CAMINOREAL: AN INTERACTIVE MATHEMATICAL NOTEBOOK

DENNIS ARNON, RICHARD BEACH, KEVIN MCISAAC, and CARL WALDSPURGER

Xerox Palo Alto Research Center

ABSTRACT
Four broad categories of mathematical software are numerical computation, mathematical typesetting, computer algebra (symbolic mathematics), and "technical electronic mail" (mail that contains formatted mathematical expressions). Powerful and sophisticated systems are currently available in each of these categories. Simultaneous, integrated access to all four types of functionality is not yet realized, however. CaminoReal is a working system that addresses this need. It is part of Cedar, the programming environment of Xerox PARC's Computer Science Laboratory, and is used in conjunction with Tioga, Cedar's multimedia document editor. Printing and management of other document components, such as text, graphics, and voice, are provided by Tioga. For computation, CaminoReal offers a built-in algebra package based on the notions of objects and domains, plus access to "algebra servers" on a network. Mathematical expressions are exchanged among CaminoReal, Tioga, and these algebra servers in abstract syntax. Our current servers are the Reduce, SMP, and SAC-2 computer algebra systems. This document has been produced using Tioga and CaminoReal.

There is no 'royal road' to geometry.
Euclid, said to Ptolemy I

1. Introduction

1.1. *Overview of CaminoReal*

CaminoReal is a system for direct manipulation of mathematical expressions, whether as part of a technical document or as inputs and outputs of computational engines. Its user interface offers interactive, syntax-directed, two-dimensional, WYSIWYG editing of mathematical expressions, placing or fetching such expressions in or from formatted documents, and two sorts of computational facilities: a built-in algebra package and well-known algebra systems such as Reduce [Hearn82], SMP [Wolfram85], and SAC-2 [Collins80]. The internal algebra package is based on an object-oriented paradigm that supports polymorphic procedures. For example, one can easily create and perform simple arithmetic on matrices of polynomials with complex number coefficients, or matrices of such matrices, etc.

CaminoReal has much in common with other recent work, but we believe it has two unique features as of this writing: the tight coupling of its computation facilities with a sophisticated document system, thereby opening interesting opportunities for computed and interactive documents (details in Sections 3, 4, and 5), and its access to computational "algebra servers" on our local network (details in Section 4). These capabilities have been facilitated by building it upon the rich base provided by Cedar, the programming environment of the Computer Science Laboratory at Xerox PARC [Swinehart86, Donahue86]. CaminoReal is closely linked to Tioga, Cedar's multimedia document editor, which provides facilities for printing and management of other document content types such as text, graphics [Pier88], and voice [Zellweger88a]. The screen, mouse actions, and keyboard input for CaminoReal and Tioga are managed by the Cedar viewers package (as for most Cedar tools). A viewer is a window that can be scrolled and resized, and which can have buttons and pop-up menus that invoke commands. The mouse is used to point and select text or expressions.

CaminoReal implements a recently proposed [Arnon87] standard architecture for "mathematical systems" that provides for interactive editing of mathematical notation, display of math in its traditional two-dimensional appearance, symbolic and numerical computation, and technical document production. We have found this architecture useful, and we summarize it in Section 2. Section 3 discusses Meddle, CaminoReal's interactive mathematics expression editor, and Tioga. Section 4 covers AlgebraStructures (CaminoReal's built-in algebra package) and algebra servers. In Section 4.2, we explain why associating domains with mathematical expressions can be a significant help in document creation, proofreading, and computation.

CaminoReal supports the creation of "interactive" technical documents. For example, the user can browse a typeset draft of a technical document on the workstation screen, select, edit, and compute with mathematical expressions in the document, and insert the resulting expressions back into the document. One can extend this paradigm to the notion of a "computed document," that is, a document with embedded computations. Two examples of computed documents are mathematical form letters and spreadsheets, which we discuss in Section 5. CaminoReal is one of several current experiments with active documents in Cedar [Zellweger88b].

1.2. *Previous work*

In the 27 years since J.C.R. Licklider presented the notion of man-computer symbiosis [Licklider60], there have been continuous efforts to define and build such "symbiotic systems" for mathematical work. Clapp and Kain's Magic Paper system [Clapp63], Minsky's MATHSCOPE proposal [Minsky63], and Martin's Symbolic Mathematical Laboratory [Martin67] appear to be the earliest. Numerous papers chronicle subsequent progress and offer pointers to the future [Griesmer71, Sundblad74, Ng79, Berman79, Hearn80, Hearn82, Allen81, Foster84, Engeler85]. There is much current work relevant to CaminoReal [Bloomberg87, MathSoft87, Smith86, Spirkovska86].

Following Licklider and most of the authors just cited, we do not in this paper consider the questions of "intelligent behavior," learning, or automated reasoning in mathematical systems. These are issues of compelling interest, but we believe that certain more mundane topics deserve priority at present. Some examples of topics we exclude are: "mentor" and problem-solving capabilities projected for MACSYMA [Martin71], English-speaking MACSYMA Advisor [Genesereth77], artificial intelligence aspects of mathematical systems design [Calmet87], tutoring abilities of a calculus environment [Suppes87], and theorem proving and heuristic reasoning functionality [Bundy86].

