LOCAL AND REMOTE USER INTERFACE FOR ELIMINO THROUGH OMEI

YONGWEI WU
Institute of Systems Science, Chinese Academy of Sciences, Beijing 100080, China.

WEIDONG LIAO
Institute for Computational Mathematics, Department of Computer Science, Kent State University, Kent, Ohio 44242, U.S.A.

DONGDAI LIN
State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China.

PAUL S. WANG
Institute for Computational Mathematics, Department of Computer Science, Kent State University, Kent, Ohio 44242, U.S.A.

ELIMINO is a new computer-mathematics research system developed to support Wu's method for computing characteristic sets of polynomials and other related operations. By implementing OMEI as its application programming interface, ELIMINO could make itself IAMC compliant, gain access to the graphical user interface Dragonfly locally and remotely, and provide interoperability with other OMEI compliant tools, such as editors and word processing systems, with minimal efforts.

1 Introduction

ELIMINO[5] is a new symbolic computation system being developed at the Institute of Systems Science of the Chinese Academy of Sciences. Capabilities of ELIMINO include manipulation of multi-precision numbers and polynomials, computation of characteristic set in Wu's method[9], polynomial equation solving, geometric theorem proving, etc..

To make ELIMINO easy to use and widely accessible, a good graphical user interface (GUI) is needed. Preferably, the same GUI will also make ELIMINO accessible remotely, via a LAN and/or the Internet.

Our approach is to apply the *Open Mathematical Engine Interface* (OMEI)[4], an application programming interface (API) specification for mathematical compute engines. OMEI enables tools and applications to use the same uniform interface to access any compliant mathematical engine. The concept of OMEI is similar to that of Microsoft's *Open Database Connectivity*

(ODBC) which has become the standard for PCs and LANs. With OMEI, distributed mathematical components, such as front ends, servers, and tools, can interoperate with different mathematical engines.

OMEI is used in the *Internet Accessible Mathematical Computation* (IAMC) framework[7] as a server-to-external-engine interface. The IAMC framework uses MCP (the mathematical computation protocol)[8] to connect users to remote mathematical engines across the Internet. IAMC already has an OMEI-ready server prototype (Starfish)[1], and a client prototype (Dragonfly)[2] supplying a nice GUI for back-end mathematical systems.

By adopting OMEI as an application interface, ELIMINO achieves the following results with minimal efforts.

- Using Dragonfly as a local GUI
- Using Dragonfly through IAMC as a remote GUI
- Ready to become an IAMC compliant compute engine through Starfish
- Interoperability with other OMEI compliant tools, such as editors and word processing systems.

We begin by briefly overviewing ELIMINO, IAMC, and OMEI. The C++ implementation of OMEI in ELIMINO is described. The way Dragonfly is interfaced to ELIMINO locally and remotely is discussed. Because ELIMINO is the first new symbolic computation system to adopt OMEI, the work tests the feasibility of the OMEI concept and design. A working system will be demonstrated in the conference.

2 ELIMINO

ELIMINO is a new computer-mathematics research system developed at the Mathematics Mechanization Research Center(MMRC), Institute of Systems Science, Chinese Academy of Sciences, as part of the "Mathematics Mechanization and its Applications" project. A long-standing goal at MMRC is to automate Wu's method independent of existing computer algebra systems. In ELIMINO, many different kinds of mathematical objects and data structures are provided. As an interactive system, ELIMINO is designed to focus on the implementation of Wu's method for researchers to perform sophisticated mathematical computations. It has very general capabilities for treating numbers, polynomials and characteristic sets[9].

To facilitate mathematical research, ELIMINO is kept open and flexible. From the user viewpoint, it consists of three parts:

- **Kernel part** is the soul of the system, it contains implementation of number system, polynomial manipulation system, characteristic set method. In fact, it is a powerful algebraic compute engine.
- **Applications** are packages or programs developed using the ELIMINO library. Examples include the polynomial system solver and the geometry theorem prover. A package may be build-in or loaded into ELIMINO on demand.
- **Front-end** is the interface between the system and users, it handles I/O between the user and the system.

