multiple windows; the graphical display of mathematical symbols, formulas, curves and surfaces; convenient mouse-menu interaction with the user; interactive selection of subexpressions; and the separation of the user interface from the compute engine that it controls.

Drawing on experiences gained through these earlier systems, we embarked on the design and implementation of a new user interface system called SUI. Initial design work started in the latter part of 1988. SUI is a window-mouse oriented graphical user interface system intended to control/integrate computation systems used in engineering and scientific work including symbolic systems, numeric packages, graphics and document formatting. A standard protocol is used by SUI to interact with the various *client* systems that it controls. Hardware and operating system independence is achieved through the X11 protocol. The main features of SUI are discussed in section two.

An initial version of SUI was taken to the ISSAC'89 conference as a demo to collect reactions and suggestions from a wider audience. The resulting prototype system represents a collection of features we felt appropriate and important, subject to compromises between user convenience and system performance, as well as the available manpower for implementation. A demonstration of SUI is planned for ISSAC'90. ¹

2 Principal Features of SUI

SUI enables a user to invoke different computation systems to run concurrently either on the local workstation or on powerful remote hosts. It allows convenient and useful interactions among these systems. For example, a symbolic

¹Work reported herein has been supported in part by the National Science Foundation under Grants CCR-8714836 and EET-8714628

manipulator, such as Vaxima, can produce formulas that can be included directly in a document and viewed on the screen before a hard copy is produced. Results derived in a symbolic system can be passed to a code generator through SUI to produce ready to compile numerical code. A symbolically derived function can be passed to a numerical package for repeated evaluation. A set of points produced by one client representing a curve or surface of a mathematical function derived by another client can be graphically displayed and interactively manipulated via another graphics engine. The resulting graphics can be included in a document. Forgot exactly how to use that command? Just type part of the command and activate SUI supported client specific *command template* processing to help. Want to transfer information from one client to another? Just use SUI cut-and-paste with inter-client translation. SUI also defines a protocol for inter-client requests that allow clients to take advantage of one another's computational capabilities.

A good user interface is a substantial piece of software that requires significant efforts for its design and implementation. Thus, it stands to reason that the design of a user interface should be totally independent of the underlying compute engines, called *client systems*, that it controls. This will allow us to easily adapt the user interface to work with a number of existing systems, and to allow future compute engines to take advantage of SUI without difficulty. To achieve *client system independence*, the user interface must define a protocol for information exchange with a client compute engine. The evolving protocol that SUI uses will be discussed in detail.

In order for SUI to work on a wide variety of computer systems, the user interface must be easily portable to many popular workstations/computers running under different operating systems. This *host system independence* is achieved through the standard X11 window system. Thus, SUI is written in C for speed, portability and ease of interface to X11.

Another major feature of SUI is that it provides a scientific computing environment in which symbolic, numeric, graphic, code generating, document formatting, and other systems used for scientific computing are integrated. It is also an experiment to identify the desirable user interactions in such an integrated environment. Other major features of SUI are as follows.

1. SUI supports a set of well selected features for the graphical display and interactive manipulation of mathematical symbols and formulas on a bit-mapped display. Major functionalities include interactive selection of parts of expressions, cut-and-paste, and interactive expression perusal.
2. User input is handled nicely with features like visual input editing, cut-and-paste from output to input, client-definable command templates, history facility,

and incremental 2-dimensional display of mathematical input.

3. SUI supports network-based operations and can initiate/control clients running on remote hosts.
4. SUI has an associated graphics facility, XGP, for the display and interactive manipulation of 2-D and 3-D graphs of mathematical functions. XGP evolved from the SIG system [18] and works as a client of SUI.
5. For client systems, SUI provides facilities for reporting system status, setting options, displaying error/diagnostic messages, interactive help, and consulting on-line documentation.
6. SUI is customizable and extensible.