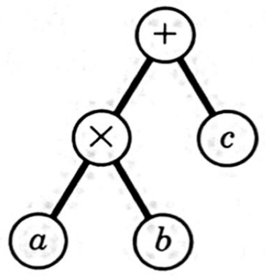
CaminoReal, like most similar current work, builds on certain hardware and software developments of recent years. In hardware, these are: (1) high-resolution screens and printers, (2) pointing devices, e.g., mice, (3) high-speed networks, and (4) personal workstations. Important software developments are: (1) the emergence of large collections of sophisticated and powerful symbolic and numerical algorithms, (2) the rapid diffusion of high-quality electronic publishing systems that utilize the hardware base just enumerated, and (3) the broad acceptance of certain user interface paradigms for interactive software, such as WYSIWYG editing, direct manipulation [Shneidermann83], menus, windows, holophrasting [Koved86], icons, and browsers. Certain of these developments originated at Xerox PARC [Crecine86], and most are well integrated into the Cedar environment. We view electronic mail as a form of electronic publishing and, in fact, multimedia, WYSIWYG Tioga documents are routinely exchanged by electronic mail in Cedar.

Previous work on mathematical systems has not dealt conclusively with the integration of documents and computation. There have been a number of translators for converting the internal expression representations of algebra systems into typeset quality output [Wactlar64, Foderaro79, Fateman87]. However, this has been a one-way path. Once the conversion is done, one cannot subsequently access the expressions for computation. Although a number of interactive WYSIWYG mathematical systems, such as MathCAD [MathSoft87], MILO [Avitzur87], and INFOR [Schelter87], allow the production of some sort of document, only INFOR could be said to support professional-quality typesetting for text and mathematics. However, it has no provision for other media, such as voice or complex graphics, and is not currently connected to computational systems. Recent multimedia document systems such as Diamond [Crowley87] and Andrew [Morris86] support a range of media, but are only beginning to support math in documents and do not yet address the issue of computations with the math in a document. The experimental PEN system [Allen81] had some of the same goals as CaminoReal; its design decisions are a continuing resource as we attempt to build systems of broader scope. Spreadsheet systems can be said to integrate documents and computation, but they are typically limited in the types of documents that can be created and the media that those documents can contain. For us, documents such as multimedia technical papers, automatically generated logs, and audit trails (see Section 5.2 for a discussion of the latter), are crucial parts of a mathematical system. Furthermore, we want all of the documents in our system to be "interactive." That Tioga and CaminoReal provide a professional-quality publishing system may be gauged by the fact that this document has been produced with them. Tioga/CaminoReal documents are interactive. Hence we call CaminoReal a system for interactive mathematical notebooks.

2. Standard mathematical systems

2.1. *Introduction*

The material in this section is adapted from a recent workshop summary [Arnon87]. We postulate that there is an abstract syntax for any mathematical expression, which consists of an operator and a list of arguments, where each argument is (recursively) a piece of abstract syntax (cf. [McCarthy62]). Functional notation, Lisp S–expressions, directed acyclic graphs, and *n*-ary trees are isomorphic representations of abstract syntax. For example, the expression $ab+c$ could be represented in abstract syntax by the functional notation Plus[Times[a,b],c], or by the binary tree:



A "Standard Mathematical Component" (abbreviated SMC) is a collection of software and hardware modules, each with a single function, which, if it reads mathematical expressions, reads them in abstract syntax and which, if it writes mathematical expressions, writes them in abstract syntax. A "Standard Mathematical System" (abbreviated SMS) is a collection of SMCs, which are used together and which communicate with each other in abstract syntax over "wires" that connect them. In Section 2.2 we identify five possible types of SMCs. Any particular SMS may have zero or more instances of each.

2.2. *Standard mathematical components*

2.2.1 *ED - math editors*

An ED component edits abstract syntax to abstract syntax. A particular system may have editors that operate on other representations of mathematical expressions, such as bitmaps or $\TeX$ math notation [Knuth84]; however, such editors do not qualify as EDs.

2.2.2 *DISP - math displayers*

DISP components are suites of software packages, device drivers, and hardware devices that take in an expression in abstract syntax and render it. Two examples might be (1) the combination of an abstract syntax to $\TeX$ translator, plus $\TeX$ itself, plus a printer, or (2) a plotting package plus a plotting device. A DISP component may or may not support "pointing" (selection) within an expression it has displayed. For example, a DISP that renders onto a printer probably doesn't, but a DISP driving a display screen may. If pointing is supported, then the DISP must be able to pass back the selected subexpression(s) in abstract syntax.

2.2.3 *COMP - computation systems*

A COMP takes in an expression in abstract syntax, performs some computation on it, and returns the result in abstract syntax. Examples of COMPs are numerical libraries and computer algebra systems.