From a software architecture viewpoint, ELIMINO has these five layers as shown in Figure 1:

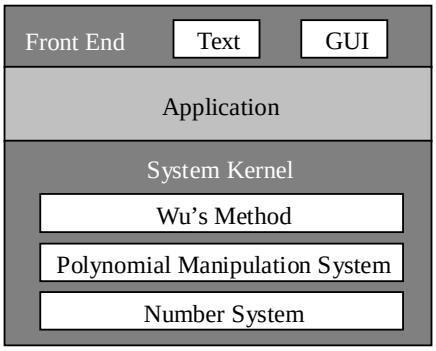

Figure 1. ELIMINO System Architecture

Each layer is an independent module, written in ANSI standard C++,providing a collection of closely-related and tightly organized services. A lower layer never calls functions in an upper layer.

The current ELIMINO front-end is text based. A good graphic user interface (GUI) can make ELIMINO easier to use. It would be even better if such a GUI can also make ELIMINO network accessible.

3 IAMC Framework

The IAMC framework is designed to make mathematical computing easily accessible and usable on the Web/Internet. Figure 2 shows the overall IAMC framework architecture.

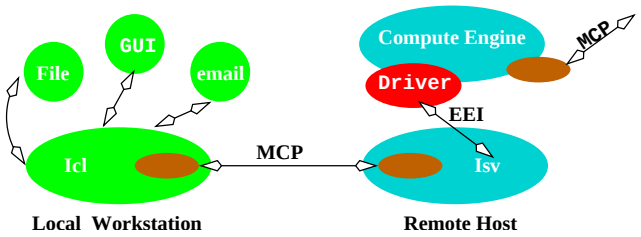

Figure 2. IAMC Architecture

The IAMC framework consists of these components:

1. IAMC client (Icl) — An end-user agent for accessing services provided by any IAMC server.
2. IAMC server (Isv) — A program to provide mathematical computation powers through the MCP protocol. An Isv may or may not employ an external compute engine to perform mathematical computations.
3. Protocol — IAMC clients and servers are connected by the Mathematical Computation Protocol[7]). MCP aims to be a simple and effective request-response protocol to support one-time transactions and interactive sessions.
4. Mathematical Data Encoding — Standard and user-defined mathematical data encodings can be used.
5. External Engine Interface (EEI) — A specification and API implementation for binding existing compute engines to IAMC servers.

4 Open Mathematical Engine Interface (OMEI)

Open Mathematical Engine Interface(OMEI) is an application programming interface(API) specification that aims to be an interface general enough to work for most mathematical compute engines. It specifies a set of function prototypes together with their syntax and semantics to give a unified programming level interface for heterogeneous mathematical engines. These prototypes supports all operations that are necessary for server-engine interaction, including connecting to/disconnecting from a compute engine, querying engine capabilities, creating and executing commands, etc..

As an attempt in standardizing programming interface for compute engines, OMEI can achieve several objectives:

- **Make Compute Engine IAMC-Capable**
  OMEI and IAMC framework together can help making compute engine internet accessible. The IAMC framework follows a 3-tier architecture that consists of IAMC client, IAMC server, and back-end compute engine. The OMEI can be used to connect IAMC server and back end engines. Once an OMEI driver of a compute engine has been developed according to OMEI specification, the IAMC server can be configured to dynamically locate and load this OMEI driver, and so will be able to forward the computational requests to the engine and results to remote IAMC clients.

- **Application Portability**
  An application or user interface developed using any OMEI-compatible interface would be portable among different compute engines, as long as those compute engines have OMEI drivers available. That is, the compute engine and its user interface or applications can be developed separately with the help of OMEI specification.

- **Heterogeneous Integrated Compute Engine**
  Since an application can access multiple engines by loading multiple OMEI drivers, an integrated compute engine with combined capabilities can be accomplished under OMEI programming model. A parallel/distributed problem solving environment would be easily achievable over the OMEI programming paradigm.

