OpenMath – a Mathematics Lingua Franca

Peter Horn
formerly University of Kassel
peter.horn@provideal.net
What’s the meaning?

- `diff(sin(x), x)`
- `D[Sin[x], x]`
- `\frac{\text{d}}{\text{d}x}\sin(x)`
- `derive` sinus of `x` w.r.t. `x`
- `\frac{d}{dx}\sin(x)`

Even worse: What’s `derive`, what’s `x`, what’s `sinus`?
Representing Mathematics

Every mathematical software has two solutions:

- Internal representation (generally object tree)
- Interface representation (in-/output syntax)

Generally, these representations may vary!

Even worse: semantics may vary!

General question is:
How to represent mathematical semantics (on a computer)
Mission Objective

Create a Lingua Franca for Mathematics that is

- Extensible
- Machine-processable
- Universal
- Open (i.e., not barred by patents)
- Effective and Efficient
OpenMath is a language for representing mathematical objects focused on semantics (not visualisation as MathML). It allows different representations: XML, Binary, and more. OpenMath is maintained by the OpenMath Society. All standard documents can be found at www.openmath.org...
<table>
<thead>
<tr>
<th>Language Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMA</td>
<td>Apply</td>
</tr>
<tr>
<td>OMB</td>
<td>Binary</td>
</tr>
<tr>
<td>OMBIND</td>
<td>Binding</td>
</tr>
<tr>
<td>OME</td>
<td>Error</td>
</tr>
<tr>
<td>OMF</td>
<td>Float</td>
</tr>
<tr>
<td>OMFOREIGN</td>
<td>Foreign</td>
</tr>
<tr>
<td>OMI</td>
<td>Integer</td>
</tr>
<tr>
<td>OMATTR</td>
<td>Attribution</td>
</tr>
<tr>
<td>OMOBJ</td>
<td>Object</td>
</tr>
<tr>
<td>OMR</td>
<td>Reference</td>
</tr>
<tr>
<td>OMSTR</td>
<td>String</td>
</tr>
<tr>
<td>OMS</td>
<td>Symbol</td>
</tr>
<tr>
<td>OMV</td>
<td>Variable</td>
</tr>
</tbody>
</table>

What do they mean?
Atomic Elements

- Integer: Arbitrary high precision integers
- Floats: IEEE doubles
- Strings: Character Sequences
- Symbols: see next slide
- Variables: Variables

Just technical stuff – not much semantics so far...
Symbols

- Consist of a `cd` and a `name` attribute
- are organized in “Content Dictionaries”, CDs
- in the CD, the mathematical *meaning* of the symbol is specified

That’s where the semantics comes from!
OpenMath Content Dictionary: arith1

plus

Role:
application

Description:
The symbol representing an n-ary commutative function plus.

Commented Mathematical property (CMP):
for all a,b \ a + b = b + a

Formal Mathematical property (FMP):
\forall \ a, b . a + b = b + a

Signatures:
sts
Role:
constant

Description:

A symbol to convey the notion of pi, approximately 3.142. The ratio of the circumference of a circle to its diameter.

Commented Mathematical property (CMP):
ipi = 4 * the sum as j ranges from 0 to infinity of ((1/(4j+1)))-(1/(4j+3))

Formal Mathematical property (FMP):

\[ \pi = 4 \sum_{j=0}^{\infty} \frac{1}{4j+1} - \frac{1}{4j+3} \]

Example:
3.142 = The decimal approximation to 3 significant places of pi

Signatures:
sts
<CDDefinition>
  <Name> interval </Name>
  <Role>application</Role>
  <Description>
  A symbol to denote a continuous 1-dimensional interval without any
  information about the character of the end points (used in definite
  integration). The arguments are the start and the end points of the interval
  in that order.
  </Description>
  <Example>
  The interval 1.0, ..., 10.0.
  <OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0" cdbase="http://www.openmath.org/cd">
    <OMA>
      <OMS cd="interval1" name="interval"/>
      <OMF dec="1.0"/>
      <OMF dec="10.0"/>
    </OMA>
  </OMOBJ>
  </Example>
  <FMP>
  <OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0" cdbase="http://www.openmath.org/cd">
    <OMA>
      <OMS cd="set1" name="subset"/>
      <OMA>
        <OMS name="interval" cd="interval1"/>
        <OMV name="a"/>
        <OMV name="b"/>
      </OMA>
    </OMA>
    <OMA>
      <OMS name="interval_cc" cd="interval1"/>
      <OMV name="a"/>
      <OMV name="b"/>
    </OMA>
  </OMOBJ>
  </FMP>
</CDDefinition>
• OMOBJECT as outer wrapper around every OpenMath Application of one OM Expression (to others)