2.2.4 *DOC - document systems*

The simplest example of a DOC is a text editor allowing abstract syntax representations of mathematical expressions to be stored in documents. However, a DOC need have no relation to printing, so, for example, database, hypertext, and electronic mail systems are all possible DOCs. Essentially, a DOC is just a system that manipulates certain data structures (which we may think of as "documents" or "databases") in which we can "park" multiple math expressions represented in abstract syntax.

Note that if math is editable in place in a document, this is not part of the DOC *per se*, but the result of cooperation between a DOC, an ED, and perhaps also a DISP. The interactions that occur among the various components when editing an expression in such a situation are: (1) the DISP and the ED process the user's input and modify the abstract syntax representation of the expression, (2) the DISP queries the DOC to be given an area (typically a box) within which to display the new expression, and (3) the DISP paints the expression in the given box.

2.2.5 *MAN - system managers*

It is the function of a MAN to coordinate and connect the ED, DISP, COMP, and DOC components that comprise an SMS. Typically, an SMS has one MAN component, although it might be able to switch between several. SMSs may vary greatly in the face they present to the user, and such differences may well arise from the differing amounts of "knowledge" or sophistication possessed by their MAN components. MAN components do whatever is necessary to make the other components play together and perhaps more besides to make life easier for the user. For example, abstract syntax provides a syntax for communication among SMCs, but leaves unspecified the semantics of sentences in this language. Thus, if the function (i.e., operator) names of the abstract syntax used by an ED component differ from those used by a COMP, the MAN provides the translation. The user of an SMS need not, and is encouraged not to, understand the naming conventions of the particular components it contains. As another example, a MAN may have some "knowledge" of the relative merits of several COMP components for different tasks and provide automatic routing of computational requests to the most appropriate one.

2.3. *Further notes on the overall architecture*

A typical SMS (e.g., CaminoReal) will have a WYSIWYG expression editor that consists of an ED and a DISP that are much more closely coupled than is suggested here. For example, the internal representations of abstract syntax in an ED and a DISP (such as a tree of boxes), might have pointers back and forth, or the two may even share a common data structure. This is acceptable, but it should always be possible to access the two components in the canonical decoupled way. This would mean that the ED should be able to receive a standard abstract syntax representation for an expression plus an editing command in abstract syntax (for example, Edit[expr, cmd]) and return an abstract syntax representation for the result. Similarly, the DISP should be able to receive abstract syntax over the wire and display it and, if it supports pointing, be able to return selected subexpressions in abstract syntax.

Since abstract syntax is human-readable, any text editor can be used as an ED. However, many users of an SMS will only want to interact with mathematics that has a typeset appearance; they should not need to know that their system talks abstract syntax within itself or to the outside world.

Katz' recent proposal [Katz87] and others in the typesetting and document communities, distinguish the form (appearance) of a mathematical expression from its content (meaning or value). It is a thesis of the SMS architecture that such a distinction cannot usefully be made. Rather, we claim that abstract syntax can convey form, content, or both, and that its interpretation is strictly in the eye of the beholder(s). In other words, "meaning is just a handshake between sender and recipient" [Arnon87].

3. CaminoReal as a prototype SMS: basic editing, documents

3.1. *The expression editor (ED and DISP)*

A stand-alone interactive expression editor called Meddle is "in control" when a CaminoReal tool (i.e., viewer) is instantiated. Meddle's user interface paradigm follows Tioga quite closely. Multiple selections are made via a mouse pointing device. Operations upon the selections are initiated both from the keyboard and from mouse activation of menu commands. These input actions are parsed by a user interface management system that uses state transitions rather than a formal grammar specification. Incremental modifications to the expression are updated on the display after each keystroke or mouse action. There may be more than one CaminoReal tool active at any one time. Meddle provides several "creature comforts" suited to interactive editing of complex notation, especially the undo command and various scaling operations to improve the readability on low resolution display screens.

Meddle's internal data structure for mathematical expressions is *n*-ary trees, with operators as interior nodes and atoms as leaves (i.e., abstract syntax). For each operator, there is a template that defines its displayed notation. The components of such a notation are zero or more glyphs representing the "operator name," a notation for each subexpression (these are either atomic notations or, recursively, notations of the same sort), and any desired additional glyphs. For example, the summation template specifies that a Greek sigma, Σ, be used to denote the operator and that there be three subexpressions: a lower bound, an upper bound, and a summand. Such templates are written as procedures in the Cedar language and dynamically loaded with CaminoReal. When templates are first presented, any unfilled subexpressions are represented by placeholders ($\boxtimes$). CaminoReal tools are initialized to contain an expression consisting of the single placeholder $\boxtimes$.

If, for example, we replace a selected placeholder with a summation template (chosen from a menu), we see

$$\sum_{\boxtimes}^{\boxtimes} \boxtimes .$$

If we then replace the summand placeholder with an indefinite integral template, we get

$$\sum_{\boxtimes}^{\boxtimes} \int \boxtimes \, d\boxtimes .$$

We may continue to fill in placeholders, eventually arriving, for example, at

$$\sum_{i=0}^{\infty} \int \frac{1}{x^i} \, dx .$$

One may select a subexpression corresponding to a subtree by pointing at it and clicking the mouse; the actual selection is the "smallest" subexpression whose bounding box contains the hit. This strategy insures that Meddle constructs only structurally correct expressions, at the expense of freedom to manipulate all the glyphs in a notation. The intention is that additional operator templates be added to accommodate new notation rather than permitting the user to rearrange subexpressions arbitrarily. To facilitate keyboard entry, a complete set of navigational commands is provided, such as select parent, sibling, or child of the current selection.

