Internet Accessible Mathematical Computation

A Progress Report

Paul S. Wang
Institute for Computational Mathematics
Kent State University
http://horse.mcs.kent.edu/~pwang

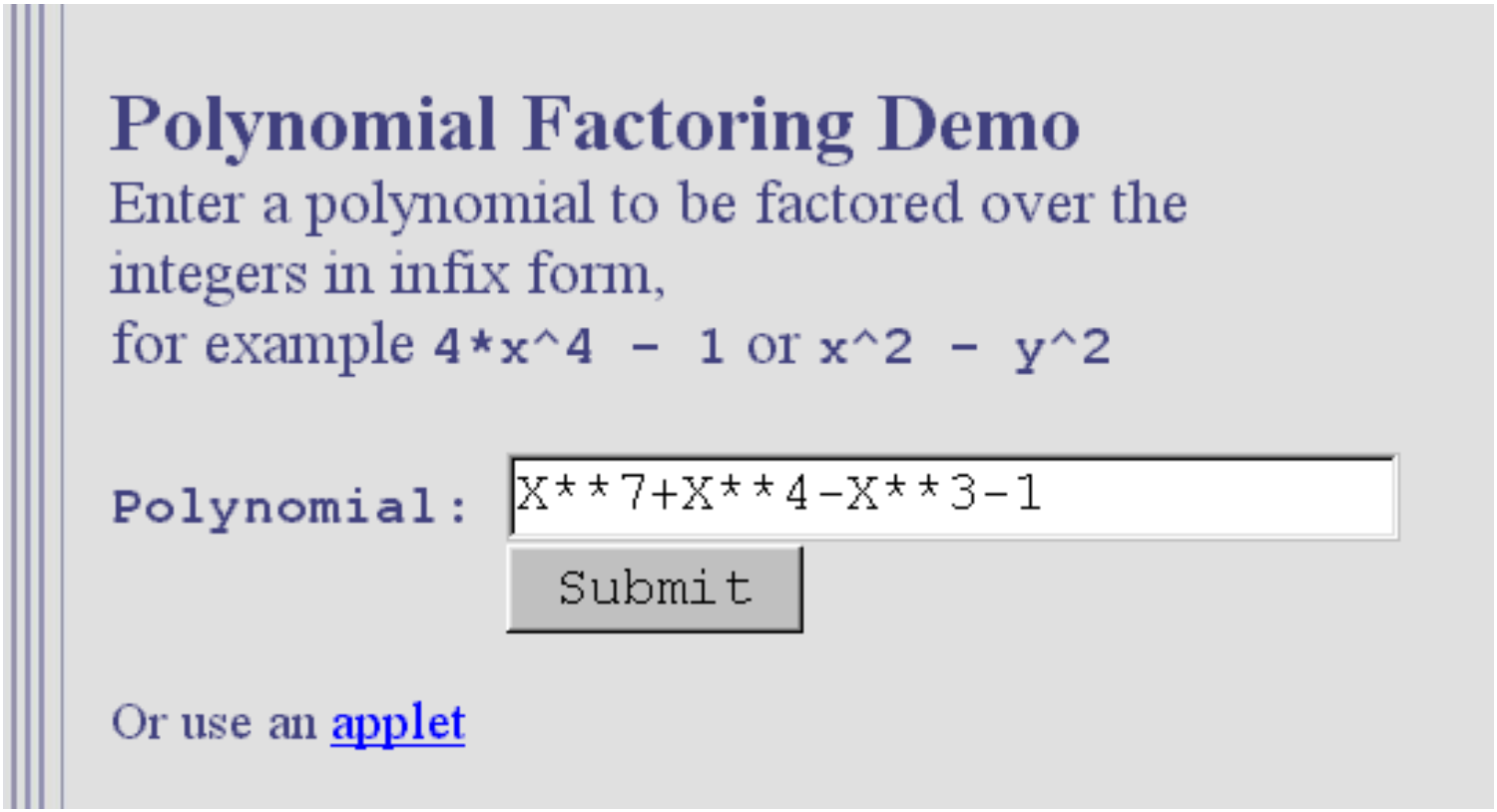
Contents

- Math Communication on the Web/Internet
- Some Examples
- Standards for Math Encoding
- Math Computation Power, the IAMC Approach
- Applications and Architecture
- Client and Server Design
- Server Interface to Compute Engine
- MCP Protocol
- Further Work

Examples Today

- Table of integrals — mathematical database at the U. C. Berkeley.
- Live computation demos — derivatives, polynomial factoring, Fortran code generation, curve/surface plotting on `SymbolicNet` at ICM/Kent.
- *Techexplorer* — a Web browser plug-in that dynamically formats and displays documents containing scientific and mathematical expressions coded in TEX/LATEX by IBM Watson Research Center.
- NetSolve — a system to make numerical computation packages available to Web through a Java Applet by a joint project between the U. of Tennessee and the Oak Ridge National Laboratory.