- **3-tier/multi-tier mathematical system**
  This is very similar to IAMC framework that is MCP oriented. It is proven that the multi-tier architecture can achieve better performance and scalability over traditional client/server architecture. Besides IAMC, OMEI could also be used in other non-MCP distributed mathematical systems. This architecture is also useful for developing Web or other thin client based computational/educational systems.

5 OMEI Implementation in ELIMINO

There are basically two approaches to the implementation of OMEI for ELIMINO compute engine. The most straightforward approach is to use interprocess communication(IPC) to access ELIMINO. ELIMINO reads from *standard input* and writes to *standard output*. The OMEI driver connects to ELIMINO with a two-way pipe, sending computational requests into the pipe and reading computational results from the pipe. The IPC approach is simple and straightforward, but lacks flexibility and user control.

As current ELIMINO comes with an API of a set of ad-hoc C++ functions, we have adopted the same-process approach to implement OMEI for ELIMINO (see Figure 3(right)). This approach is more flexible and allows

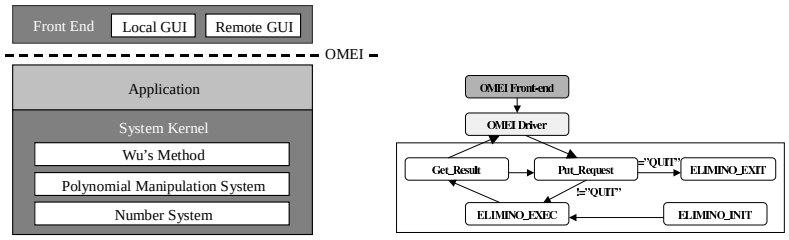

Figure 3. OMEI via Fuction Calls

users to set up the computation environment, input/output mode, command encoding format, and resource allocation. In-process data transfer is also more reliable than IPC.

In our same-process approach, the OMEI is implemented above the application layer of ELIMINO, but below the front-end layer (see Figure 3(left)), providing standard programming interface for front-end layer to access computation powers of the ELIMINO compute engine.

Our implementation of OMEI drivers for ELIMINO is a standard C++ library. It contains totally 18 functions for connecting to/disconnecting from ELIMINO, querying ELIMINO's capabilities, creating computational commands and executing these commands, each of them is programmed as the function prototypes specified in the OMEI specification. The functions `Put_Request`,`Elim_exec` and `Get_Result` are added to ELIMINO for OMEI to send requests, execute commands and obtain results. The three functions access the ELIMINO API to perform computations.

As an example, let's see how to perform the simple computation *gcd((x+y+z)*(7*x-8*z),(7*x-8*z)*(37*z-87*y*x))*. in ELIMINO via OMEI.

```
#include "ELIMINO.h"
char *request="gcd((x+y)*(7*x-z),(7*x-z));";   // (1)
char *result;
int main() {
  ELIMINO_INIT();                              // (2)
  Put_Request(char * request);                 // (3)
  ELIMINO_EXEC();                              // (4)
  Get_Result(char * result);                   // (5)
  ELIMINO_EXIT();                              // (6)
```

```
    return 0;
}
```

First, two variables, `request` and `result`, are allocated to store the computational request from IAMC servers or other compliant tools and computational result or error information. Second, ELIMINO is initialized as necessary by setting up computation environment and input/ouput modes (line 2). The request is then sent to (line 3) and evaluated by the ELIMINO compute engine (line 4). Finally the result is read into the buffering variable `result` and the ELIMINO engine quits.

6 Applying OMEI to ELIMINO

Dragonfly is an IAMC client prototype and *Starfish* an IAMC server prototype which are implemented in Java. The Dragonfly-Starfish approach makes interactive mathematical computations accessible and interoperable over the Internet. By adopting OMEI as an application interface, ELIMINO can easily use *Dragonfly* as its local interface (see Figure 4 (a)) or remote user interface (see Figure 4 (b)).

