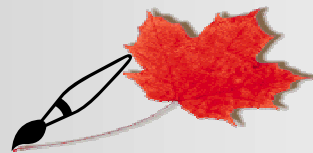




The University of Western Ontario

**Sandy Hurter, Elena Smirnova,
Clare M. So, Zheng Wang, Stephen M. Watt**

Providing Mathematical Web Services Using Maple in the MONET Architecture





WP4:

Demonstrator Services and Applications

Task 4.2:

Symbolic Solver Service

Task 4.3

Mathematical Problem Solving Environment

Overall approach

User (service author)'s point of view:

One XML-based configuration file for each service

3 parts to file:

- service MSDL skeleton,
- service interface,
- service implementation (Maple code in our case)

```
<mathServer>
  <msdl>
    <service name="service_A">
      {MSDL for a service A}
    </service>
    {MSDL for other services}
  </msdl>

  <services>
    <service name="service_A" call="function_call_for_service_A"/>
    {interface for other services}
  </services>

  <implementation language = "maple">
    {Maple implementation for each service}
  </implementation>
</mathServer>
```



Advantages of this approach

- Ease of use:
the author of service has only to provide Maple code, but does not need to know about Java, web-services, XML tools, etc.
- Flexibility:
service property (interface and maple code) could be easily changed by updating the configuration file (service re-installation is not required)
- Portability:
service described in the configuration file can be deployed elsewhere without performing any changes



Developer's point of view:

Matreshkas:



- Java baby-sits Maple and reports to Axis wrapping servlet.
- All the maple code and interface between service call and Maple is loaded from the XML configuration file.
- Specially designed shell scripts drive process of service installation on a web server

Developed services

- Indefinite integration service
- Definite symbolic integration
- Definite numeric Integration
- Ordinary differentiation
- Fractional differentiation
- Symbolic-order differentiation of rational functions
- Limit calculation
- Series expansion
- Approximate GCD
- Root-finding service (including parametric solutions)
- Solving systems of polynomials
- Math format conversion

▪ Examples of Monet Service for Differentiation

```
<mathServer>
  <msdl>
    <service name="NthDiffService">
      <classification>
        <taxonomy taxonomy="http://gams.nist.gov" code="Gams0"/>
        <problem href="http://monet.nag.co.uk/problems/differentiation"/>
        <format>http://www.openmath.org</format>
        <directive-type href="http://monet.nag.co.uk/owl#evaluate"/>
      </classification>

      <implementation>
        <software href="http://monet.nag.co.uk/owl#Maple9"/>
        <hardware href="http://monet.nag.co.uk/owl#PentiumSystem"/>
      </implementation>

      <service-interface-description href = ""/>

      <service-binding>
        <map operation = 'runService' action="invokeService"
              problem-reference="http://monet.nag.co.uk/problems/differentiation"/>
        <message-construction io-ref = 'http://monet.nag.co.uk/problems/differentiation#f'
                              message-name = 'runServiceRequest' message-part='in0/*[1]'/>
        <message-construction io-ref = 'http://monet.nag.co.uk/problems/differentiation#derivative'
                              message-name = 'runServiceResponse' message-part='runServiceReturn'/>
      </service-binding>

      <service-metadata/>

      <broker-interface>
        <service-URI/>
        <broker-interface-description/>
      </broker-interface>

    </service>
  </msdl>
```



```

<services>
  <service name="DiffService" call="My_diff_module:-my_diff"/>
</services>

```

```

<implementation language = "maple">
  My_diff_module := module()
    export my_diff;
    my_diff:=proc(f,x); diff(f,x); end:
  end:
</implementation>
</mathServer>

```

===== OR =====

```

<services>
  <service name="DiffService" call="SymbolicNthDerivative:-nthd_ratpolyc"/>
</services>

