

Version: 1.0
Date: March 2003

MONET – Mathematical Problem Ontology

Olga Caprotti¹, David Carlisle², Arjeh M. Cohen³ and
Mike Dewar²

¹RISC-Linz ²NAG ³TUE

Deliverable D11 (Public)

Contents

1	Introduction	2
2	Mathematical Problem Ontology	2
2.1	Problem Context	3
2.2	Evaluate	4
2.3	Prove	4
2.4	Decide	5
2.5	Lookup	6
2.6	Common Metadata	8
3	XML Representation	8
4	More Examples	9
5	Relation to Other MONET Ontologies	10
6	Conclusion	10

1 Introduction

This document describes one part of the MONET architecture [4] for mathematical web services. It is concerned with how a client describes mathematical problems to be submitted to a MONET broker or planner. The objects of the mathematical problem ontology represent concrete questions such as

- *find the real roots of the polynomial $x^2 - 2$*
- *prove Robbins problem, namely that the Huntington equation follows from associativity, commutativity, and from Robbins axiom*
- *decide whether 1234567891 is prime*
- *what does the notion “regular” stand for when dealing with matrices?*

Because these questions are considered in the context of web services, they will be expressed in a suitable XML language. Ideally problems from clients are instances of problem descriptions in the mathematical problem library [5]. The exact syntax by which a client expresses in XML the parts that make up a problem instance is strictly related with the syntax used for the mathematical problem description library. Moreover, the language used by the client also borrows elements from the mathematical service description language to state directives and to classify the problem. For this document, we focus on using OpenMath as the mathematical markup language used to represent content, however the framework is designed to allow a choice of languages, for example one may use Content MathML instead of OpenMath.

This deliverable is structured as follows. We first describe the ontology of mathematical problems in Section 2. Next we give the XML grammar for expressing mathematical problems in MONET queries. Finally we look at some examples and list the relationships between this and other MONET Ontologies.

2 Mathematical Problem Ontology

The mathematical problems addressed in this deliverable may be constructed by subscribers to and clients of MONET services, for example they might be generated on-the-fly by software that needs specialist computations, or be found in interactive online documents [2]. The intended goal of stating a problem is finding a solution for it: in the MONET architecture this is achieved by sending the problem statement wrapped in a query to a MONET broker, namely a program that is aware of mathematical services and may return a URL where the client can find the answer [4, 6]. In essence, a mathematical problem consists of a *directive* to be performed on one or more OpenMath objects.

Recall that an OpenMath object is a mathematical object that might correspond to any of the following notions: a functional expression, a logical statement, a class of data types, or an object of a certain type. The richness of OpenMath can be exploited for describing mathematical problems with varying degrees of formalism. Clearly, a problem may be described to full extent, by providing all information that is mathematically relevant for solving the problem (its context), but it may also be under-specified.

A compact straightforward way in which to state a problem is to use the OpenMath object representing the required notion in combination with a directive such as `evaluate`. For instance, to find the squarefree factorization of the polynomial expression $x^4 - 4x^2$

```

<mql:mqol xmlns:mql="http://monet.nag.co.uk/monet/mql"
          xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
          xmlns="http://www.openmath.org/OpenMath">

<mql:query>
  <mql:mproblem>

    <msdl:classifications>
      <msdl:directive-type>
        http://monet.nag.co.uk/monet/directive#evaluate
      </msdl:directive-type>
    </msdl:classifications>
    <msdl:problem name='sqrfree_factors'>
    <msdl:header/>
    <msdl:body>
      <msdl:output name='f'>
        <OMOBJ>
          <OMA><OMS name="squarefree" cd="poly"/>
            <OMA><OMS name="plus" cd="arith1"/>
              <OMA><OMS name="power" cd="arith1"/>
                <OMV name="x"/>
                <OMI>4</OMI>
              </OMA>
            <OMA><OMS name="times" cd="arith1"/>
              <OMI>-4</OMI>
              <OMA><OMS name="power" cd="arith1"/>
                <OMV name="x"/>
                <OMI>2</OMI>
              </OMA>
            </OMA>
          </OMA>
        </OMOBJ>
      </msdl:output>
    </msdl:body>
  </msdl:problem>
</mql:mproblem>
</mql:mqol>

```

In this example, the client conveys some information in a single OpenMath object, but leaves other information (such as the field of coefficients) unspecified. In general however, the mathematical problem ontology may be used to express a problem unambiguously by introducing contextual knowledge such as a requirement on the signature of the result.