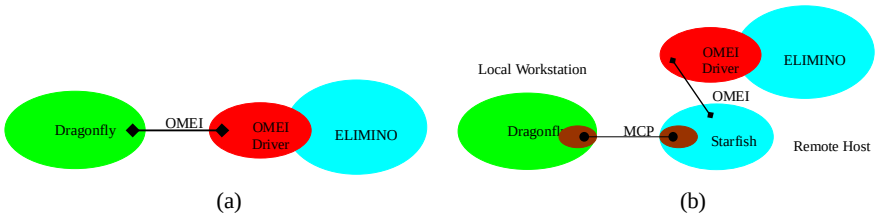

Figure 4. Use Dragonfly as local interface (a) or remote user interface (b)

A graphical user interface like Dragonfly can not only help beginning users to get started in ELIMINO quickly, but also provide potentials for adding into ELIMINO more functionalities, such as mathematical graphing and plotting. To interface Dragonfly to ELIMINO as a local GUI, a Java OMEI driver has to be developed because Dragonfly is developed in Java. In other words, Dragonfly has to load this Java OMEI driver to interact with ELIMINO. The Java OMEI driver could be thought of as a Java Wrapper for OMEI function calls and it is implemented by using Java Native Interface (JNI). Figure 5 shows the user interface of Dragonfly as the local GUI for ELIMINO.

Using Dragonfly through IAMC as a remote GUI is another story. It is possible to turn the ELIMINO compute engine into an IAMC server by

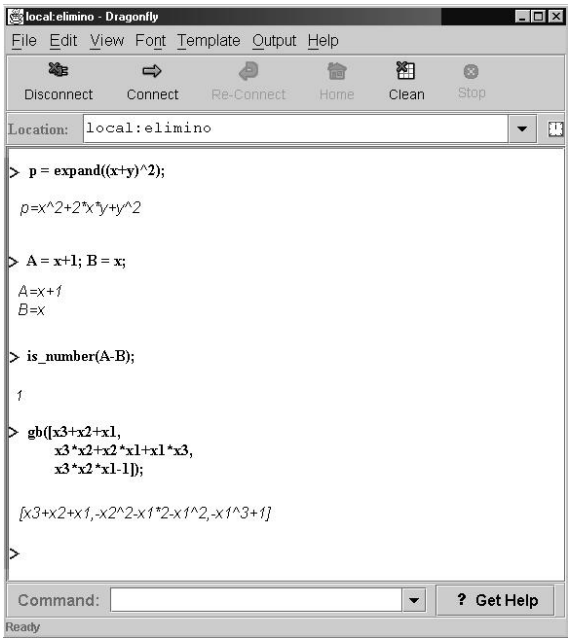

Figure 5. Dragonfly as a local GUI

integrating MCP library into it. Nevertheless, with the OMEI interface, it is much easier to integrate ELIMINO into an existing IAMC server, such as our IAMC server prototype *Starfish*. *Starfish* is configured to dynamically locate and load the OMEI driver. With the loaded driver, *Starfish* will be able to forward the computational requests to ELIMINO and results to *Dragonfly*, or any other IAMC clients. Figure 6 shows the user interface of *Dragonfly* as the remote GUI for ELIMINO.

Turning ELIMINO into an IAMC server directly (by integrating MCP library) or indirectly (by using an IAMC server like *Starfish*) also has some other potentials. This will make ELIMINO an IAMC compliant engine and it will be able to serve any MCP requests from Internet. For example, a Web-based mathematical education system may send MCP requests to ELIMINO to get answers for some algebra questions, and a distributed crypt algorithm may issue MCP requests to ELIMINO to get some big prime numbers.