ICM Demos


Polynomial Factoring Demo
Enter a polynomial to be factored over the integers in infix form,
for example 4*x^4 - 1 or x^2 - y^2
Polynomial:
X**7+X**4-X**3-1
Submit
Or use an applet

Answer Computed:

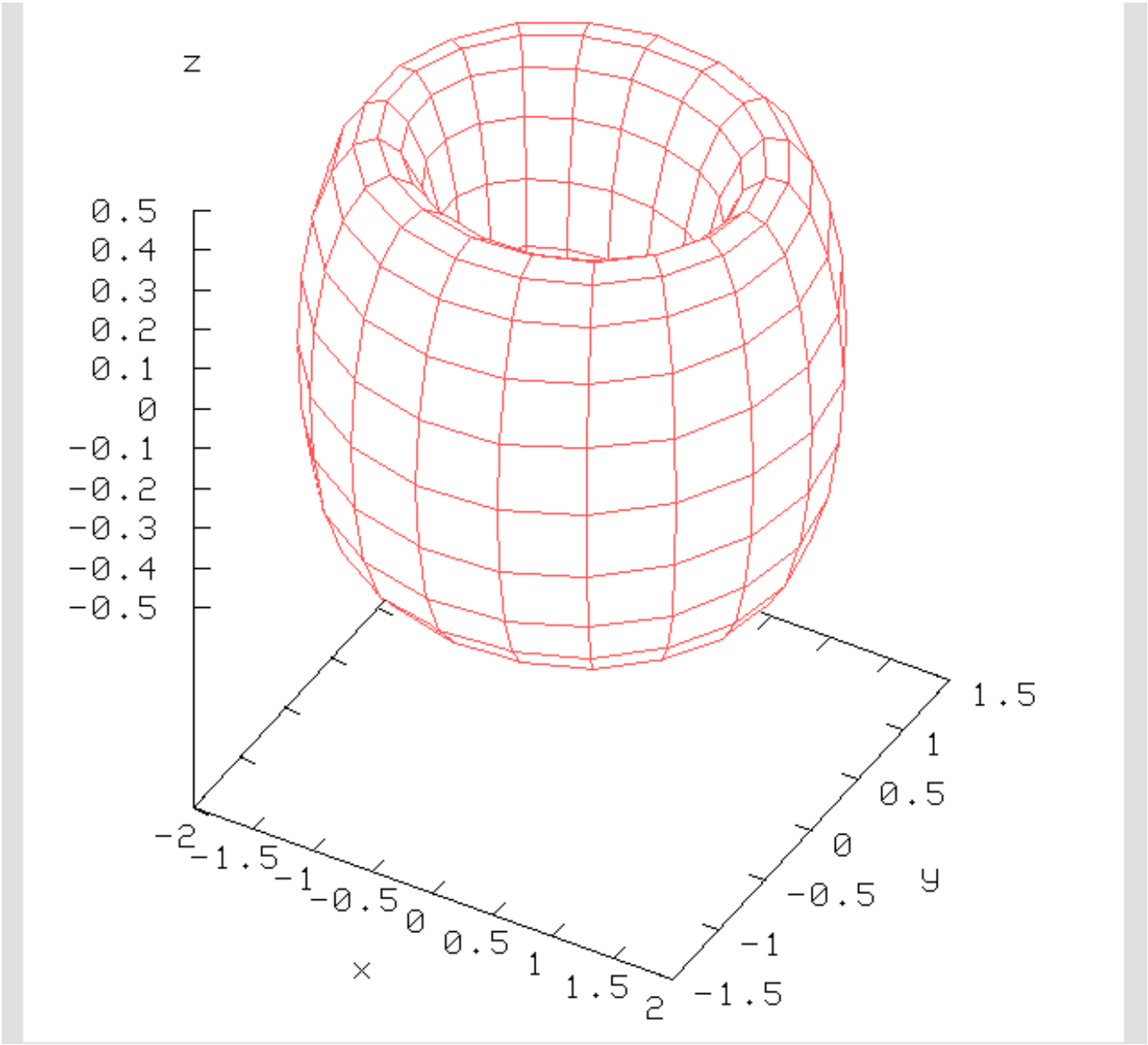
```
(C3) factor(X**7+X**4-X**3-1);
                                2   2         2
(D3)          (X - 1) (X + 1)  (X  + 1) (X  - X + 1)
```

Parametric Surface Plot Demo

Enter u-range, v-range and functions x(u,v), y(u,v), z(u,v) to plot for example:

x(u,v):	1.5+0.5*cos(v))*cos(u)
y(u,v):	(3/2+cos(v)/2)*sin(u)
z(u,v):	sin(v)/2
range of u:	0:2*pi
range of v:	0:2*pi

Submit


z
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x
1.5
1
0.5
0
-0.5
-1
-1.5
y

LaTeX Conversion Demo

Enter a mathematical expression to convert to LaTeX
in the form `sin(sqrt(x)/2)^y`
or `matrix([a,b],[c,d])`

Latex: matrix([a^2, b/3],[c+2, 15*d])

Submit

LaTeX Code Computed:

```
$$\pmatrix{A^{2}&{B \over 3}&\cr
           C+2&15\>D&\cr }$$
```

Here is the LaTex file containing the computed codes.