```

```

<implementation language = "maple">
#-----#
# package SymbolicNthDerivative
# author Dr.Mhenni BenGhorbal,
# Ontario Centre for Computer Algebra,
# University of Western Ontario
# puprose calculates the symbolic nth derivative
# of rational univariate polynomials in
# both explicit and compact forms
# version 1.0
# date Winter 2004
#-----#
SymbolicNthDerivative := module() option package;
export nthd_ratpolyc,
      nthd_ratpolyc;
#-----
# explicit form
#-----
nthd_ratpolyc := proc( f )
  local A, w, c, c1, c2, c3, c4, c5, c6, i,

```



```

s, t1, t2, t3, v, parfrac1, parfrac2, parfrac3, nthderiv,
terms, List, Power, subterms, Select;
parfrac1 := convert( f, fullparfrac, x, 'normal' );
if type( parfrac1, function ) then
  A := [ allvalues( parfrac1 ) ];
  t1:= 0;
  for w from 1 to nops(A) do
    c := ( op( 1, A[w] ) );
    c1 := [ op( c ) ];
    c2 := nops( c1 );
    c3 := op( c2 );
    parfrac2 := Product( -i, i = 1..n )*subs(op(1,op(c3,c))=op(1,op(c3,c))^(n+1),A[w]);
    nthderiv := normal(value(parfrac2));
    t1 := t1 + nthderiv;
  od;

elif ( type( parfrac1, `*` ) and has( [parfrac1], x )) then
  List := [ op( parfrac1 ) ];
  Select := select ( has, List, x );
  if type( op( Select ), symbol ) then
    t1 := normal(Product( 1 - k, k = 0..n-1 )*combine( parfrac1 * x^( -n ), power));
  else
    Power := op( 2, op( Select ) );
    t1 := normal( combine( parfrac1 *
      op( 1, op( Select ))^(-n), power) *
      Product( Power - k, k = 0..n-1 ) );
  fi;
elif ( type( parfrac1, `^` ) and has( [ parfrac1 ], x )) then
  subterms := [ op( parfrac1 )];
  if type ( subterms[1], `symbol` ) then
    Power := subterms[2];
    t1 := normal(combine(subterms[1]^(Power-n),power)*Product(Power-k,k= 0..n-1));
  else
    Power := subterms[2];
    t1 := normal( combine( subterms[1]^(Power-n)*Product(Power-k,k=0..n-1),power ) );
  fi;

elif (type(parfrac1,`symbol`)and has([parfrac1],x))then

```



```

t1 := GAMMA(2)/GAMMA(2-n)*x^(1-n);

elif not has( [(parfrac1)],x) then
t1 := parfrac1 * Product(1-k,k=0..n);
else
terms := [ op( parfrac1 ) ];
t1 := 0;
for i from 1 to nops( terms ) do
if has( terms[i], x ) then
if type( terms[i], `` ) then
subterms := [ op( terms[i] ) ];
Power := subterms[2];
nthderv := normal(combine(subterms[1]^(Power-n)*Product(Power-k,k=0..n-1),power));
elif
type( terms[i], `symbol` ) then
nthderv := GAMMA(2)/GAMMA(2-n) * x^( 1 - n );
elif
type( terms[i], function ) then
A := [ allvalues( terms[i] ) ];
t2 := 0;
for w from 1 to nops(A) do
c := A[w];
c1 := value( c );
if type( c1, `+` ) then
t3 := 0 ;
for v from 1 to nops(c1) do
c2 := [ op(c1) ];
c3 := c2[v];
c4 := [ op(c3) ];
c5 := c4[ nops(c4) ];
c6 := [op( c5 )];
Power := c6[2];
nthderv:=normal(Product(Power-k,k=0..n-1)*combine(c6[1]^(-n)*c3,power));
t3 := t3 + nthderv;
nthderv := t3;
od;
else
c2 := [ op(c1) ];