3 SUI Architecture

3.1 Overview of X11

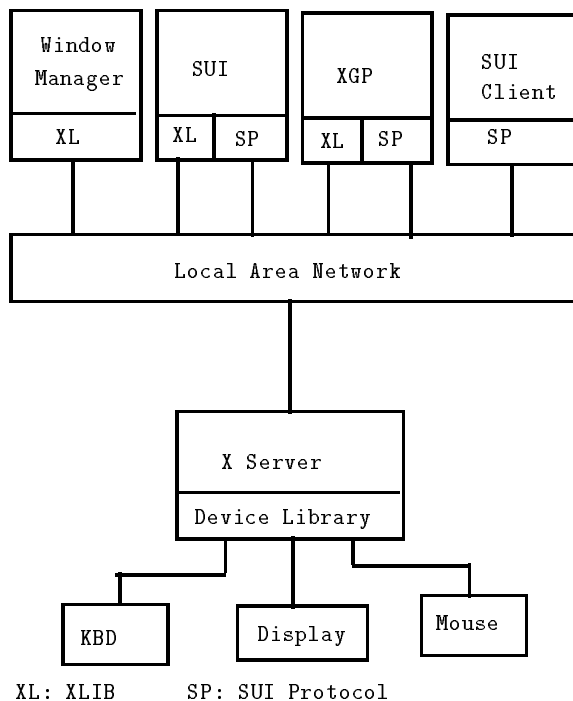


Figure 1: SUI and X System Architecture

The X Window System [6] provides software facilities to support a window-mouse oriented user interface environment. X11, the latest version of X, is supported by a consortium of hardware and software vendors who have made a commitment to use this standard across their product lines. The X11 software is in the public domain and is

distributed freely by MIT and also redistributed by others. The basic X11 consists of two major components: the X *server* and the *Xlib*. On top of the basic X11, there are also easy to use *toolkits* [11], [15]. The X server uses device dependent drivers that handle I/O for one or a set of graphics *displays* (keyboard, mouse and CRT display) being controlled by the server. An X server performs actual I/O on behalf of *client* processes which make requests to the server to perform desired I/O operations. The X server defines a standard protocol for I/O requests. Any program that operates a graphics terminal through the X server is a client. And the X sever can simultaneously take care of multiple clients and displays. The X client is always device independent and will work with any graphics equipment that comes with an X server. The distributed X11 package already contains servers for most popular graphics workstations and displays. The client program may run on the same graphics workstation as the server or on another computer, in which case it can communicate with the server through a network.

The Xlib is a library of C functions that can be used by a client program to interact with the X server. Xlib provides, for example, a function to initiate connection with a particular X server, a function to establish a window of a certain size and type, and a function to draw a line or a polygon in a graphics window. Once a client program is compiled and linked with the Xlib, it will work without change with any X server transparently across the network. X also supports multiple networking protocols including TCP/IP [16], DECnet and Chaos. Figure 1 illustrates the way SUI and its clients fit into the X architecture.

3.2 SUI Organization and Interfaces

The entire SUI package is organized into three major parts: a graphics terminal part that is X11 dependent, a scientific interface part that supplies the symbolic mathematical expertise and the integrating environment, and a client interface part that communicates with SUI clients using a protocol defined by SUI.

SUI sits between the user and the clients (Figure 2). It receives keyboard and mouse input from the user to control the clients. On the other hand, SUI responds to I/O and other requests from the clients and performs the desired actions through the X11 window system.

4 Using SUI

When SUI is initially invoked, a control panel appears that contains *buttons* to select actions or to initiate activities managed by SUI. Each of these buttons selects a different second level menu. There is a button for each type of client that SUI can control and there is one for SUI itself.

By selecting the SUI button, the user can control, setup and customize SUI. This customization and setup involves