2.1 Problem Context

A mathematical problem typically lives in a certain context, e.g. a domain of computation or a specific axiomatization of a theory. For instance, the sentence $\exists x x^2 = -1$, is true over the

complex numbers but it is false over the reals. If the client specifies the problem context (e.g. $x \in \mathbb{R}$), then the broker should use the information to direct the search of a service and return the most appropriate candidate. If the broker cannot match the context, then it should report a partial match.

Specification of the context is not compulsory. When not given, the broker (or the planner) is free to return the best match according to its knowledge. In particular, an intelligent planner might be called upon in order to supply the necessary missing knowledge.

Contexts can be specified in a variety of ways. When the context is involved, it is best given by reference to the URI of a document in one of the mathematical markup languages such as OpenMath, Content-MathML, OMDoc, Coq-XML, ... In the MONET framework we will primarily use OpenMath however the ontology markup language has been designed so that it is extensible by other namespaces. These references to context can be given in `msdl:semantic-ref` or the `msdl:semantics` element within `msdl:classifications`. Context information is also conveyed by the `msdl:signature` elements, both of inputs and outputs to the problem, or by constraints imposed as `msdl:pre-conditions` or `post-conditions`.

2.2 Evaluate

A problem to evaluate involves a certain object representing a mathematical notion which has to be computed/evaluated. The input and the output could be subject to conditions or constraints. More subtle distinctions between *solving* (finding the roots of, finding an assignment) or *computing* (evaluating a closed form of an expression) or *simplifying* (finding a normal form for, or a simpler form with respect to a reduction relation) may be expressed by the choice of the proper OpenMath object representing the mathematical concept or by stating enough constraints on the result.

Examples of problems that can be given as evaluation problems are the following:

- compute the integral $\int_0^\infty e^{x^3-x} dx$
- simplify an expression
- factorise an integer or a polynomial
- solve a polynomial system of equations numerically
- find an object A such that the property P holds for A , e.g. find an integer that is a Carmichael number.

2.3 Prove

A problem using the directive `prove` involves an assertion for which the client is requesting a proof of validity, or a disproof. The validity of the assertion may not be decidable in which case the service may indicate this in its response, also in the case that the assertion is not provable, the service may or may not be able to provide a disproof or counter-example.

The result of queries of this type can be given in a number of representations in OpenMath by selecting the appropriate OpenMath CDs: as a lambda term, as a natural deduction, or even as a list of witnesses which certify the assertion as advocated in [1]. In the MONET architecture however, one may also expect to receive a proof that uses an encoding such as OMDoc. The details on how to specify such a requirement in the query are given in [6]

```

<mql:mqol xmlns:mql="http://monet.nag.co.uk/monet/mql"
          xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
          xmlns="http://www.openmath.org/OpenMath">

<mql:query>
  <mql:mproblem>

    <msdl:classifications>
      <msdl:semantic-ref>
        http://www.mathweb.org/logics/FOL
      </msdl:semantic-ref>
      <msdl:semantic-ref>
        http://www.openmath.org/cd/logic1.oed
      </msdl:semantic-ref>
      <msdl:semantic-ref>
        http://www.openmath.org/cd/logic3.oed
      </msdl:semantic-ref>
      <msdl:directive-type>
        http://monet.nag.co.uk/monet/directive#prove
      </msdl:directive-type>
    </msdl:classifications>
    <msdl:problem name='primality'>
    <msdl:header/>
    <msdl:body>
      <msdl:output name='P'>
        <OMOBJ>
          <OMA>
            <OMS cd="numbertheory1" name="isprime"/>
            <OMI>1234567891</OMI>
          </OMA>
        </OMOBJ>
      </msdl:output>
    </msdl:body>
  </msdl:problem>
</mql:mproblem>
</mql:mqol>

```

In this case the contents of the `msdl:classifications/msdl:semantic-ref` elements determine the logic and the calculus chosen for the proof, and will be used by the broker to select the appropriate service (different logical systems will use different CDs and return different proof encodings). OpenMath provides symbols for representing proofs in various logical systems, for example first-order logic or lambda calculus. As an alternative one may also indicate different encodings, such as OMDoc, to encode the resulting proof.