Meddle depends on four types of selections: primary, copy, move, and keyboard. The *primary selection*, made by pointing and clicking with the mouse, establishes the focus of most operations. The *copy selection*, made by chording (holding down) the keyboard SHIFT key and clicking the mouse, supplies the argument for copying a subexpression to replace the primary selection. A *move selection*, made by chording the keyboard CTRL key and clicking the mouse, supplies the argument for moving a subexpression to replace the primary selection. The *keyboard selection* is a "primary selection in progress" for keyboard input of multi-character identifiers and numbers. If, for example, a primary selection exists and we strike 'M' on the keyboard, then a new keyboard selection of M is set and replaces the previous primary selection. If we next strike 'a', we get a new keyboard selection of (the identifier) Ma, whereas if we next strike '↑', the keyboard selection M is converted into the primary selection, wrapped (see next paragraph) with a superscript template, and we get $M^{\boxtimes}$ with the primary selection set to the superscript placeholder. When two or more selections are active at one time, e.g., primary and copy, the selected expressions can be in different CaminoReal viewers.

Modifications to the primary selection may occur by either a *replace* or a *wrap* operation. In replace, the subexpression in the primary selection is deleted and a new subexpression, determined by either keyboard input or a menu choice, is inserted in the expression tree. In a wrap, the primary selection is retained and replaces one of the placeholders in the new subexpression. For example, consider the first placeholder in the addition template $\boxtimes+\boxtimes$ as the primary selection. Typing the letter 'r' replaces the first placeholder with r, resulting in $r+\boxtimes$, and subsequently typing the key '↑' wraps a superscript template around the r to produce $r^{\boxtimes}+\boxtimes$ (whereas replacing the r with a superscript template, chosen from a menu, would produce $\boxtimes^{\boxtimes}+\boxtimes$).

All editing operations are available via menu buttons; certain common ones can be invoked from the keyboard. We have not so far succeeded in harmonizing the goal of simple and uniform semantics for selections and wraps with the goal of supporting standard operator precedence in keyboard input. For example, one might expect to type the keyboard sequence 'x↑2+1=0' to enter $x^2+1=0$, but we actually require 'x↑2<CTRL-P>+1<CTRL-P>=0', in which the <CTRL-P> keystroke invokes the 'select parent' operation.

Previous mathematical expression editors include the Xerox Star [Xerox86b, Becker87], EDIMATH [Quint83, 84], DREAMS [Foster84], MathScribe [Smith86], MILO [Avitzur87], and INFOR [Schelter87]. The Star editor has existed since the mid-1970's, and although we, like many others, have been influenced by it, Meddle has been designed and built from scratch in Cedar.

3.2. *Character sets*

CaminoReal deals with the challenge of presenting the rich collection of multilingual and technical symbols in mathematics by utilizing the Xerox character code standard [Xerox86a]. This standard assigns a unique code (possibly 1, 2 or 3 bytes long) to each symbol and subsumes several international standards, such as ISO and JIS, as well as *de facto* standards, such as the AMS TEX math symbols [Knuth84]. Fonts that conform to the Xerox character code standard have mathematical symbols in standard code positions. Should a font lack particular symbols, a backstop "kitchen sink" font with a glyph for every code is automatically substituted. The scalable font algorithms in the Cedar Imager [Swinehart86] insure that symbols appear on the display or in the document with appropriate size and shape information.

Nonetheless, there remains the challenge of entering symbols that do not occur on typical keyboards. CaminoReal employs a tentative solution by using menus for special symbols and Greek letters. Other Xerox systems employ the attractive techniques of virtual keyboards to map keys into arbitrary character codes, and abbreviation name lookup translations, to map token identifiers into symbols [Becker87].

3.3. *The integrated document formatter (DOC and DISP)*

The integration of mathematical content into a Tioga multimedia document is accomplished by defining a mathematical object class in Tioga. One part of this class definition is the specification that the "data" for particular mathematical objects (i.e., expressions) in documents are strings. The actual string used for a given expression is just a linearization of Meddle's internal data structure for it. Thus, moving expressions between Tioga documents and CaminoReal tools is straightforward. Also included in the Tioga math object class definition are (Cedar) procedures that return a bounding box for an expression and that paint it into a given imaging context. These are merely slightly revised versions of Meddle procedures for similar tasks. Mathematical objects in Tioga documents may occur either in-line (i.e., within a line of text) or displayed (i.e., set out); both uses can be seen in this paper.