F77 Generation Demo

Enter a mathematical expression to convert to F77, for example

```
matrix([sin(k/7),5/2],[y+10,cos(-y)])
```

f77: matrix([sin(k/7),5/2],[y+10,cos(-y)])

Submit

F77 Codes Generated:

```
VAR(1,1) = SIN(K/7.0)
VAR(1,2) = 5.0/2.0
VAR(2,1) = Y+10
VAR(2,2) = COS(Y)
```

Here is the F77 file containing the generated codes.

Representation Standards

- MathML — a language for markup of mathematical expressions by a group at W3 consortium. (also the Amaya browser)
- OpenMath — a char-based math expression encoding format by the OpenMath group.
- MP — a binary mathematical expression encoding format and transfer protocol by the MP group.

Flexible Technical Content Types

MathML

$$x^2 + 4x + 4 = 0$$

```
<mrow>
  <msup> <mi>x</mi> <mn>2</mn> </msup>
  <mo>+</mo> <mn>4</mn> <mi>x</mi>
  <mo>+</mo> <mn>4</mn>
  <mo>=</mo> <mn>0</mn>
</mrow>
```

Content encoding of the second derivative $\frac{d^2}{dx^2}\ f(x)$

```
<apply><diff/>
  <apply><fn> f </fn>
    <ci> x </ci>
  </apply>
  <bvar>   <ci> x <ci> </bvar>
  <degree> <cn> 2 </cn> </degree>
</apply>
```

MP Format

- Binary parse tree data encoding
- Annotations Each tree node may be *annotated* with supplementary information.
- Optimizations for reduced data size
- Dictionaries for semantics

MP Dictionary Entry

```
<ConstDef>
  <DefName> Pi </DefName>
  <DefTag>  3  </DefTag>
  <Description> Circumference/diameter of circle.
  </Description>
  <CMP> 3.14159265358979323 85, approximation
        to 20 digits </CMP>
</ConstDef>
```

MP Encoding Example

$(f := x \rightarrow x * *3 - 1)_{\text{(source maple)}}$

```
op    1  :=     2    (1 annot 2 args)
src str  0  maple    (annot)
id    0  f          (arg 1)
op    0  ->     2    (arg 2)
id    0  x
op    0  -      2
op    0  **     2
id    0  x
int   0  3
int   0  1
```

MathML Code Generation

Example expressions are:

```
a^2+3*b+sqrt(5)          %e^(%pi+%i*theta)
binomial(a-b,c)          diff(g(x),x,2)
sum(i^2,i,a,b)           matrix([a^3,b],[x,y/2])
```

Expression: diff(g(x),x,2)

Submit

Presenation Code Generated

```
<math> <mrow>
   <mfrac><msup><mo> &dd;</mo><mn>2</mn></msup>
   <mrow>
      <mo> &dd; </mo><mo>&InvisibleTimes;</mo>
   <msup><mi>X</mi><mn>2</mn></msup>
   </mrow></mfrac>
      <mo>&InvisibleTimes;</mo>
      <mi>G</mi><mo>&ApplyFunction;</mo>
   <mrow>
      <mo>(</mo><mi>X</mi><mo>)</mo>
   </mrow>
</mrow> </math>
```

Content Codes Generated

```
<math>
<apply> <diff/>
      <apply> <fn>G</fn><ci>X</ci> </apply>
      <bvar><ci>X</ci>
    <degree><cn type="integer">2</cn></degree>
      </bvar>
</apply>
</math>
```

What is IAMC

Distributed *Internet Accessible Mathematical Computaion* system aims to

- Make math-oriented data and services easily and widely accessible on the Internet – directly, via the Web, and by email
- Support interactive use of user-designated remote *compute servers* almost as if they were local programs
- Provide effective and efficient communication of mathematical data over the Internet
- Allow exchange and further processing of computational results among different compute servers