```



```

        c3 := c2[nops(c2)];
        c4 := [ op(c3) ] ;
        Power := c4[ nops(c4) ];
        nthderv:=normal(Product(Power-k,k=0..n-1)*combine(c4[1]^(-n)*c1,power));
    fi;
    t2 := t2 + nthderv;
    nthderv:=t2 ;
od;
else
    List := [op(terms[i])];
    Select := select (has,List,x);
    if type(op(Select), symbol) then
        nthderv:= normal(Product(1-k, k=0..n-1)*combine(terms[i]*x^(-n),power));
    else
        Power := op(2,op(Select));
        nthderv:= normal(combine(terms[i]*op(1,op(Select))^(-n),power)*
            Product(Power-k, k=0..n-1));
    fi;
fi;
else
    nthderv := normal( terms[i] * Product( k, k =(1-n)..1) );
fi;
t1 := t1 + nthderv;
od;
fi;
return t1;
end;#nthd_ratpolyc

```

```

#-----
# compact form
#-----
nthd_ratpolyc := proc( f )

```

```

    local A, B, w, c, c1, c2, c3, c4, c5, c6, i, t1, t2, t3, s, t, v,
    parfrac1,parfrac2, parfrac3, nthderv, terms, List, Power, subterms, Select;
    parfrac1 := convert( f, fullparfrac, x, 'normal' ) ;
    if type( parfrac1, function ) then
        c := op( 1, parfrac1 );

```



```

    c1 := [ op( c ) ];
    c2 := nops( c1 );
    parfrac2 := product(-i,i=1..n)*subs(op(1,op(c2,c))= op(1,op(c2,c))^(n+1),parfrac1);
    nthderv := (parfrac2);
    t1 := nthderv;
elif (type(parfrac1,`*`) and has([parfrac1],x )) then
    List := [ op( parfrac1 ) ];
    Select := select ( has, List, x );
    if type( op( Select ), symbol ) then
        t1 := product( 1 - k, k = 0..n-1 )*combine( parfrac1 * x^( -n ), power);
    else
        Power := op( 2, op( Select ) );
        t1 := combine( parfrac1 *
            op( 1, op( Select ))^(-n), power)*product( Power - k, k = 0..n-1 );
    fi;
elif ( type( parfrac1, `^` ) and has( [ parfrac1 ], x )) then
    subterms := [ op( parfrac1 ) ];
    if type ( subterms[1], `symbol` ) then
        Power := subterms[2];
        t1 := combine( subterms[1]^(Power-n), power )*product( Power - k, k = 0..n-1 );
    else
        Power := subterms[2];
        t1 := combine( subterms[1]^(Power-n)*product( Power - k, k = 0..n-1), power );
    fi;
elif( type( parfrac1, `symbol` ) and has([parfrac1], x )) then
    t1 := GAMMA(2)/GAMMA(2-n) * x^( 1 - n );
elif
    not has( [ (parfrac1) ], x ) then
    t1 := parfrac1 * product( 1-k, k = 0..n );
else
    terms := [ op( parfrac1 ) ];
    t1 := 0;
    for i from 1 to nops( terms ) do
        if has( terms[i], x ) then
            if type( terms[i], `^` ) then
                subterms := [ op( terms[i] ) ];
                Power := subterms[2];
                nthderv := combine( subterms[1]^(Power-n)*product( Power - k, k=0..n-1), power );
            fi;
        fi;
    end do;

```



```

    elif
      type( terms[i], `symbol` ) then
        nthderv := GAMMA(2)/GAMMA(2-n) * x^( 1 - n );
    elif
      type( terms[i], function ) then
        c := op( 1, terms[i] );
        c1 := [ op( c ) ];
        c2 := nops( c1 );
        c3 := op( c2, c1 );
        c4 := [ op( c3 ) ];
        Power := c4[ nops( c4 ) ];
        parfrac2 := product(Power-k,k=0..n-1)*subs(op(1,op(c2,c))=op(1,op(c2,c)))^
          ((-Power + n)/(-Power)),terms[i]);
        nthderv := (parfrac2);
    else
      List := [ op( terms[i] ) ];
      Select := select ( has, List, x );
      if type( op( Select ), symbol ) then
        nthderv := product( 1 - k, k = 0..n-1 ) * combine( terms[i]*x^(-n), power );
      else
        Power := op( 2, op( Select ) );
        nthderv:=combine(terms[i]*op(1,op(Select))^(-n),power)*
          product(Power-k,k=0..n-1);
      fi;
    fi;
  else
    nthderv := terms[i] * Product( k, k =(1-n)..1);
  fi;
  t1 := t1 + nthderv;
od;
fi;
return t1;
end; #nthd_ratpolyc
end; #module
</implementation>
</mathServer>