Tioga actually views a math object as a single, decorated character, so expressions can be Tioga selections just as any other character. Putting this another way, mathematical expressions in Tioga documents are indivisible units: they can be moved around within the document just like text, but there is no way to get at subexpressions while they reside in the document. Thus math in a Tioga document cannot currently be edited "in place," but must be extracted into a CaminoReal tool, edited there, and reinserted into the document. However, the expression evaluation operations provided by the CaminoReal tool interface (cf. Section 4) can operate equally well on Tioga and CaminoReal selections. Thus computation on expressions in documents can be performed by simply pointing at them (in the document) and invoking the desired actions in a CaminoReal tool.

4. CaminoReal as a prototype SMS: computation

CaminoReal provides two methods for algebraic computation, both accessible through the same tool interface: (1) an experimental domain-oriented package, AlgebraStructures, executed locally in Cedar, and (2) several traditional algebra packages accessed as algebra servers over the network. One may evaluate expressions at several levels of detail, from selected subexpressions to complete expressions to entire documents (see Section 5.1 for an example of the latter). The result of an evaluation operation may either be presented in place or in a new CaminoReal tool. The combination of these computation facilities and the interactive editing environment, especially the latter's undo command, provides a handy tool for exploring complicated expressions.

4.1. *The AlgebraStructures package*

The Cedar AlgebraStructures package supports a limited set of algebraic operations defined within mathematical domains. Primitive (built-in) domains include general expressions, booleans, Cedar integers, arbitrary precision rationals, Cedar reals, and Cedar complexes. Constructors for building new domains of sets, sequences, vectors, matrices, and polynomials over suitable existing domains are provided. The application of the AlgebraStructures evaluator to a Meddle expression proceeds bottom-up: given an operator and domains for its (evaluated) arguments, a suitable domain (general expressions, if nothing narrower can be determined) and value for that operator's subtree is determined. Every AlgebraStructures domain has procedures for generating Meddle representations of its elements, which are invoked to display the result of an evaluation.

For example, if we create the following Meddle expression in a CaminoReal tool:

$$\begin{bmatrix} 1 & 0 & -3x & -x^3 & 0 \\ 0 & 1 & 0 & -3x & -x^3 \\ x & -15x^3 & -1 & 0 & 0 \\ 0 & x & -15x^3 & -1 & 0 \\ 0 & 0 & x & -15x^3 & -1 \end{bmatrix}$$

and apply the AlgebraStructures evaluator to it, its domain will be determined to be 5X5 matrices of polynomials in x with integer coefficients. Evaluation of the (1,1) element of the matrix finds its domain to be the integers. We can select the matrix and ask for the operations available on it, which causes AlgebraStructures to evaluate it to determine its domain, and then display a menu of the operations available on elements of that domain. For the matrix, one of the entries of this menu is Determinant, which if selected yields a result of

$$-3375x^{12}+46x^9+630x^7-9x^4+6x^2-1\,,$$

whose domain is polynomials in x with integer coefficients.

Domains could be useful for run-time checks that mathematical expressions belong to some asserted domain when evaluated. Such "semantic" checking is not the same as the "syntax" checking that is performed on Meddle keyboard input. Domain information is also useful for guiding a MAN component to choose appropriate algebra servers for particular operations. For example, the AlgebraStructures package currently uses the SAC-2 package for polynomial GCD calculations but does its own integer GCD computations. The design of the algebra system using domains provides a convenient framework for organizing such choices.

The design of AlgebraStructures was influenced by VIEWS [Abdali86] and SCRATCHPAD [Jenks84], among others.

4.2. *Computation with algebra servers*

Algebraic computation with systems other than AlgebraStructures is accomplished by selecting an expression or subexpression in a CaminoReal viewer and sending it to the desired algebra server (COMP) for evaluation. CaminoReal's MAN first converts the expression to abstract syntax appropriate for that server, and then sends it to the server over the network. The result of the algebra system's evaluation returns to CaminoReal by the same steps in reverse. Currently, a new process on the remote machine is created for each such request.

Suppose we have the following integral in a CaminoReal tool:

$$\int \frac{1-2x^3}{(1+x^3)^2}\,dx$$

Since AlgebraStructures has no integration algorithms, its evaluator returns a value of:

$$\int \frac{-2x^3+1}{x^6+2x^3+1}\,dx$$

If we send the integral to Reduce for evaluation, after about ten seconds we get back:

$$\frac{x}{x^3+1}$$

in our CaminoReal viewer.

The abstract syntax into which CaminoReal converts the integral for transmission to Reduce is:

```
int(quotient(difference(1, times(2, expt(x, 3))), expt(( plus(1, expt(x, 3)) ), 2)), x)
```

For transmission to SMP, the abstract syntax is:

```
Int[Div[Minus[1, Mult[2, Pow[x, 3]]], Pow[( Plus[1, Pow[x, 3]] ), 2]], x]
```

Thus, CaminoReal allows the user to compute with different algebra systems without knowing the different command names and styles used in each. CaminoReal's style of interchange of mathematical expressions may be contrasted with the encoded directed-acyclic graphs of the Iris system [Leong86].

4.3. *Issues for future work on computation*

The ability to access multiple algebra systems through CaminoReal has raised several issues requiring future work. One is the synchronization of outstanding algebra requests with concurrent interactive editing. Because some algebra operations may take a considerable length of time, it is unreasonable to suspend editing completely during such an operation. Long running computations also raise the need for inquiries to the algebra system for status and progress reporting, as well as additional controls to suspend or abort computations.