• Binding of variables to symbols

• Attributions of any object with key-value pairs

• Errors

Compound objects construct bigger trees
Examples

4 mod 7

<OMA>
  <OMS cd="integer2" name="class" />
  <OMI>4</OMI>
  <OMI>7</OMI>
</OMA>
Examples

$4 \cdot 2 < x$

\[
\begin{align*}
\text{<OMA>}
\text{<OMS cd="relation1" name="lt" />}
\text{<OMA>}
\text{<OMS cd="arith1" name="times" />}
\text{<OMI>4</OMI>}
\text{<OMI>2</OMI>}
\text{</OMA>}
\text{</OMA>}
\text{<OMV name="x" />}
\end{align*}
\]
Examples

\{1, \pi, \varphi(9)\}

\begin{verbatim}
<OMA>
  <OMS cd="set1" name="set" />
  <OMI>1</OMI>
  <OMS cd="nums1" name="pi" />
  <OMA>
    <OMS cd="integer2" name="euler" />
    <OMI>9</OMI>
  </OMA>
</OMA>
\end{verbatim}
Representations

- OpenMath allows for different representations:
  - XML
  - Binary

You don't want to type that!

but

there is hope!
Possibly Only Practical Convenient OpenMath Replacement Notation
Integers, Floats and Strings as you expect:
18, 0.6, 2.09e3, “StAndrews w/o golfing”

Symbols: cdname.symbolname

Variables: $name, References: #name

Application: arith1.plus(1,2,3)

Binding: fns1.lambda[$x -> $x + 1]

Attribution: some.thing{aa.bb -> 1}

Some abbreviations and infix operators
### Some Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Simplified</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 + 2$</td>
<td>1+2</td>
</tr>
<tr>
<td>$x \rightarrow x \cdot \pi$</td>
<td>$\text{lambda}[$x $\rightarrow$ $x \cdot \pi]$</td>
</tr>
<tr>
<td>$17.6 \cdot e$</td>
<td>17.6*e</td>
</tr>
<tr>
<td>$\frac{22}{7}$</td>
<td>$\frac{22}{7}{	ext{somecd.approx}$ $\rightarrow$ 3.14$}$</td>
</tr>
<tr>
<td>$\int_{0}^{1} \frac{1}{x^{3} + \sin x} dx$</td>
<td>$\text{calculus1.defint}(0 .. 1, \text{lambda}[$x$ \rightarrow 1/(x^{3} + \sin(x))])$</td>
</tr>
</tbody>
</table>

Nifty, eh?
OpenMath is an open language for representing mathematical semantics.

The semantics is organized through the Symbols that are defined in CDs.

OpenMath can easily be extended.

OpenMath allows different representations for different use cases.

But how do you **use** OpenMath?
Support OpenMath in some way

What about YOUR Application?

- GAP
- KANT
- Maple
- MuPAD
- Trip
- Magma
- Macaulay2
- Mathematica
- YACAS
- ...

Support in Software
Pure Java Library to handle OpenMath objects
import and export to different formats
easy to add new importers and exporters
packed with convenience-methods for assembling and analyzing object-trees
support for Groovy short-notation
support for Scala matchers etc.

org.symcomp.openmath
This encapsulates all common functionality.
Import and Export Formats

- OpenMath XML
- POPCORN
- LaTeX (export only, import under consideration)
- Strict Content MathML (under development)
- Custom Renderers (e.g., Magma)
Parsing and generation is wired into `org.symcomp.openmath`

- `OpenMathBase.parsePopcorn("1+\$a")`\(\rightarrow\) `<OMA><OMS cd="arith1" name="plus"/>
  <OMI>1</OMI><OMV name="a"/></OMA>`

- `(new OMSymbol("cdn", "moo")).apply((new OMNumber(42)).toPopcorn())`\(\rightarrow\) `cdn.moo(42)`
• Creation Examples:
  
  - `str = new OMString("asa")`
  - `symb = new OMSymbol("cdname", "symbname")`
  - `v = symb.apply(str, new OMInteger(9))`

• Analysis Examples:
  
  - `str.isString() → true`
  - `v.isApplication() → true`
  - `v.isApplication("cdname", "symbname") → true`
Creating custom renderers extremely simple

derive \texttt{org.symcomp.openmath.AbstractRenderer}

override some simple \texttt{render} functions if necessary

Create classes \texttt{CdName} for every CD you want to support

Create methods in \texttt{CdName}:

\texttt{void symbol(out)} to render the symbol alone

\texttt{void symbol(out, OpenMathBase[] params)} to render the application of the symbol
Handling OpenMath in Java/Scala is now simple, fast, convenient, and flexible.

The library is available under an Apache 2 license.

