A Front-end Tool for Mathematical Computation and Education in a Network Environment

T. Sakurai[1], Y. Zhao[2], H. Sugiura[3] and T. Torii[4]

[1]University of Tsukuba
[2]Nagasaki Institute of Applied Science
[3]Nagoya University
[4]Nanzan University

Abstract

In this paper, we discuss what is required for a computer program to assist in the process of mathematical thinking, writing, computing, and the exchange of results and algorithms. A front-end tool for mathematical computation and education is proposed. This front-end tool is developed in JAVA, and can be run over a network using a Web browser. Common notations are used in the whole process of computing. Users can browse a document including mathematical expressions, and manipulate these expressions. Moreover, this tool is useful in mathematical education and network-based communication of mathematical expressions.

1. Introduction

As computer technology, and networking in particular, progresses, many computer programs and mathematical results are distributed over different networks. Exchange of ideas and products by networks plays an important role in cooperative work and research. In particular, computer algebra systems (CASs) are useful for assisting in mathematical thinking by their brief expressions, graphical output, and interactive operations. However, they use various types of descriptions for mathematical results, and computer programs. We can not directly manipulate and access mathematical expressions on the Web, and we still can not send electronic mail that includes mathematical expressions.

In mathematical education using CASs, there is still a problem that both teachers and students have to learn one or even more than one front-end language of CASs. They still have to use some unfamiliar representation, and translate between these representations and mathematical notation. In order to let teachers and students focus on their mathematical problem-solving directly, we establish a common interface using natural mathematical notations as close as possible. In addition, the Web has been used as a tool of Internet based mathematical education [FYT98], [HCM], [HIE], [HJE], [HMA], [HPM], [Mar97], mathematical expression is also a problem in Web-based mathematical education.

In this paper, we discuss what is required for a computer program to assist in the process of mathematical thinking, writing, computing, and the exchange of results and algorithms. Mathematical expressions are powerful in representation and universal in use. User interfaces which can treat mathematical expressions have been developed for many CASs [DW90], [Kaj92], [KS98], [LW95], [Soi95],

[YW87]. Several CASs (Mathematica, MathView, and MathCAD) have two-dimensional front-ends for editing mathematical expressions. Many CASs also have interactive Web browsing front-ends, but they can only receive Fortran-like textual mathematical expressions.

In [ZSTS94] and [ZSST98], a formal representation to treat natural mathematical notations has been defined. Formalized mathematical expressions can be translated into source code of high-level programming languages, front-end languages of mathematical computation systems, and LATEX source files for typesetting. A user interface to generate the formal representation has also been proposed in [SZST96]. Continuing on these results, we propose a front-end tool for mathematical computation and education.

The current prototype of this front-end tool consists of 1) a graphical user interface (GUI) to input and edit a document that includes mathematical expressions, 2) filters to output front-end languages of computer algebra systems and typesetting languages, and 3) a client-server program to manage communication between users and a central host. The front-end tool is developed in JAVA, and can be run over a network using a Web browser. This tool makes it possible to use common notations in the whole process of computing, i.e., from editing of a computing requirement to obtaining and re-editing of its computing result. Users can browse a document including mathematical expressions, and manipulate any sub-structures of these expressions.

Many systems for scientific computation, simulation, and education have been developed and used at present. A front-end tool which uses a specific language or notation system can not be adapted for various systems. Our front-end tool represents mathematical expressions as combinations of simple structures. The translation to front-end languages of CASs and typesetting languages only depends on filters, thus we can adapt a filter to output suitable representations for various systems.

2. Formalization of Mathematical Notations

In this section, we consider a formalization to treat mathematical expressions on a computer preserving their computational meanings. We usually use mathematical expressions for mathematical thinking. However the correctness of mathematical expressions does not always guarantee that the numerical results of the corresponding computer program, or the printed document, will agree. The following expression

$$y_i = \sum_{j=1}^{m} \alpha_{ij} \beta_j, \quad i = 1, \ldots, n$$

corresponds in FORTRAN with

```
DO I=1,N
   GAMMA(I)=0.0
   DO J=1,M
```

```
        GAMMA(I)=GAMMA(I)+ALPHA(I,J)*BETA(J)
      ENDLOOP
    ENDLOOP
```

, and its LATEX (a popular typesetting language among mathematicians) representation is

```
\gamma_i=\sum_{j=1}^m \alpha_{ij}\beta_j, \quad i=1,\ldots,n
```

We need different expressions for computing or printing. Possible error in their notations should be checked. For example, if we forget

```
GAMMA(I)=0.0
```

in the above FORTRAN program, then this program will produce incorrect results, while the assignment does not have a direct correspondence to the original mathematical expression. In the LATEX expression, if we mistakenly type a letter "n" instead of the letter "m" for the upper range of the sum, then its correct meaning will be lost.