It would be desirable to permit multiple concurrent algebra requests from CaminoReal. This raises two issues: (1) the multiplexing of algebra requests to a single server and (2) the definition of the computational state for each request. The simplest scheme would probably be to fork a separate process for each request, giving it a clean initial state, with a limit on the number of such processes active at any one time, and a queue for requests that arrive after the limit is reached.

Additional error-handling and error-reporting protocols are necessary for the full incorporation of algebra servers into the CaminoReal user interface. At present, the data and message streams are combined into one, making it difficult to identify errors returned by the server. The overhead and low bandwidth of communications between clients and servers may mandate compression and caching of conversations, facilities which are provided by Cedar's underlying communications software, but are not currently used in CaminoReal.

5. Computed documents

5.1. *Mathematical form letters*

CaminoReal's expression language supports an assignment statement. Also, expressions assigned to variables (by the evaluation of an assignment statement) are maintained in a symbol table that is global across all Tioga documents and all CaminoReal tool instances, and so constitutes an environment with respect to which evaluations are performed. Variables which have not been assigned values evaluate to themselves. Supporting utilities include the function $killAll[]$, which clears the environment, the notation ❜*expression*❜ (abstract syntax: quote[*expression*]), whose semantics are that $Eval[$❜*expression*❜$]$ returns *expression* unevaluated, and the function $killVariable[$❜t❜$]$, which removes the value of the variable t from the environment.

When CaminoReal's internal EvalBeforePaint switch is off (the default), math expressions in a Tioga document are painted just as they are stored. When EvalBeforePaint is on, they are evaluated before being painted. As might be expected, the order in which expressions are evaluated and painted in the latter case is left-to-right and top-to-bottom order in the Tioga document. A Cedar command is provided for toggling the EvalBeforePaint switch. References to previous expressions in the definitions of later expressions (via the symbol table), coupled with EvalBeforePaint on, make possible Tioga documents that are spreadsheets, "mathematical form letters," or, more generally, "computed documents." For example, here is the "definition level" (EvalBeforePaint off) of a sample form letter:

Let F be defined to be $F \leftarrow y^3-3xy+x^3$.
Let G be defined to be $G \leftarrow xy^2-15x^3y+1$.
Let M be the Sylvester matrix of F and G:

$$M \leftarrow sylvesterMatrix[F,G]$$

Then the resultant is the determinant of M, that is

$$det[M]$$

The right sides of the first two assignment statements constitute the "input data" to the form letter, in the sense that the right sides of the last two assignment statements are functions of them. If we now turn EvalBeforePaint on, and simply repaint the document (on screen or printer), we get:

Let F be defined to be $y^3-3xy+x^3$.
Let G be defined to be xy^2-15x^3y+1.
Let M be the Sylvester matrix of F and G:

$$\begin{bmatrix} 1 & 0 & -3x & x^3 & 0 \\ 0 & 1 & 0 & -3x & x^3 \\ x & -15x^3 & 1 & 0 & 0 \\ 0 & x & -15x^3 & 1 & 0 \\ 0 & 0 & x & -15x^3 & 1 \end{bmatrix}$$

Then the resultant is the determinant of M, that is

$$3375x^{12}-44x^9-720x^7+9x^4+6x^2+1$$

Suppose now that we change the input data, for example, let us replace (at the definition level) the expression $F \leftarrow y^3-3xy+x^3$ by $F \leftarrow y^3-x^3$. Then by turning EvalBeforePaint on and repainting the document, we get the correctly updated matrix and determinant:

Let F be defined to be $-x^3+y^3$.
Let G be defined to be xy^2-15x^3y+1.
Let M be the Sylvester matrix of F and G:

$$\begin{bmatrix} 1 & 0 & 0 & -x^3 & 0 \\ 0 & 1 & 0 & 0 & -x^3 \\ x & -15x^3 & 1 & 0 & 0 \\ 0 & x & -15x^3 & 1 & 0 \\ 0 & 0 & x & -15x^3 & 1 \end{bmatrix}$$

Then the resultant is the determinant of M, that is

$$-3375x^{12}+x^9+45x^7+1$$

Having the math in a technical paper computed "on the fly" minimizes the introduction of typographical errors and thereby lightens the proofreading task.

5.2. *Future work on automated proofreading, spreadsheets, audit trails*

The example in the previous section showed the use of the expression environment and evaluation for generating documents. In the future, we expect to refine the use of these tools to provide more proofreading features. For example, suppose in the form letter above that we stored both the unevaluated and evaluated form of each expression. Then one need only do the (possibly expensive) evaluations occasionally, when one wishes to check and verify that the document is correct. The unevaluated expressions could thus be viewed as "assertions" about the actual expressions in the document, which can be checked by evaluating them and checking for equality with the evaluated form.