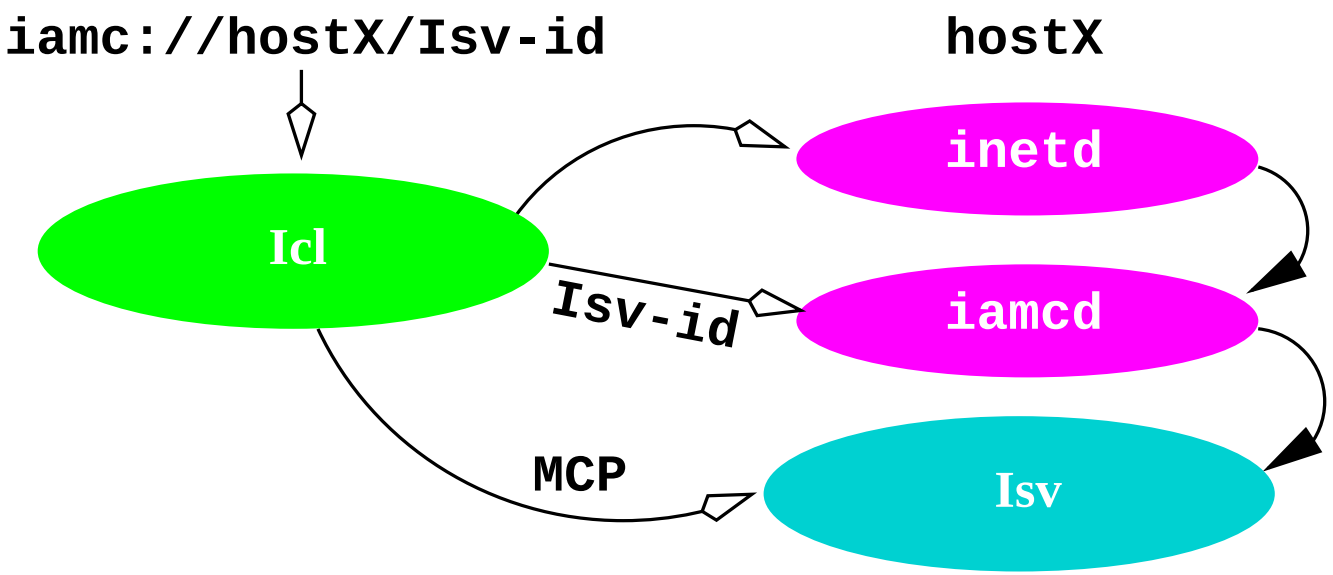
The IAMC Approach

- Each *IAMC server* (Isv) provides a specific computational service and has a URL in the form

  `iamc://`*hostname:port/server-id*

- An Isv can perform the computational tasks either directly or through a separate compute engine.
- An end user accesses IAMC through an *IAMC client* (Icl) supporting interactive computations.
- Client-server communication uses the *Mathematical Computation Protocol* (MCP) designed specifically for the purpose.
- IAMC services available by direct connection, via Web, or email.

IAMC Setup



IAMC Applications

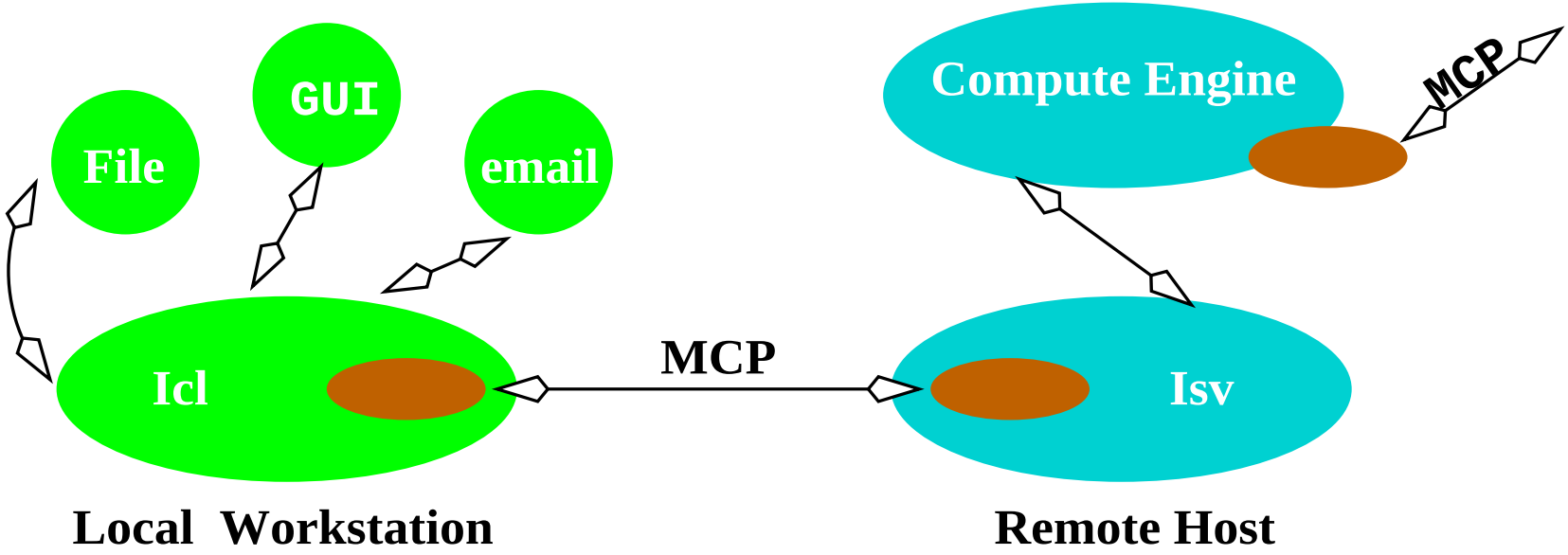
- Making available research and experimental computing systems in mathematics, science, and engineering
- Use in mathematical education and distance learning
- Access of remote scientific databases
- Making parallel/super computing more accessible
- Computing via NetPC for high school or occasional users