If we can use common notations through the process of thinking, computing, and printing, then these mistakes will be reduced. To unify descriptions, we have focused on mathematical notations as common expressions.

Mathematical expressions consist of two-dimensional structures such as superscript, subscript, fraction, etc. TEX, a language for typesetting a document including mathematical expressions, can be used to represent the structures. However, if we only express the location of symbols by TEX, the corresponding structure will be lost. For example, the notation

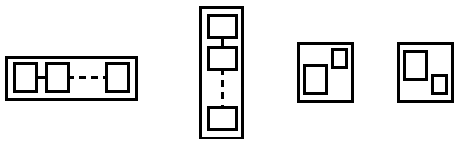
is represented in TEX by

```
{}_m C_n
```

This notation only shows that the letter "m" is located at the position of subscript, and it does not include the connection between the letters "m" and "C".

Structure of mathematical expressions

To represent structures of mathematical expressions on a computer, we must formalize mathematical expressions. In mathematical expressions, font types and locations of symbols have specific meanings, while the size of these symbols does not show different meanings. The connection of these symbols also has a meaning. A box representation for these two-dimensional structures, e.g.,



preserves information of locations of symbols and their connections. A set of all these structures of expressions is called formal representation (FR in short). In our

former research [ZSTS94] and [ZSST98], the simplest FR has been defined and used. FR has their corresponding textual or internal representation, like as Row[...], Column[...], Superscript[...], Subscript[...], etc. We can denote various mathematical expressions in combination of this FR. However, this simplest FR is inefficient for a human interface in practice. For example, (a, b) may mean a matrix, a vector, a complex number, a g.c.d., etc. They can not be distinguished by its structure, domain, and context declarations. However, through appending the templates for matrix, vector, complex number and g.c.d. in the editing interface, and through extending the corresponding FR to introduce the structures such as Matrix[...], Vector[...], Complex[...], and GCD[...], we can avoid this ambiguous problem. In fact, many structures can be easily input through manipulating templates over a human interface. Thus it is unnecessary to do a complicated parsing to them.

Translation into program code and typesetting languages

FR expressions are translated into corresponding programming languages or typesetting languages through respective filters.

We can define complicated expressions by combining basic FR. We only have to modify filters a little to realize the translation for this extension. For example, the product

$$\prod_{j=0}^{n}(x_k - x_j)$$

is represented by FR as

```
Product[
  Row["j","=","0"],
  "n" ,
  Row["(",Subscript["x","k"],"-",Subscript["x","j"],")"]
]
```

We modify a translation rule for the case that the first argument of Product[ ] has two rows. Then we can treat the following expression.

$$\prod_{\substack{j=0 \\ j\neq k}}^{n}(x_k - x_j)$$

The corresponding FR of this expression is as follows.

```
Product[
  Column[Row["j","=","0"],Row["j","\ne","k"]],
  "n" ,
  Row["(",Subscript["x","k"],"-",Subscript["x","j"],")"]
]
```

If we define a translation rule for this type of FR, then we get the corresponding Mathematica representation as follows.

```
Product[x[k]-x[j],{j,0,k-1}] Product[x[k]-x[j],{j,k+1,n}]
```

We also obtain its LaTeX representation like as

```
\prod_{{\scriptstyle j=0} \atop {\scriptstyle j \ne k}}^{n}
(x_{k} - x_{j})
```

This extension of notations only depends on a filter, thus we can obtain the same flexibility for various computational and educational systems.

3. A Front-end Tool for Mathematics over Networks

HTML becomes very popular for publishing on the Web. Because of the lack of support for mathematical expressions on the Web, it is not easy to publish mathematical documents on the Web. An extension of HTML to treat mathematical expressions on the Web has been progressed and implemented [HWA] (but has not been released up to the present). However, its design purpose is only the Web documentation for browsing, not the interactive manipulation of mathematical computation.