Because current spreadsheets may contain formulae and perform computations, it is natural to consider the extension of CaminoReal to provide such capabilities in Tioga documents. An interesting issue is how to provide a naming scheme for math equations and expressions that is equivalent to the cell-naming scheme of typical spreadsheets. CaminoReal's current use of a global symbol table presents some difficulties. A recent discussion of spreadsheets [Davis86] raises many of the issues of interest to CaminoReal.

Similarly, an audit trail of computations within a document is a natural extension of the ability to compute expressions. The audit trail would retain the dependency of subexpressions as well as the mathematical operations used to compute the results, so that when one expression was edited, all of the relevant changes elsewhere in the document would be computed and updated. Audit trails could be examined or edited to modify the derivation of new results.

Acknowledgments

Thanks to Michael Plass, Ken Pier, and Christian LeCocq for technical assistance. Thanks to those in the Computer Science Lab at PARC who have used CaminoReal for their documents and computations, and suggested improvements, particularly James Rauen. Thanks to Alan Perlis and Alan Demers for inspiration.

Dennis Arnon has been the chief architect and maintainer of CaminoReal and wrote AlgebraStructures. Carl Waldspurger wrote Meddle. Kevin McIsaac has handled the care and feeding of algebra servers and contributed performance improvements and design ideas. Rick Beach had the first vision of the system architecture, helped build the connections between Meddle and Tioga, and has provided continuous guidance and moral support.

References

[1] Abdali, K., Cherry, G. & Soiffer, N. (1986). An object-oriented approach to algebra system design. In *Proc. Symp. Symbolic and Algebraic Computation*, ed. B. Char, 24-30. New York, NY: ACM.

[2] Allen, T., Nix, R. & Perlis, A. (1981). PEN: a hierarchical document editor, *ACM SIGPLAN Notices*, **16**, 6, 74-81.

[3] Arnon, D. (1987). Report of the workshop on environments for computational mathematics, *Request for Comments No. RFC1019*. Menlo Park, CA: ARPANET Information Center.

[4] Avitzur, R. (1987). MILO system [a MacIntosh program]. Palo Alto, CA.

[5] Becker, J. (1987). Arabic word processing, *Comm. ACM*, **30**, 7, 600-610.

[6] Berman, R. & Kulp, J. (1979). A new environment for computational physics. In *Proc. MACSYMA Ùsers Conf.*, ed. V. Ellen Lewis, 622-632. Cambridge, MA: Laboratory for Computer Science, MIT.

[7] Bloomberg, D. & Hogg, T. (1987). Engineering/scientific workstation project, Xerox Palo Alto Research Center, GSL-87-01.

[8] Bundy, A. (1985). Discovery and reasoning in mathematics. In *Proc. Int. Joint Conf. Artif. Intell.*, 1221-1230.

[9] Calmet, J. & Lugiez, D. (1987). A knowledge-based system for computer algebra, *ACM SIGSAM Bulletin*, **21**, 1, 7-13.

[10] Clapp, L. & Kain, R.Y. (1963). A computer aid for symbolic mathematics, In *Proc. AFIPS Fall Joint Comp. Conf.*, **24**, 509-517.

[11] Collins, G. (1980). ALDES and SAC-2 now available, *ACM SIGSAM Bulletin*, **14**, 2, 19.

[12] Crecine, J. (1986). The next generation of personal computers, *Science*, **231**, 4741, 935-943.

[13] Crowley T., Forsdick, H., Landau, M. & Travers, V. (1987). The Diamond multimedia editor. In *Proc. Unix Users Conf. (USENIX)*, 1-17.

[14] Davis, R. (1986). Knowledge-based systems, *Science*, **231**, 4741, 957-963.

[15] Donahue, J. (1985). Integration mechanisms in Cedar, *ACM SIGPLAN Notices*, **20**, 7.

[16] Engeler, E. (1985). Scientific computation: the integration of symbolic, numeric and graphic computation, *Lecture Notes in Computer Science*, **203**, 185-200. New York, NY: Springer-Verlag.

[17] Fateman, R. (1987). TEX output from MACSYMA-like systems. Dept. of Electrical Engineering and Computer Science, Univ. of CA, Berkeley, unpublished manuscript.

[18] Foderaro, J. (1979). Typesetting MACSYMA equations In *Proc. MACSYMA Users Conf.*, ed. V. Ellen Lewis, 345-361. Cambridge, MA: Laboratory for Computer Science, MIT.

[19] Foster, G. (1984). *DREAMS: display representation for algebraic manipulation systems*, Dept. of Electrical Engineering and Computer Science, Univ. of CA, Berkeley, CSD 84/193.

[20] Genesereth, M. (1977). An automated consultant for MACSYMA, *Proc. MACSYMA Users Conf.*, publication NASA CP-2012, 309-314. Washington DC: National Aeronautics and Space Administration.

[21] Griesmer, J. & Jenks, R. (1971). Scratchpad/I - an interactive facility for symbolic mathematics. In *Proc. Second Symp. Symbolic Algebraic Manip. (SIGSAM '71)*, ed. S. Petrick, 42-58. New York, NY: ACM.

[22] Hearn, A. (1980). The personal algebra machine, *Proc. IFIP '80*, 620-628. Amsterdam: North-Holland.