What the heck is Scala???
Probably the most exciting new programming language

- Runs on the JVM (also .Net)
- Fully interoperable with Java
- provides an Interpreter, too
Concise, Clear, Clever

```scala
def showMatcher(om: OpenMathBase): String = {
  om match {
    case OMInteger(x) => "Integer: "+x
    case OMSymbol(cd, name) => "Symbol with cd="+cd+" name="+name
    case OMApply(h, p) => "Application"
  }
}
```

- **Match on Object structure**
- **implicitly statically typed!**
- **implicit new**
Roundup

- A very modern high level language on the JVM
- extremely concise
- can be arbitrarily mixed with Java
- scientific foundation (developed at Ecole Polytechnique Fédérale de Lausanne)

www.scala-lang.org
OpenMath can be used to...

- exchange mathematical objects between computational systems
- store mathematical objects in databases
- process and manipulate mathematical objects
- Annotate mathematical presentation with semantics

But if you want to manipulate mathematical objects, you also need a protocol...
Symbolic Computation
Software Composability Protocol

Direct linking CAS to CAS

Linking CAS to other systems

Middleware

Grid-Clients
SymGrid

SOAP-Clients
Web-Apps, Java, C#, other CAS, ...

GET-Clients
Scripts, other simple apps, ...

Humans
Administration

GAP
GAP

KANT
MuPAD
Maple

GAP

KANT

MuPAD

Maple

CAS1

CAS2

CAS3

http GET

http html

http

http html
Protocol for communication between CASes
- OpenMath based
- Lightweight, simple sockets
- Basis for symbolic computation on Clusters and Grids
- Described in the “SCSCP standard (version 1.3)”, and scscp1 and scscp2 Content Dictionaries
Possible applications

- Cross-program: Software A can do things B can’t,
- Cross-program: A can do things much faster than B,
- Cross-platform: A is only available on *nix,
- A is a pain to compile and install,
- etc…

http://www.symbolic-computation.org/scscp
Symbolic Computation Software Composability Protocol

Synchroneous communication on sockets
Version negotiation after connect
Mathematics and protocol messages are OpenMath
Request is wrapped in a `scscp1.procedure_call` application
Response is wrapped in a `scscp1.procedure_completed` application

metadata is xml PIs:
`<?scscp start ?>`

<?xml version="1.0"?>

<OMOBJ>
  <OMATTR>
    <!-- Attribution pairs (only call_id is mandatory) -->
    <OMATP>
      <OMS cd="scscp1" name="call_id" />
      <OMSTR>call_identifier</OMSTR>
      <OMS cd="scscp1" name="info_runtime" />
      <OMI>runtime_in_milliseconds</OMI>
      <OMS cd="scscp1" name="info_memory" />
      <OMI>used_memory_in_bytes</OMI>
      <OMS cd="scscp1" name="info_message" />
      <OMSTR>some extra information</OMSTR>
    </OMATP>
    <!-- Attribution pairs finished, now the result -->
    <OMA>
      <OMS cd="scscp1" name="procedure_completed" />
      <!-- The result itself, may be OM symbol for cookie -->
      <!-- OM_object_corresponding_to_the_result -->
    </OMA>
  </OMATTR>
</OMOBJ>
Server Querying

- Server is supposed to be able to describe itself: `scscp2.get_allowed_heads`
- Result may be stored and referenced later
- Some session management
- To allow special functions, transient CDs may be used
- CD `scscp2` contains these functions.
### SCSCP Messages

- **Procedure Call:** `procedure_call`
- **Procedure Completed:** `procedure_completed`
- **Procedure Terminated:** `procedure_terminated`

### Options

- **Option Runtime:** `option_runtime`
- **Option Debug Level:** `option_debuglevel`
- **Option Min Memory:** `option_min_memory`
- **Option Max Memory:** `option_max_memory`
- **Option Return Object:** `option_return_object`
- **Option Return Cookie:** `option_return_cookie`
- **Option Return Nothing:** `option_return_nothing`

### Standard Errors

- **Error Runtime:** `error_runtime`
- **Error Memory:** `error_memory`
- **Error System Specific:** `error_system_specific`

### Info

- **Info Runtime:** `info_runtime`
- **Info Memory:** `info_memory`
- **Info Message:** `info_message`

### Remote Objects

- **Store Session:** `store_session`
- **Store Persistent:** `store_persistent`
- **Retrieve:** `retrieve`
- **Unbind:** `unbind`

### Special Procedures

- **Get Allowed Heads:** `get_allowed_heads`
- **Is Allowed Head:** `is_allowed_head`
- **Get Transient CD:** `get_transient_cd`
- **Get Signature:** `get_signature`
- **Get Service Description:** `get_service_description`

### Special Symbols

- **Signature:** `signature`
- **Service Description:** `service_description`
- **Symbol Set:** `symbol_set`
- **Symbol Set All:** `symbol_set_all`
- **No Such Transient CD:** `no_such_transient_cd`
Implementations as of today

- GAP, KANT, MuPAD, Maple
- Even more: Macaulay2 (out of box), TRIP (out of box), Coq (prototype), Magma (wrapper), Mathematica (coming soon), ...
- Java OpenMath and SCSCP API: java.symcomp.org
- A collection of tools and prototypes that were built around this API (WUPSI, ISS, LattViz, SkySym, ...)