IAMC Architecture

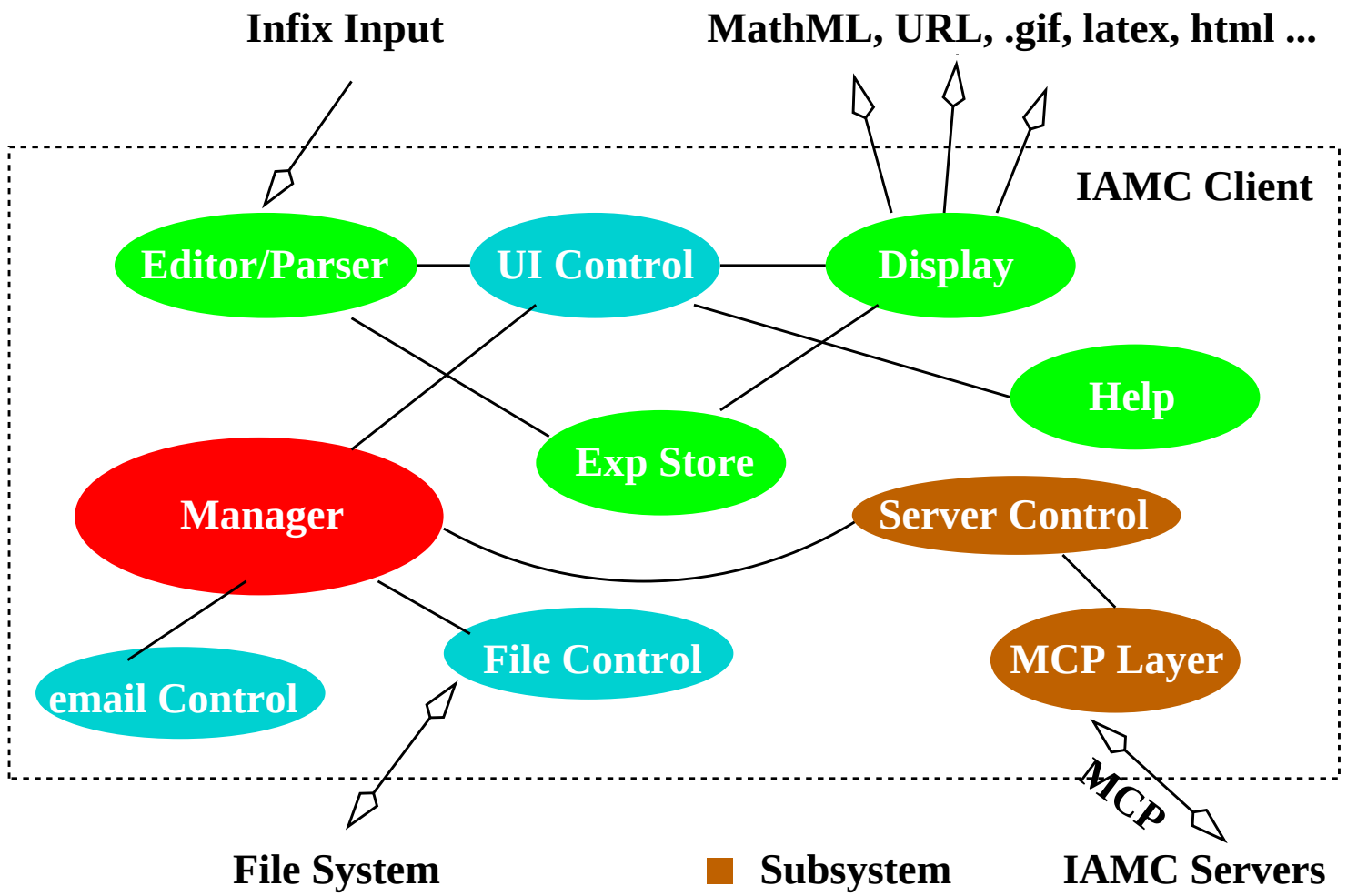
The IAMC system consists of the following components:

- Icl
- IAMC daemon (`iamcd`)
- Isv
- MCP
- Mathematical Data Encoding
- Compute engine

IAMC Architecture Overview



IAMC Client


Infix Input
MathML, URL, .gif, latex, html ...
IAMC Client
Editor/Parser
UI Control
Display
Help
Exp Store
Manager
Server Control
email Control
File Control
MCP Layer
MCP
File System
Subsystem
IAMC Servers

Integration Command Template

An integration command template may display, together with textual explanations,

$$\int_a^b \; f(x)\, dx$$

followed by a dialogue box for entering an integration command.

Example: `integrate(sin(x),x,0,pi/2)`

integrate(, integrand f(x)

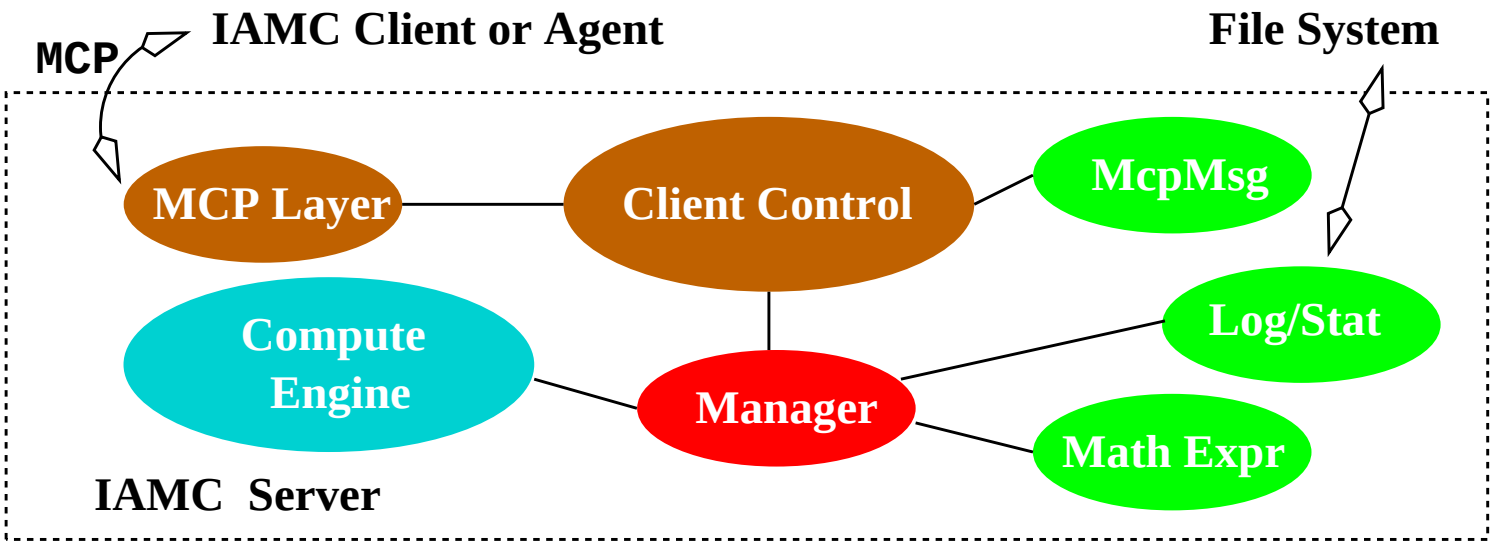
, variable x

, lower limit a

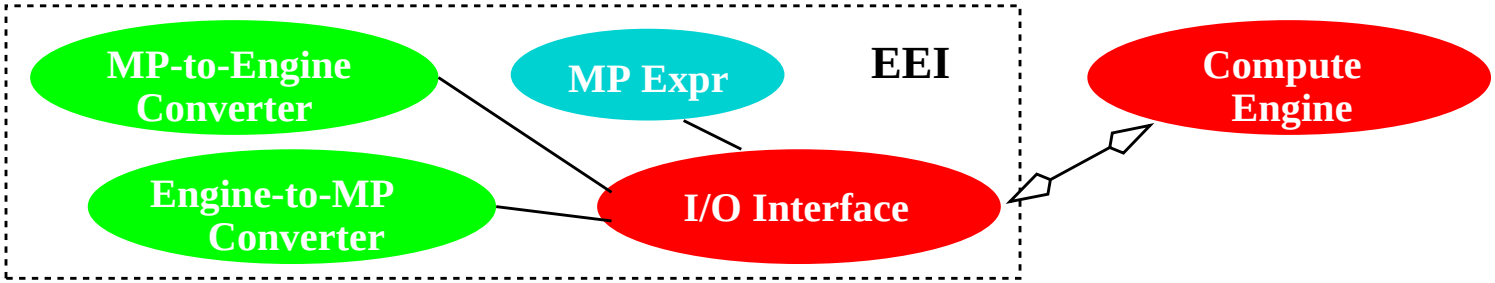
) upper limit b

Enter Command

IAMC Server


MCP
IAMC Client or Agent
File System
MCP Layer
Client Control
McpMsg
Compute Engine
Log/Stat
Manager
Math Expr
IAMC Server

Server Interface to Compute Engine



MCP Protocol Design Considerations

- Meeting client-to-server and server-to-client requirements
- Supporting various data-transfer and data compression encodings
- Allowing different mathematical representation formats
- Employing two-way, sequenced, reliable connection for computation sessions
- Assuming peer-to-peer interactions, no multiple client connections to the same server
- Handling computation, dialog, and control requests

MCP Requirements

- An Icl needs to send/receive control requests and responses; send computation commands; receive computation results in various encoding forms; issue commands synchronously (waiting for result before next command) and asynchronously (no waiting); receive results synchronously and asynchronously; abort an on-going computation; handle dialog requests.
- An Isv needs to send/receive control requests and responses; receive synchronous and asynchronous commands; send computed results in various formats (MathML, MP, OpenMath, GIF, PDF, ...); use different transfer encodings; allow various data compression methods; send back results; query the end-user for information.

MCP Message Format

- Modeled after HTTP but is session oriented
- A MCP message is either a request or a response
- *header* and *body*
- Each header entry is a *key-value pair* on one line
- The first line of an MCP message is one of

  `Request` *Class seqNo*

  `Response` *Class seqNo statusCode* [*statusString*]

MCP Message Classes

- `Computation` — A class of server-side operations to perform application supported computations;