```



Implementation details

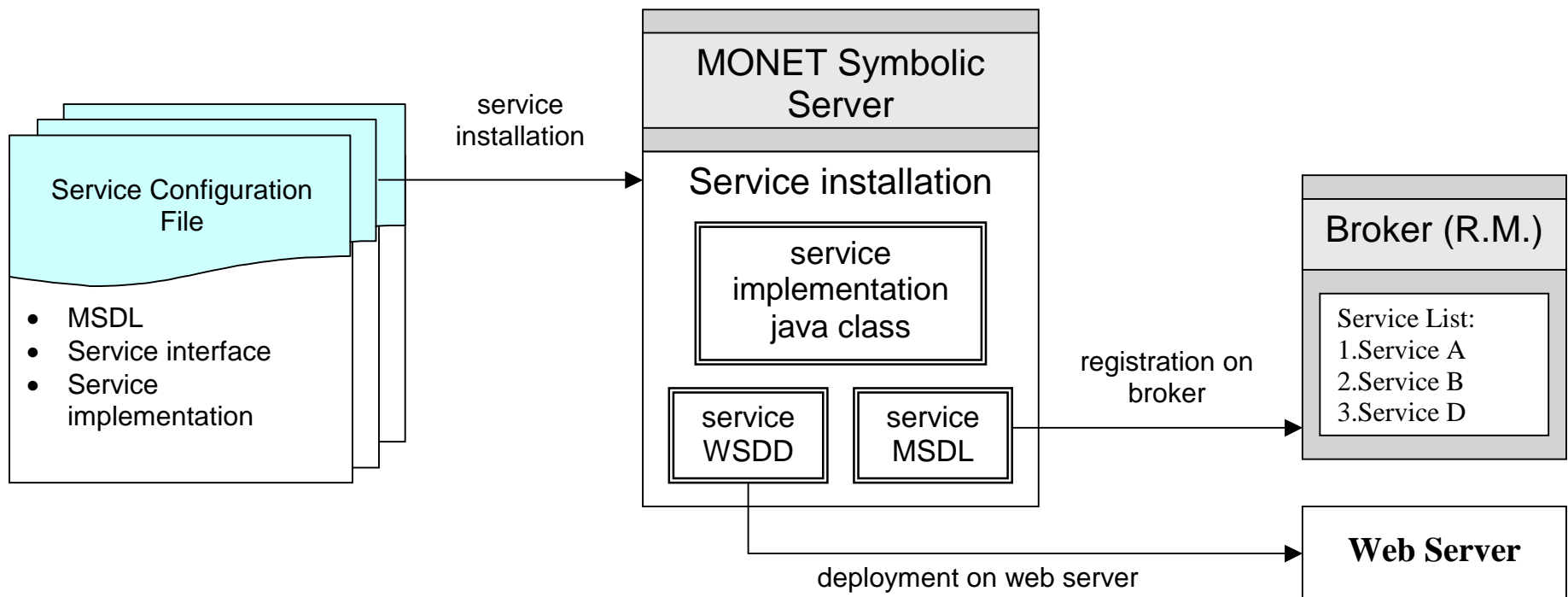
3 parts of software for each service:

- service-developing tools (common for all services)
 java + shell scripts
- generated core java classes for service implementation
 use service configuration file and service-developing tools
- interface to broker and client
 java + jsp