Again: What about YOUR Application?
org.symcomp.scscp

- Pure Java Library implementing SCSCProtocol,
- Both client and server side,
- Uses org.symcomp.openmath for OpenMath objects,
- Includes examples for both client and server apps.

aim: Implement your own client or server by doing nothing more than strictly necessary.
Creating an SCSCP Client
(44 lines)

Java lyrics
Init: host, port
Construct OpenMath question & call
Send request, and wait....
Done, we are!

```java
package org.syncomp.frameworkDemo.client;
import org.syncomp.openmath.*;
import org.syncomp.scscp.*;

public class Addition extends SCSCPClient {

    public Addition(String host, int port) {
        super(host, port);
        loglevel = 10;
    }

    public void demo() {
        // Construct the question
        OMSymbol nd = new OMSymbol("scscp.transient.1", "addition");
        OMInteger[] ints = { new OMInteger(2), new OMInteger(3) };
        OMApply op = new OMApply(new OMSymbol("list1", "list"), ints);
        OMApply[] args = { op };

        // Wrap the question into an SCSCP-request
        ProcedureCall req = new ProcedureCall("42", nd, args);

        // Send the request to the server, and wait for the result
        System.out.println("Sending request...");
        System.out.println(req.getOMObject().toXml());
        String token = compute(req);
        System.out.println("Waiting for result....");
        while (!resultAvailable(token)) {
            try { Thread.sleep(80); } catch (Exception e) {}
        }

        // Print the state
        Integer st = getComputationState(token);
        if (st == ComputationState.WAITING) System.out.println("ComputationState: WAITING");
        if (st == ComputationState.COMPUTING) System.out.println("ComputationState: COMPUTING");
        if (st == ComputationState.READY) System.out.println("ComputationState: READY");
        if (st == ComputationState.ERRORIOUS) System.out.println("ComputationState: ERRORIOUS");

        // Print the result.
        OMAppliClose res = getResult(token);
        if (res == null) System.out.println("null returned");
        else System.out.println(res.toXml());
    }
}
```
Creating an SCSCP Server (1)
ProcedureCallHandler (36 lines)

```java
package org.symcomp.frameworkDemo.server;

import org.symcomp.scscp.*;
import org.symcomp.openmath.*;

public class AdditionHandler extends ProcedureCallHandler {
    public String getServiceNameStr() { return "addition"; }
    public String getDescription(OMSymbol om) { return "addition(1st): Add numbers"; }

    public OpenMathBase handlePayload(OpenMathBase om) throws OpenMathException {
        //OMB contains the actual call to addition as 1st argument, so we bypass that.
        OpenMathBase p = ((OMApply) om).getParams()[0];
        OpenMathBase[] lst;

        //If the first argument is an integer, we treat the arguments as the arguments
        if (p.isInteger()) {
            lst = ((OMApply) om).getParams();
        } else if (p.isApplication("list1", "list") || p.isApplication("linear2", "vector")) {
            lst = ((OMApply) p).getParams();
        } else {
            throw new OpenMathException("Invalid arguments to Addition");
        }

        //Construct the answer
        StringBuffer q = new StringBuffer();
        for (int i = 0; i < lst.length; ++i) {
            q.append(" (OMInteger) lst[i].getValue().toString() ");
        }
        OMInteger res = new OMInteger(q.toString());

        //Return the result
        return res;
    }
}
```
Creating an SCSCP Server (2)

**SCSCPServer** (25 lines)

Java lyrics

Server name, version & description

Constructor: init & add handlers

Usage: `spawn` or `breed`
Creating an SCSCP Server or Client in Java is now simple, fast, convenient, and easy. And we use it, too! For examples:
- For the Webproxy
- For MuPAD
- For Magma
- For Mathematica (JLink)
What else?

- Grid and Cluster Infrastructure
- Webproxy, a Web based Administration and orchestration tool offering SOAP access
- WUPSI, the Wonderful Universal Popcorn SCSCP Interface: a great testing and demo tool
Thank you...

...for your attention.

The libraries are available at
http://java.symcomp.org