- `Dialog` — A class of client-side methods to solicit information from the end user;
- `Control` — Both client and server have their own control class to supply non-computation methods for the control and management of the MCP session;
- `Initialization` — Both client and server have their own initialization class for setting up session parameters after client-server connection.

Computation Headers

- `Method:` *name*
- `Mode: sync` (or `async`, default is `sync`)
- `Send-result: yes` (or `no`, default is `yes`)
- Methods include `command`, `commandString`, `help`, and `template`.

Sample MCP Communication Scenarios

Initialization

```
    Request Initialization C1
    Method: setup
    Version: MCP/1.0
    User-agent: MathBrowser
    Accept: application/x-math-MP,text/MathML,
      application/x-math-OpenMath,text/MathML,
      image/GIF,text/HTML,application/PDF

Response Initialization C1 100 OK
```

Initialization Continued

```
Request Initialization S1
Method: setup
Version: MCP/1.0
Server-name: PolyFactor
Greeting: Performs univariate and multivariate
  polynomial factoring over the integers
CanDo: factor,expand,ratsimp

    Response Initialization S1 100 OK
```

Computation Request

```
    Request Computation C2
    Method: commandString
    Send-result: no
    Content-type: text/plain
    Content-length: ...

    p : 4*xˆ2-1

Response Computation C2 100 OK
```

Next Computation Request

```
    Request Computation C3
    Purpose: commandString
    Content-type: text/plain
    Content-length: ...

    factor(p)

Response Computation C3 100 OK
Content-type: application/x-math-MP
Content-length: 26

<body contains (2*x + 1)*(2*x - 1) in MP format>
```

Terminating

```
    Request Control C23
    Method: disconnect

Response Control C23 100 OK
```

Querying the End User

```
Request Dialog S7
Method: formQuery
Content-type: text/HTML
Content-length: 145

<HTML form for user>

    Response Dialog S7 100 OK
    Content-type: application/x-www-form-urlencoded
    Content-length: ...

    <name=value pairs separated by &>
```

Help

```
Request Computation C37
Method: help
Topic: integration
```

Help Response

```
Response Computation C37 100 OK
Content-type: text/plain
Content-length:  245

integrate(f(x),x, a, b) computes the exact definite
integral of f(x) from a to b; integrate(f(x),x)
computes the antiderivative of f(x);
romberg(f(x), x, a, b, eps) computes the numerical
quadrature of f(x) from a to b with accuracy eps.
```

Command Template

```
Request Computation C67
Method: template
Command-name: limit
```