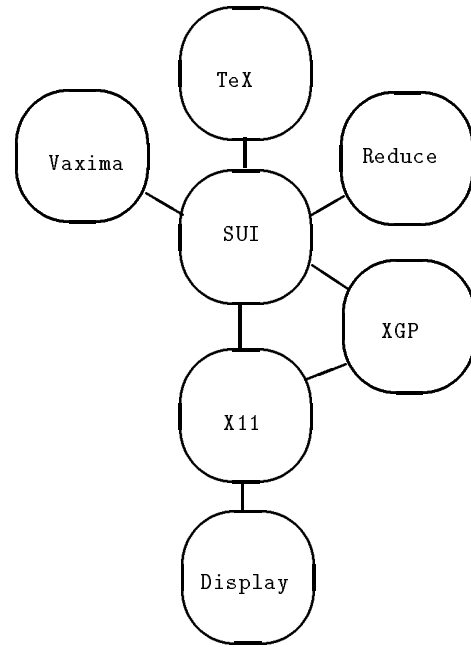


Figure 2: SUI Interfaces

several commands. A user may ADD a client through a dialogue box to fill in the necessary information in order to setup the configuration file for that particular client. Other menu choices allow a user to DELETE and CHANGE the configuration file of any client.

To start a client system is just as simple. Selecting a specific client button offers such options as: START, SAVE CONFIGURATION, CHANGE STATUS, SEND SIGNAL, SAVE YOURSELF and EXIT. Once a client is established, another instance of that same client cannot be started. If another instance of the same client is needed, a user should have another entry in SUI for it.

If START is selected, and the client is invoked successfully, windows and control panels for the desired client appear. The configurations of windows and/or control panels depend on the client requirements and user preferences. The information is kept in a per client configuration file. For example, the information may specify input, output, help and error message windows to work with the client program. The configuration file is generated when a new client is introduced into SUI.

To terminate a running client, the user may issue a particular exit command directly to that client from its *input window* or, alternatively, may select the EXIT button on the client control menu.

SUI provides tools for easy customization of client window layout. The user may choose the window types, sizes, positions, names, colors and other attributes interactively until the screen looks just right. Then the configuration

can be saved by simply clicking a mouse button. SUI will use the saved configuration every time until it is modified again. After a window is established, it can also be modified dynamically.

5 The Input Windows

Input to a client program is entered in an *input window* supplied by SUI when the client is started. Input windows for different clients may of course exist at the same time. Keyboard input goes to the *current input window*, which is selected using the mouse. Input typed is basically passed directly to the client program. There is no checking of client-defined syntax on the part of SUI.

The basic input window provides a simple **emacs** or **vi** style visual editor, depending on the user's preference. The editor buffer contains all the text ever entered in that particular input window. The editor is convenient for correcting typing errors and modifying one or more previous command to be reentered. The edited buffer can also be saved in a file for later use.

If the client keeps a history of the received commands, there is an optional window which can be used to display the history. The client may choose to use the SUI protocol to employ this window for the display of history resulting in a generalized *redo* facility.

Helping the user enter commands is important. To this end, SUI supports *command templates* that guide the user in entering commands and arguments correctly. SUI maintains a table of command names and their arguments for each type of client. This table can be loaded from a file, transferred from the client at any time, or augmented by the user interactively to support user-defined functions. To use the command templates, the user simply types a command (or a prefix of it) followed by a *hot key* (default '?'). This action causes *command template processing* and the command constructed will be transmitted to the client. If the given prefix matches more than one command, all matches are displayed. Otherwise the selected command is used to guide the entering of the arguments. A similar facility helps the user examine and set client defined flags, switches and options.

Mathematical expressions entered in infix notation can be displayed incrementally (in real time) in two dimensional text book fashion. This helps greatly the input of mathematical formulas.

It is important that keyboard interrupts accepted by a client system be processed correctly by SUI. For example, a Control-C causes Vaxima to abort its current computation and to go into a “break loop”. Therefore, the configuration file also contains information of keyboard interrupt characters and their functions. An interrupt character detected by SUI causes an appropriate *out of band* message to be sent by an SUI daemon on the target host to the client. Other special characters may be substituted by predefined characters and sent to the client directly. This character-action mechanism works in general and is not restricted to handling interrupts.