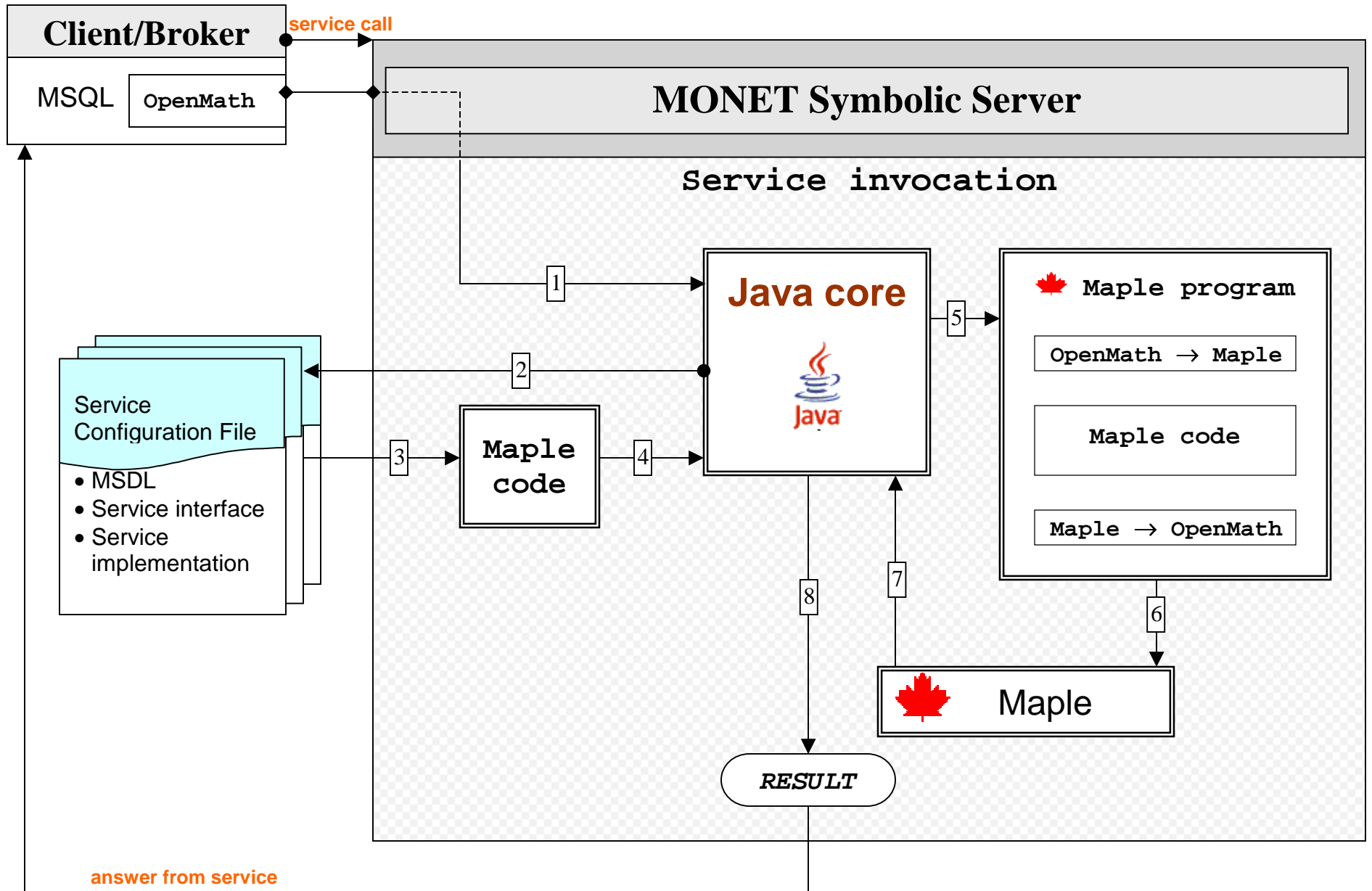
Service installation

- Installation of a service requires a service configuration file and includes the following steps:
 - generating java code, implementing the service itself
 - creating service descriptor file, used by Apache Axis for deploying service on a web server
 - updating MSDL file from configuration file and service registration with the MONET Broker
- A shell script, provided with software tools kit, drives the process of service installation:

```
mathservice_install.sh <service config.file> <broker URL> [path to Maple] [port of service deployment]
```