Communication tool

Exchange of views and products by networks plays an important role in cooperative work and research. To send a document that includes mathematical expressions, we often have to denote crude mathematical expressions as follows:

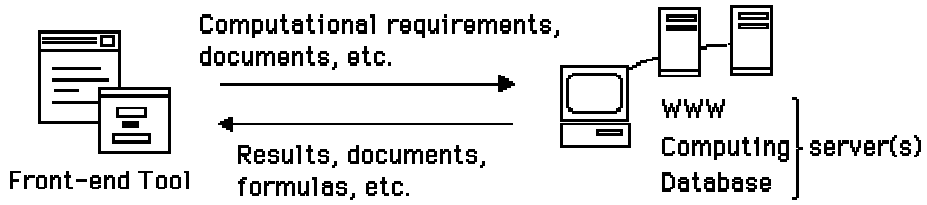
```
Let w_K(z) = z^K - 1 and let t_k = exp(2 pi i k/K), k=1,...,K.
From the definition of <<,,

                K    g_{K-1}(t_k)
<<1, z^p = \sum --------------- t_k^p
            k = 1  w'_K(t_k)

       1     K
   = ---- \sum  g_{K-1}(t_k) t_k^{p-K+1}
       K   k = 1

       1     K
   = ---- \sum g_{K-1}(exp(2 pi i k/K)) exp(-2 pi i k/K)^{K-1-p}
       K   k = 1
```

When a document includes more complicated expressions, we prefer to send a LaTeX source code or Postscript file. These methods are awkward to make communication instantly. Such expressions can not be evaluated or calculated directly up to the present.

Client-Server

The Web is useful to distribute a hyperdocument, whereas it is less suitable for the interactive communication of mathematical expressions between a user and a host. However, by using our front-end tool, users can send a computation requirement to a server. Then the computation requirement is executed on the server, and its computational result is replied to the user; or it is directly executed on the client. Thus, if users have a Web browser, and know how to express mathematical notations, they can do mathematics immediately by using the front-end tool in the

way shown below. It is also possible to use this client-server architecture to establish a database for formulas and algorithms on the Web. Hereby, enabling users to instantly access documents from this database for their own problem-solving algorithms.



4. Implementation of the Front-end Tool

In this section, we describe the outline of the front-end tool, and show an example that has been obtained by using this front-end tool. The current prototype of the front-end tool is developed in JAVA. A GUI provides an editor to browse, input, and modify documents that include mathematical expressions. Some templates and menus are prepared to input two-dimensional structures of expressions. An input document is represented as FR in the tool. This inner-representation can be translated into front-end languages of CASs or typesetting languages by using filters. By adding filters, we will be able to get a different output for other systems.

After a user connects to the WWW server by using a Web browser on a local machine, Java programs of this tool are loaded, and some windows for editing appear on the client. Some data of fonts and symbols are also loaded from the server. When an extended FR file on the server is loaded, a corresponding document is displayed on a window. This document can be modified by the user. Some mathematical symbols and structures can be input by using palettes and menu items. By using a filter of this tool, the user obtains its corresponding source code file of a CAS on the local machine. This file can be executed by a CAS on the local machine.

Client-server programs enable the user to exchange text files with the server. The user can send a program source code and a request to the server, and the user can obtain its computational results. In this case, an application to execute a source code of a CAS is not required on the local machine.

An Example

As example, we demonstrate an exercise in numerical analysis.

1. A teacher creates the following document by this tool on a local machine.

The Lagrange polynomial of degree n is defined as

$$P_n(x) := \sum_{k=0}^{n} f(t_k) \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - t_i}{t_k - t_i}.$$

Let interpolation points be

$n = 10,$

$t_j = -1 + \frac{2}{n} j,\ j = 0, \cdots, n.$

For a function

$$f(x) := \frac{1}{25x^2 + 1},$$

observe a behavior of the polynomial by

plot $P_n(x)$ for $-1 \leq x \leq 1$

The above image is a hard copy of a screen of the tool.

2. This document is saved as a textual extended FR file on the server.

3. When a student connects to the server by a Web browser, the tool starts on a local computer. After selecting a menu to open the document located on the server, the document is displayed on a window.