6 The Output Windows

SUI provides multiple output window types. The two most important types are the text display window and the 2-D mathematics display window. The former is simply a terminal emulation window supported by X11. The latter is used to display mathematical formulas and to allow interactive selection of expressions using the mouse. In addition there are the client *help window*, the client *choice menu*, and, of course, the *interactive graphics* window. Within the mathematics display window, each expression returned by the client is displayed in a separate subwindow. Each subwindow contains only one expression with its optional label. If the expression does not fit in the subwindow, horizontal and/or vertical scrollbars will appear for the user to view the rest of the expression. The size of each subwindow can also be enlarged to take in more of a large expression. At any time, one expression is the *current expression*, which is always the last expression displayed. It can, however, be temporarily set to any other expression using the mouse. The subwindow containing the current expression is clearly distinguishable from the other subwindows. Operations such as scrolling and subexpression selection are always performed on the current expression (CE).

In the mathematics window there are a variety of facilities to help the user manipulate and understand mathematical expressions. The following operations are supported.

- **Cross copy:** Cut and Paste is an important way to interface systems under SUI. Copying of text strings is supported by X and readily available to an SUI user. In addition, the CE or a subexpression can be copied into certain input windows. This operation may involve interactive selection of the subexpression and transformation into the correct input syntax for the target input window. For example, Vaxima output can be cut and pasted in the Vaxima, Reduce \TeX ,

GENTRAN, or GENCRAY input window. The expression is run through a converter which transforms SUI internal representation to the correct input syntax. The conversion is only limited by the type of converters programmed.

- **Zoom:** Going into and out of component parts of the CE.
- **Mouse apply:** An interactively selected subexpression of the CE is replaced by the result obtained by applying a user supplied command to the subexpression. The result is displayed as a new expression. The mouse apply operation was first described in GI/S.
- **Substitution:** All occurrences of an interactively selected subexpression in the CE are replaced by a variable or value supplied by the user. The variable is automatically assigned the subexpression as its value. The result is displayed as a new expression.
- **Expression skeleton:** The CE can be displayed in more or less detail to help the user digest large expressions. For example, a skeleton expression may be shown as a sum of 16 terms without displaying the details of any of the terms. Any subexpression may be examined by expanding it or by making it the *current expression* using the *zoom* function. Individual terms can be accessed also by their position in the expression. A user can, for example, make term 1310 the CE, select term 105 of a sum or scroll the window to the last term. All these actions are bound to certain keys that can be changed by the user.

7 The Graphics Windows

SUI has an associated interactive graphics package, XGP, which is developed as part of the master’s thesis work of D. Bennett [3] at KSU. XGP evolved from SIG [18] and Hsu’s master thesis work [8]. XGP is written specifically to interface to SUI and plots 2D/3D graphics in a graphics window that allows interactive manipulation of the graphics displayed. Any graphics requests received by SUI from a client are directed to XGP for processing. The language used between SUI and XGP to control graphing is a subset of the language defined for SUI-client communication.

Figure 4 shows the interactions that take place when a client requests graphics display from SUI. Part of the SUI-client protocol has to do with graphics requests such as establishing a graphics window, display of 2D/3D graphics, axes, labels, and so forth. The graphics window is an *object* that can handle many interactive graphing requests from the user including rotation, zoom, overlay etc. with great speed.

A graphics engine for mathematical functions runs as an SUI client. It can perform many computations for plotting functions including finding extreme points, limit points

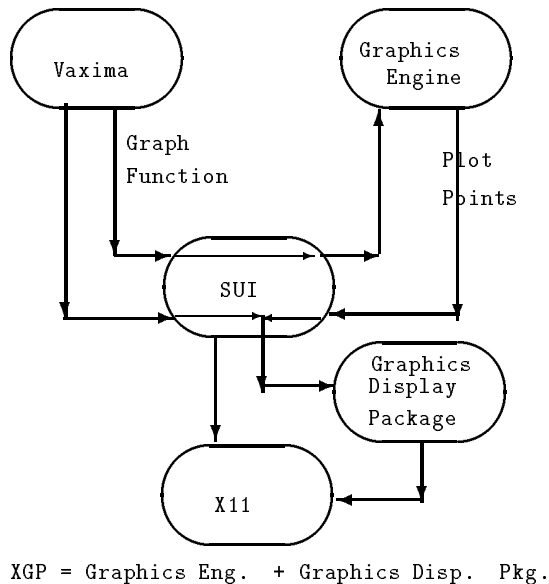


Figure 3: XGP-SUI Relationship

(points where the function goes to infinity), and evaluating the function in the given plot interval. All these are done by generic requests to SUI that are then taken care of by an appropriate SUI client.