Service invocation



■ Other service implementation features


- Service logging (available for explanation and debugging)
- SOAP monitoring
- Documentation
 - plain text
 - hypertext format (web pages)
- Error catching
 - in Maple use “try – catch – finally” statement
 - By default caught run-time errors will be returned as <OME>
- Service is able to return result in format, different than OpenMath
 - Content MathML
 - Presentation MathML,
 - LaTeX
- Conversion between Maple and OpenMath is an important part of service implementation



Conversion between OpenMath and Maple

▪ OpenMath to Maple

Relatively easy

- First attempt: Bath phrasebook written in Java
- Problems:
 - Uncaught exceptions kill Tomcat 
 - Fixed set of CDs
 - For updating Java code users need to know the implementation.

OpenMath To Maple Conversion - Our Approach

- Program written in Maple
- Driven by *mapping definition files*
 - User needs to provide mapping files
 - Extensible!
- Why this approach?
 - OpenMath is extensible
 - Similar conversion patterns
- Similarities Between OpenMath and Maple

OpenMath	Maple
<pre><OMOBJ> <OMA> <OMS cd="transc1" name="sin"/> <OMV name="x"/> </OMA> </OMOBJ></pre>	$\sin(x)$



About Mapping Files

- Describe the uses of OMSs and their Maple equivalent
- One mapping file per CD
- No conversion for a particular OMS is hard-coded in the Maple program
- Currently, some experimental CDs and almost all CDs in the MathML group are implemented.
- Sample Mapping File Entry

```
<map>
  <OpenMath>
    <OMA>
      <OMS cd="transc1" name="sin"/>
      <arg pos="1"/>
    </OMA>
  </OpenMath>
  <Maple>
    <head>sin</head>
    <arg pos="1"/>
  </Maple>
</map>
```



OpenMath To Maple Conversion – Challenges

- Not all conversions follow the same pattern
 - Allow Maple code to be inserted
 - No Maple equivalent for OMS



Maple To OpenMath Conversion

- More difficult
- First attempt: Maple → Content MathML (Maple export) → OpenMath (XSLT stylesheet)
- Problems:
 - Lose semantics
 - Does not know about various CDs

Maple To OpenMath Conversion – Our Approach

- Program written in Maple
- Driven by the **SAME** mapping files for the OpenMath → Maple conversion

Maple To OpenMath Conversion – Challenges

- Overloaded operators in Maple:
int(x,x) and int(x,x=1..10)
- It is on-going work to develop “*disambiguation hooks*”



Exposing service interface to the outside world

- **BROKER**

Registration with broker (broker prototype OR new fully-functional Monet broker)

- automatically performed at the moment of service installation
- can be done at any time by using software tool supporting Service to Broker interface

- **CLIENT**

Client-side software tool-kit provides 3 client interfaces for each service:

- universal command line and graphical interface for any MONET Maple-UWO service
- client direct web interface, automatically generated for deployed services



Example of service invocation log:

2004-03-14 02:52:45.419

Configuration for mathservice DefIntService has been loaded from /scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/samples/configfiles/intServiceConfig.xml

2004-03-14 02:52:45.419

Service agrument(s):

0:

<OMA>

<OMS cd ='arith1' name ='power'/>

<OMV name ='x'/>

<OMI>-3 </OMI>

</OMA>

1:

<OMV name ='x'/>

2:

<OMV name ='a'/>

2004-03-14 02:52:45.449

Invoke the service

Math Formats:

Service input : OpenMath

Maple input : OpenMath

Maple output : OpenMath

Service output: OpenMath

Available tools for math format conversion :

OpenMath to Maple package : /scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/maple/omo_to_mob.mpl

Maple to OpenMath package : /scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/maple/mob_to_omo.mpl

Content MathML to OpenMath XSLT : file:///scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/xslt/cmmltoom.xsl

2004-03-14 02:52:45.46



Maple operation to evaluate:
My_int_module:-my_def_int

2004-03-14 02:52:45.46

List of OpenMath args, submitted to Maple:

```
["<OMA><OMS cd='arith1' name='power'/><OMV name='x'/><OMI>-3</OMI></OMA>","<OMV name='x'/>","<OMV name='a'/>"]
```