4. A corresponding source code of a front-end language of a CAS is generated by a filter. Here is the output for Mathematica:

```
(* The Lagrange polynomial of degree n is defined as *)
P[x_,n_]:=Sum[f[t[k]] Product[(x-t[j])/(t[k]-t[j]),{j,0,k-
1}]
Product[(x-t[j])/(t[k]-t[j]),{j,k+1,n}],{k,0,n}];
(* Let interpolation points be *)
n = 10;
Do[t[j]=-1+2/n j,{j,0,n}];
(* For a function *)
f[x_]:=1/(25 x^2+1);
(* observe a behavior of the polynomial by *)
Plot[P[x,n],{x,-1,1}];
```

5. The student can obtain a computing result of the source code by running a CAS on the local machine. The student can also edit this document on a window of the tool. Various graphs are obtained by changing interpolation points or the function f(x) in the document, which helps a user's intuition.

In addition, the corresponding LaTeX source code can also be generated by a filter. We can use a collection of example documents on the WWW as teaching materials of mathematics.

5. Conclusions

In this paper, we proposed a front-end tool for mathematical computation and education. This front-end tool is developed in JAVA, and can be run over a network using a Web browser. The tool makes it possible to use common notations in the whole process of computing. Users can browse a document including mathematical expressions, and manipulate these expressions interactively.

The tool is based on simple representations of mathematical expressions, and can be easily adapted for various existing computational and educational systems.

Acknowledgments

The authors would like to thank M. Okeguchi and K. Nakatogawa for their assistance in developing JAVA programs. They would also like to thank D. Verschaeren for his helpful advice.

References

[DW90] Doleh, Y. and Wang, P.S., SUI: a system independent user interface for an integrated scientific computing environment, *Proc. International Symposium on Symbolic and Algebraic Computation*, 88-94 (1990).

[FYT98] Furukawa, A., Yagi, Y., and Takahashi, T., On the efficient use of computer algebra systems in mathematics education, *J. of IPSJ*, **39** (2), 122-127 (1998) (in Japanese).

[HCM] Calculus & Mathematica at http://www-cm.math.uiuc.edu/

[HIE] IES, Mathematical Education and Technology at http://www.ies.co.jp/math/

[HJE] Joetsu Univ. of Education, Dept. of Math at http://202.25.238.80/math/Math_homepage.html.

[HMA] Math Actives at http://archives.math.utk.edu/

[HPM] Univ. Plymouth, Center for Teaching Math at http://www.tech.plym.ac.uk/maths/.

[HWA] W3C Activity: Math at http://www.w3.org/Math/Activity.

[Kaj92] Kajler, N., CAS/PI: a portable and extensible interface for computer algebra systems, *Proc. ISSAC'92*, 376-386 (1992).

[KS98] Kajler, N. and Soiffer, N., A survey of user interfaces for computer algebra systems, J. Symbolic Computation, **25** (2), 127-159 (1998).

[LW95] Lee, H.J. and Wang, J.S., Design of a mathematical expression recognition system, *Proc. Third International Conference on Document Analysis and Recognition*, 1084-1087 (1995).

[Mar97] Marchioro, T.L., Web-based education in computational science and engineering, *IEEE Computational Science and Engineering*, April-June, 19-26 (1997).

[SZST96] Sakurai, T., Zhao, Y., Sugiura, H. and Torii, T., A user interface for natural mathematical notations, *Trans. Japan SIAM*, **6** (1), 147-157 (1996) (in Japanese).

[Soi95] Soiffer, N., Mathematical typesetting in Mathematica, *Proc. ISSAC'95*, 140-149 (1995).

[YW87] Young, D.A. and Wang, P.S., GI/S: a graphical user interface for symbolic computation systems, *J. Symbolic Computation*, **4**, 365-380 (1987).

[ZSST94] Zhao, Y., Sugiura, H., Torii, T. and Sakurai, T., A knowledge-based method for mathematical notations understanding, *Trans. Inform. Process. Soc. Japan*, **35** (11), 2366-2381 (1994).

[ZSST98] Zhao, Y., Sakurai, T., Sugiura, H. and Torii, T., Formalization and parsing of mathematical expressions for mathematical computation, to appear in *J. JSSAC* (1998).

Tetsuya Sakurai: sakurai@is.tsukuba.ac.jp, Institute of Information Sciences and Electronics, University of Tsukuba, Japan.

Yanjie Zhao: yzhao@cc.nias.ac.jp, Nagasaki Institute of Applied Science, Japan.

Hiroshi Sugiura: sugiura@nuie.nagoya-u.ac.jp, Department of Information Engineering, Nagoya University, Japan.

Tatsuo Torii, Department of Information Systems and Quantitative Sciences, Nanzan University, Japan.