8 Interfacing A Client System to SUI

One of the design goals for SUI is to provide the ability to interface with existing compute engines as well as future clients that are written to take advantage of SUI. SUI interfaces with existing programs by replacing their own user interfaces. The modification to existing client systems involve reducing output handling to simply sending the appropriate display requests to SUI and reading user input from SUI rather than the keyboard. For example, An interface to Vaxima has been produced by replacing its entire output display package with a single Lisp function and making a few modifications in the Vaxima input function.

8.1 The SUI-client Protocol

One important aspect of this research is to define a compute-engine-to-user-interface protocol that is both simple and effective for our purposes. It is realized that only through sufficient experimentation and usage experience

can such a protocol evolve into any standard form. Nevertheless, we shall describe what is currently being used for SUI-client communication. A programmer who writes SUI client programs needs to become familiar with this protocol.

Messages between SUI and a client are sequences of characters sent through TCP/IP stream socket connections. Protocol processing of incoming messages to SUI are handled in two levels: a low level which receives and analyzes the requests lexically. Individual requests thus identified are then sent to the next level sequentially for processing. Outgoing messages to clients are processed by the client's interface module to SUI. Conventions are used to identify the origin of the client input: from the input window, from a dialogue box, or an option menu, for example.

The protocol defines the following categories of requests:

1. Display Requests (sending information to the user):
 - (a) Material to be displayed in an output window
 - (b) Prompts for input from the user
 - (c) Pop-up menus for user selection
 - (d) Dialogue boxes for user input
2. Requests to another SUI client. In this case, SUI does not change the data, it simply acts as a pipe between the two clients.
3. Requests to SUI (not displayed):
 - (a) Operations on input/output windows. For example, OPEN, CLOSE etc.
 - (b) Create internal structures for menus or dialogue boxes.
 - (c) Change the client configuration file.
 - (d) Change the client status.
4. Requests for generic services. SUI keeps a table of certain generic capabilities of clients that may be of use to other clients of SUI. This is to provide the possibility of a client sending requests to SUI to access services provided by others. For example, XGP, as a mathematical function graphing system may request the derivative of a function $f(x)$ or the roots of a polynomial from SUI. SUI will check its internal tables and send an appropriate request to a computer algebra system or a numerical system, collecting the results and return the answer to the requesting graphical system. We expect to provide this service only for a few well defined requests. This also allows us to study the its general utility in a wider range of cases.

To interface a client to SUI, it must have a front end that communicates with SUI using the protocol. In addition, the programmer writing the interface must define all special control characters with a list of functions.

9 SUI Implementation

SUI consist of four major parts:

- Network layer: this provides the connection to clients. It sets up the socket connection to receive and send data. It understands the lower level of the client-SUI protocol.
- Kernel: this is the heart of SUI. It is responsible for directing traffic between the clients and the user. It includes all the functionalities that are hidden from the user.
- X interface: this part talks to the user through the X window system. One may modify this module to make SUI work with another window system.
- Configuration and initialization files.

An SUI *network daemon* process is used to start clients on remote hosts. The SUI network layer connects to this daemon and sends the necessary information to start and terminate the client. This daemon is also used to send interrupt signals to the client program.

The windows are implemented as *widgets*, window graphics objects, using the X toolkit and Athena widgets from M.I.T. [15]. One example is the SUI *math widget*, a self contained object with functions for equation manipulation. The object-oriented approach provides an avenue for future expansion. One may write new widgets or change existing widgets without changing other parts of SUI.

10 Discussion

The Scientific User Interface is being developed to help provide an integrated computing environment for scientists and engineers. The most important systems to integrate include symbolic mathematical systems, numerical computation systems, graphing systems for mathematical functions, document preparation/formatting systems, and program development tools: text editors, compilers and debuggers.

The purpose is to make the individual and combined use of such systems much easier for scientists and engineers regardless of their computer expertise. The implementation of SUI will continue to evolve to support additional client types and new useful interactions in this integrated environment.