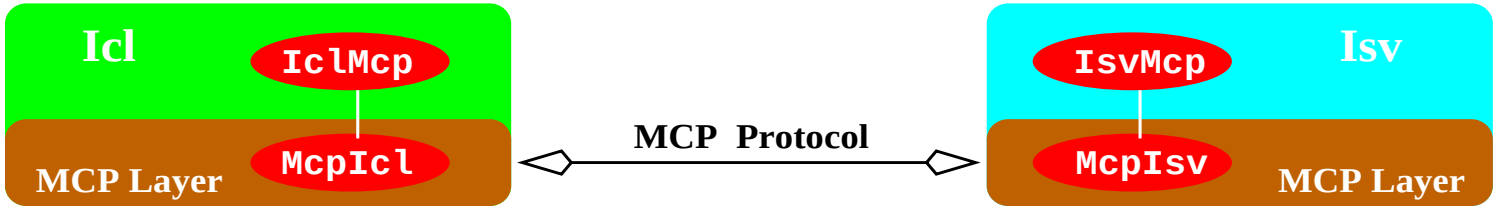
```
Response Computation C67 100 OK
Content-type: application/x-mcp-CommandTemplate
Content-length:  ...

command: limit(f,x,pt)
effect: returns the limit of f as x approaches pt
example: limit(sin(x)/x,x,0)
example: limit((1+1/x)^x,x,inf)
arg: f-an expression involving the variable x
arg: x-an identifier representing a variable
arg: pt-a constant expression not involving x,
        or INF (+infinity), MINF (-infinity),
        INFINITY (infinity)
```

Proof-of-concept System

- David Wei Wu's Master thesis
- Simple client (command based, text only) in Java
- Simple server in Java, front-ending MAXIMA
- MP-1.1.3 used for data transfer
- Direct socket networking

MCP Layer Interface



MCP-Isv Interface

The `McpIsv` class supports these methods:

- `public McpIsv(IsvMcp svr)`
  Constructor. Initializes an `McpIsv` object to receive data from the Icl via standard input and send data to the Icl via standard output. The argument `svr` is the peer object.
- `public McpIsv(InputStream in, OutputStream out, IsvMcp svr)`
  Constructor. Initializes an `McpIsv` object to perform I/O with the Icl via the given streams. The argument `svr` is the peer object.
- `public boolean putResult(McpMsg m)`
  Sends the computational result packed in the message `m` to the Icl. Returns `false` when failed.

- `public void ready(Boolean flag)`
  Indicates to the Icl that the server is or is not ready for additional work.

- `public McpMsg dialog(McpMsg m)`
  Sends the dialog request `m` for end user to the Icl and returns the information obtained.

- `public void terminate()`
  Indicates to the Icl that server is finished and disconnecting.

- `public boolean pingclient()`
  Requests client status. Returns `true` if client is ready and `false` if client is busy. Assumes client is dead after a preset timeout interval.

The `IsvMcp` class supports these methods:

- `public void command(McpMsg m)`
  Performs the computation `m` (synch or asynch). Result returned via a call to `putResult(McpMsg r)` later.
- `public void perform(McpMsg m)`
  Performs the computation `m` without returning any result.
- `public void abort(int n)`
  Aborts command `n`.
- `public void quit()`
  Terminates computation session.
- `public void reset()`
  Resets the Isv to its initial state aborting all on-going computations.

MCP-Icl Interface

The `McpIcl` class supports these methods:

- `public McpIcl(String url, IclMcp cl)`
  Constructor. Initializes an `McpIcl` object to connect to the Isv given by the `url` and to cooperate with the specified peer object `cl`.
- `public McpMsg syncCommand(McpMsg cmd)`
  Sends the command `cmd` to the Isv in synchronous mode. Returns the computational result received.
- `public boolean asyncCommand(McpMsg cmd)`
  Sends the command `cmd` to the server in asynchronous mode. Returns `false` when failed. The result produced by this method will be received via a call to the `result` method of the peer `IclMcp` object.

- `void abort(int n)`
  Sends a control message to the sever to abort the prior computation request with sequence number `n`.
- `void terminate()`
  Disconnects from server.
- `public boolean pingserver()`
  Requests server status. Returns `true` if sever is ready and `false` if server is busy. Assumes server is dead after a preset timeout interval.

The `IclMcp` class supports these methods:

- `McpMsg queryUser(McpMsg q)`
  Processes the given query (dialog) to the user and returns data obtained from the end-user.

- `boolean putAsyncResult(McpMsg r)`
  Delivers the computational result packed in the message `r`. This method is called by the `McpIcl` object to deliver a result for an earlier asynchronous command.

Further Work

- Design refinements
- Implementation of prototype Icl and Isv
- Building flexible and reusable EEI for Isv
- Building or adapting a GUI for the client
- Full specification of MCP
- A Java class library implementation for MCP
- Redesign and re-implement MP in Java
- Building MathML/MP and other format converters
- Establishing various demonstration IAMC services