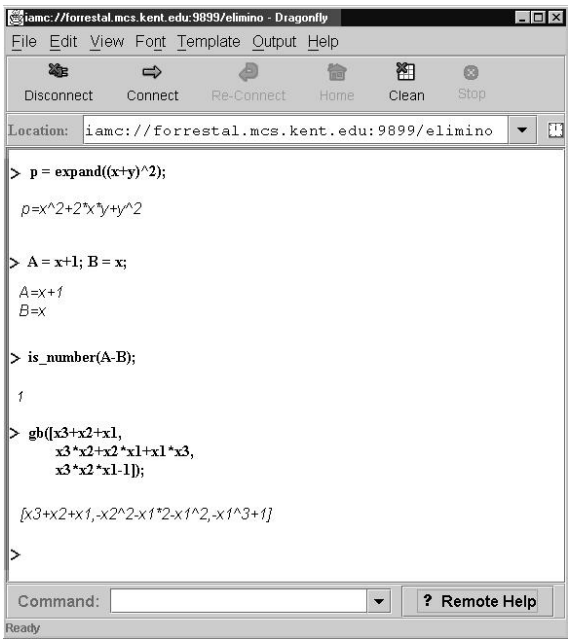

Figure 6. Dragonfly as a remote GUI

7 Conclusion and Future Work

The OMEI interface entitles several new features for ELIMINO, a new computer-mathematics research system, with minimal efforts. An existing graphical user interface, *Dragonfly*, could be used to access ELIMINO locally and remotely. As an IAMC compliant compute engine, ELIMINO can be used as a compute server for Internet based mathematical computational and education systems. Moreover, OMEI also gains interoperability with other OMEI compliant tools, such as scientific editors and word processing systems.

We will integrate XMEC [10], an eXtensible Mathematical Encoding Converter into Java OMEI driver for ELIMINO engine so that the engine could accept input and produce output encoding in standard encoding formats, such as MathML. When this work completes, users will be able to use Dragonfly or other tools to access ELIMINO engine to evaluate MathML code and generate MathML output. The ELIMINO compute engine will gain better interoperability in distributed and Internet based mathematical computation

and education environment.

Acknowledgements

Work reported herein has been supported in part by the Chinese National 973 Project NKBRSF G1998030609, Chinese 863 Project 2001AA144030, the USA National Science Foundation under Grant CCR-9721343 and INT-9722919, and the Ohio Board of Regents Computer Science Enhancement Funds.

References

1. LIAO, W. and WANG, P. S. *Building IAMC: A Layered Approach.* Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00). pp. 1509-1516.
2. LIAO, W. and WANG, P. S. *Dragonfly: A Java-based IAMC Client Prototype.* Lecture Notes on Computing Vol.8. (Proceedings of ASCM 2000.) World Scientific Press, pp. 281-290.
3. LIAO, W. and WANG, P. S. *Specification of OMEI: Open Mathematical Engine Interface.* ICM Technical Report. 2001. http://icm.mcs.kent.edu/reports/index.html.
4. LIAO W., LIN D. and WANG P. S. *OMEI: Open Mathematical Engine Interface.* Proceedings of ASCM'2001, pp 83-91, Matsuyama, Japan, September 26-28, 2001. Lecture Notes Series on Computing Vol. 9, World Scientific.
5. LIN D., LIU J. and LIU Z. *Mathematical Research Software: ELIMINO.* Proceedings of ASCM'98. pp. 107 - 116, Lanzhou Univ., China, 1998.
6. WANG, P. S. *Internet Accessible Mathematical Computation.* In the 3rd Asian Symp. on Computer Mathematics (ASCM'98), pp 1-13, Lanzhou Univ., China, 1998.
7. WANG, P. S., GRAY, S., KAJLER N., LIN D., LIAO W. etc. *IAMC Architecture and Prototyping: A Progress Report.* Proceedings of ACM ISSAC'01, University of Western Ontario, London, Ontario, Canada, July 22-25, 2001.
8. WANG, P. S. *Design and Protocol for Internet Accessible Mathematical Computation.* In Proc. ISSAC'99 (1999), ACM Press, pp. 291-298.
9. WU, W. T. *Basic Principle of Mechanical Theorem Proving in Elementary Geometries,* J. Syst. Sci. Math. Sci. 4, 207-235 (1984).
10. ZOU, X. *XMEC: An Extensible Mathematical Encoding Converter.* ICM Technical Report. 2001. http://icm.mcs.kent.edu/reports/index.html.