[23] Hearn, A. (1982). REDUCE - a case study in algebra system development, *Lecture Notes in Computer Science*, **144**, 263-272. New York, NY: Springer-Verlag.

[24] Jenks, R.D. (1984). A primer: 11 keys to new SCRATCHPAD, *Lecture Notes in Computer Science*, **174**, 123-147. New York, NY: Springer-Verlag.

[25] Katz, A. (1987). Issues in defining an equations representation standard, *ACM SIGSAM Bulletin*, **21**, 2, 19-24.

[26] Knuth, D.E. (1984). *The TEXBook*, Reading, MA: Addison-Wesley.

[27] Koved L. & Shneidermann B. (1986). Embedded menus: selecting items in context, *Comm. ACM*, **29**, 4, 312-318.

[28] Leong, B. (1986). Iris: design of a user interface program for symbolic algebra. In *Proc. Symp. Symbolic and Algebraic Computation*, ed. B. Char, 1-6. New York, NY: ACM.

[29] Licklider, J. (1960). Man-computer symbiosis, *IRE Trans. on Human Factors in Electronics*, **HFE-1**, 4-11.

[30] Martin, W. (1967). *Symbolic Mathematical Laboratory (Ph.D. dissertation)*, Cambridge, MA: MIT.

[31] Martin, W. & Fateman, R. (1971). The MACSYMA system. In *Proc. Second Symp. Symbolic Algebraic Manip. (SIGSAM '71)*, ed. S. Petrick, 59-75. New York, NY: ACM.

[32] MathSoft Inc. (1987). MathCAD system. Cambridge, MA.

[33] McCarthy, J. (1962). Towards a mathematical theory of computation, *Proc. IFIP '62*, Amsterdam: North-Holland.

[34] Minsky, M. (1963). *MATHSCOPE, part I - a proposal for a mathematical manipulation-display system*, Artificial Intelligence Project, Project MAC, MIT, MAC-M-118.

[35] Morris, J. *et al.* (1986). Andrew: a distributed personal computing environment, *Comm. ACM* **29**, 3, 184-201.

[36] Ng, E. (1979). Symbolic-numeric interface: a review, *Lecture Notes in Computer Science*, **72**, 330-345. New York, NY: Springer-Verlag.

[37] Pier, K., Bier, E. & Stone, M. (1988). Gargoyle: an interactive illustration tool, *Proc. EP'88 Int'l Conf. on Electronic Publishing, Document Manipulation, and Typography*, Nice, France.

[38] Quint, V. (1983). An interactive system for mathematical text processing, *Technology and Science of Informatics*, **2**, 3, 169-179.

[39] Quint, V. (1984). Interactive editing of mathematics. In *Proc. First Intl. Conf. Text Processing Systems*, 55-68. Dublin, Ireland: Boole Press.

[40] Schelter, W.F. (1987). *INFOR display editor*. Department of Mathematics, Univ. of Texas-Austin, unpublished manuscript.

[41] Shneiderman, B. (1983). Direct manipulation: a step beyond programming languages, *IEEE Computer*, **16**, 8, 57-69.

[42] Smith, C. & Soiffer, N. (1986). MathScribe: a user Interface for computer algebra systems. In *Proc. Symp. Symbolic and Algebraic Computation*, ed. B. Char, 7-12. New York, NY: ACM.

[43] Spirkovska, L. (1986). MUFIE: MACSYMA's user friendly interactive executive, Dept. of Electrical Engineering and Computer Science, Univ. of CA, Berkeley, M.Sc. report.

[44] Sundblad, Y., (1974). Symbolic mathematical systems now and in the future, *ACM SIGSAM Bulletin*, **8**, 3, 1-8.

[45] Suppes, P. *et al.* (1987). *Applications of computer technology to pre-college calculus, first annual report.* Inst. for Math. Studies in the Social Sci., Stanford University, Psych. and Ed. Series TR #310.

[46] Swinehart, D., Zellweger, P., Beach, R. & Hagmann, R. (1986). A structural view of the Cedar programming environment, *ACM Trans. Prog. Lang. Systems*, **8**, 4, 419-490.

[47] Wactlar, H. & Barnett, M. (1964). Mechanization of tedious algebra – the *e* coefficients of theoretical chemistry, *Comm. ACM*, **7**, 12, 704-710.

[48] Wolfram, S. (1985). Symbolic mathematical computation, *Comm. ACM*, **28**, 4, 390-394.

[49] Xerox Corp. (1986). *Character code standard.* Xerox System Integration Standard XSIS 058605.

[50] Xerox Corp. (1986). Viewpoint Document Editor: Viewpoint Series Reference Library Version 1.1. El Segundo, CA.

[51] Zellweger, P., Terry, D. & Swinehart, D. (1988). An overview of the Etherphone system and its applications, *Proc. 2nd IEEE Conf. on Computer Workstations*, Santa Clara, CA.

[52] Zellweger, P. (1988). Active paths through multimedia documents, *Proc. EP'88 Int'l Conf. on Electronic Publishing, Document Manipulation, and Typography*, Nice, France.