In addition to the features described here, SUI will also provide an *interface library*. This library will allow programmers to interface clients easily with SUI.

Even though SUI has been developed to control multiple clients it can still be used as a nice and efficient user interface for a single, stand-alone compute engine. SUI has the potential of providing a standard user interface that would be the same for many different technical applications.

References

- [1] S. Kamal Abdali, Guy W. Cherry, Neil Soiffer. "Spreadsheet Computations in Computer Algebra," Technical Report No. CR-87-14, Computer research Lab, Tektronix, Inc. Beaverton, Oregon. February 19, 1987.
- [2] Dennis Arnon, et al. "Caminoreal: An Interactive Mathematical Notebook," Proceedings of EP 1988.
- [3] Dan Bennett, "XGP: An X-based Interactive Graphics Package for Mathematical Functions," Master's thesis in progress, Department of Mathematics and Computer Science, Kent State University, Kent, Ohio, USA.
- [4] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, M. B. Monagan and Steven M. Watt, "*Maple Reference Manual, 5th Edition*, WATCOM Publications Limited, Waterloo, Ontario, Canada, 1988.
- [5] James H. Davenport and C. E. Roth, "PowerMath - A system for the MacIntosh," Conference Proceedings of Symsac 86, pp13-23. 1986
- [6] Jim Gettys, Ron Newman, and Robert W. Scheiffler "*Xlib - C Language X Interface*" X Window System, Protocol Version 11, Release 3, May 1988.
- [7] Anthony C. Hearn, ed. *Reduce User's Manual*, Version 3.0, The Rand Corporation, Santa Monica, California. April 1983.
- [8] Chia-Kai Hsu, "An Object-oriented Interactive Graphics Package Based on the X Window System", Master thesis, Department of Mathematical Sciences, Kent State University, Kent Ohio, Dec. 1988.
- [9] B. L. Leong "Iris: Design of a User Interface Program for Symbolic Algebra," ACM 0-89791-199-7/86/0700-0001. Proceedings of Symsac. 1986
- [10] William A. Martin, "Computer Input/Output of Mathematical Expressions," Proc. of the Second symposium on Symbolic and Algebraic Manipulation. March, 1971.
- [11] Joel McCormack, Paul Asente, and Ralph R. Swick "*X Toolkit Intrinsic - C Language X Interface*" X Window System, X Version 11, Release 3, May 1988.
- [12] Richard Pavelle and Paul S. Wang, "MACSYMA from F to G", Journal of Symbolic Computation, vol. 1, 1985, pp. 69-100, Academic Press.
- [13] Tekla S. Perry and John Voelcker, "Of mice and menus: designing the user-friendly interface," IEEE Spectrum pp 46-51. September 1989.
- [14] Carolyn J. Smith and Neil M. Soiffer "MathScribe: A User Interface for Computer Algebra Systems," ACM 0-89791-199-7/86/0700-0007, Conference Proceedings of Symsac 86. 1986.
- [15] Ralph R. Swick and Terry Weissman "*X Toolkit Athena Widgets - C Language X Interface*" X Version 11, Release 3, May 1988.
- [16] Paul S. Wang, *Introduction to Berkeley Unix*, Wadsworth Publishing Company, Belmont, California, USA, 1988.
- [17] Paul S. Wang, "Integrating Symbolic, numeric, and graphics computing techniques", *Mathematical Aspects of Scientific Software*, The IMA Volumes in Mathematics and Its Applications, vol. 14, Springer-Verlag, 1988, pp. 197-208.

- [18] Paul S. Wang, "A System Independent Graphing Package for Mathematical Functions," Proceedings, International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO'90), April 10-12, Capri Italy.
- [19] Douglas A. Young and Paul S. Wang, "GI/S: A Graphical User Interface For Symbolic Computation Systems," Journal of Symbolic Computation, Academic Press, Jan. 1988, pp. 365-380.
- [20] Alain Zarli and Delia Balbontin. "Two approaches for the graphical edition of mathematical formulas: attributed grammars and object-oriented languages," ISAAC 1988.