2004-03-14 02:52:45.461

Maple input file content:

```
=====
restart;
# generated temporary filename for Maple program output
outputFileName:="/scl/packages/jakarta-tomcat-5.0.16/temp/MathServiceOutputs/mapleOut12666.tmp";
# file containing package for OpenMath to Maple conversion
OM2MaplepackageFile:="/scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/maple/omo_to_mob.mpl";
# file containing package for Maple to OpenMath conversion
maple2OMpackageFile:="/scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/maple/mob_to_omo.mpl";
# path to OM<->Maple converter directory
MobOMpackageDir:="/scl/people/elena/MONET/SymbolicMathServer-UWO/SymbolicMathServer/maple";

My_int_module := module()

  export
    my_indef_int,
    my_def_int,
    my_num_int;

  my_indef_int:=proc(f,x);
    int(f,x);
  end;

  my_def_int:=proc(f,x,lowlimit,upperlimit)
    int(f,x=lowlimit..upperlimit,AllSolutions);
  end;
```



```

my_num_int:=proc(f,x,range,accuracy)
  evalf(Int(f,x=range,digits=-ilog10(accuracy)));
end;

end:

try
  olddir := currentdir();
  currentdir(MobOMpackageDir);
  # read package for OpenMath to Maple conversion
  read OM2MaplepackageFile;
  # read package with Maple to OpenMath conversion
  read maple2OMpackageFile;
  try
    omArgs:=["<OMA><OMS cd='arith1' name='power'/><OMV name='x'/><OMI>-3</OMI></OMA>","<OMV name='x'/>","<OMV name='a'/>"];
    # convert list of OpenMath expressions to sequence of Maple expression
    mapleArgs:=seq(OpenMathToMobConversion:-omtomaple(omArgs[i]), i=1..nops(omArgs));
    output :=My_int_module:-my_def_int(mapleArgs);
  catch :
    errorFlag:=1;
    err:= lastexception;
    errmsg:= StringTools:-FormatMessage( err[2..-1] );
    #ans:= cat("ERROR: ",errmsg);
    ans:= XMLTools:-PrintToString(MobToOpenMathConversion:-mapletoom_error(errmsg));
  finally:
    if (errorFlag<>1) then
      if type(output,list) then
        ans:= MobToOpenMathConversion:-mapletoom(output);
      else
        ans:= MobToOpenMathConversion:-mapletoom([output]);
      end if;
      ans:= XMLTools:-PrintToString(ans);
    end if;

    fd := fopen(outputFileName,WRITE):
    fprintf(fd, "%s",ans):
    fclose(fd):
  end try:
  currentdir(olddir);
catch :
  err:= lastexception;

```



```
errmsg:= StringTools:-FormatMessage( err[2..-1] );
# ans:= cat("ERROR: ",errmsg);
ans:= XMLTools:-PrintToString(MobToOpenMathConversion:-mapletoom_error(errmsg));
fd := fopen(outputFileName,WRITE);
fprintf(fd, "%s",ans);
fclose(fd);
end try;
quit;
```

2004-03-14 02:52:45.473

Maple input file stored as /scl/packages/jakarta-tomcat-5.0.16/temp/MathServiceInputs/mapleIn12665.tmp

2004-03-14 02:52:45.474

Path to Maple is "maple"

2004-03-14 02:52:45.474

Executing command maple /scl/packages/jakarta-tomcat-5.0.16/temp/MathServiceInputs/mapleIn12665.tmp

2004-03-14 02:52:54.212

Execution done. Exit value = 0

2004-03-14 02:52:54.214

The answer from Maple:

```
<OMOBJ xmlns = 'http://www.openmath.org/OpenMath'>
  <OME>
    <OMS cd = 'moreerrors' name = 'algorithm'>
      <OMSTR>my_def_int uses a 4th argument, upperlimit, which is missing</OMSTR>
    </OME>
  </OMOBJ>
```