2.4 Decide

A problem using the `decide` directive involves an assertion which requires a boolean answer (a truth value).

A famous example of decidable theory is the theory of real closed fields for which there exist implementations of decision algorithms [3, 8]. For these situations, it is always possible to decide whether an assertion is a theorem in the theory. In the general case of a decision problem in an arbitrary theory, it might still be possible to decide the specific case.

```
<mql:mql xmlns:mql="http://monet.nag.co.uk/monet/mql"
          xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
          xmlns="http://www.openmath.org/OpenMath">

<mql:query>
  <mql:mproblem>
    <msdl:classifications>
      <msdl:directive-type>
        http://monet.nag.co.uk/monet/directive#decide
      </msdl:directive-type>
    </msdl:classifications>
    <msdl:problem name='primality'>
    <msdl:header/>
    <msdl:body>
      <msdl:output name='P'>
        <OMOBJ><OMA><OMS cd="numbertheory1" name="isprime"/>
          <OMI>1234567891</OMI>
        </OMA>
      </OMOBJ>
    </msdl:output>
    </msdl:body>
  </msdl:problem>
</mql:mproblem>
</mql:mql>
```

2.5 Lookup

A lookup problem requires explanation on a mathematical notion. The explanation might consist of a definition, bibliographical references or any other kind of material related to the concepts expressed in the problem. This problem leads to a search in the MONET knowledge base.

- Request information on a mathematical notion identified via keywords in a taxonomy, for instance given in the element `msdl:classifications/msdl:classification`. In this example the request is to lookup the notion of *quadrature* in Numeral Integration.

```
<mql:mql xmlns:mql="http://monet.nag.co.uk/monet/mql"
          xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
          xmlns="http://www.openmath.org/OpenMath">

<mql:query>
  <mql:mproblem>
    <msdl:classifications>
```

```

    <msdl:directive-type>
      http://monet.nag.co.uk/monet/directive#lookup
    </msdl:directive-type>
    <msdl:classification taxonomy="http://gams.nist.gov/" code="H2"/>
  </msdl:classifications>
</mql:mproblem>
<mql:mqol>

```

To answer a lookup directive, a MONET broker would consult the sources it has: OpenMath CDs, library of algorithms, library of problems, but also mathematical knowledge bases such as MathWeb and HELM.

- Context might also play a role: for instance a lookup of *Euler's Gamma function* instead of the *gamma constant 0.5772*.

```

<mql:mqol xmlns:mql="http://monet.nag.co.uk/monet/mql"
  xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
  xmlns="http://www.openmath.org/OpenMath"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

```

```

<mql:query>
  <mql:mproblem>
  <msdl:classifications>
    <msdl:directive-type>
      http://monet.nag.co.uk/monet/directive#lookup
    </msdl:directive-type>
    <msdl:semantic-ref>
      http://www.openmath.org/cd/hypergeo0.ocd#gamma
    </msdl:semantic-ref>
  </msdl:classifications>
  </mql:mproblem>
</mql:mqol>

```

- A further usage of a lookup directive is for returning the name of a mathematical notion described by giving an example. For instance, Is the list of integers 1,2,4,8,16,... a known series? Does it have a name?

```

<mql:mqol xmlns:mql="http://monet.nag.co.uk/monet/mql"
  xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
  xmlns="http://www.openmath.org/OpenMath">

```

```

<mql:query>
  <mql:mproblem>
  <msdl:classifications>
    <msdl:directive-type>
      http://monet.nag.co.uk/monet/directive#lookup
    </msdl:directive-type>

```

```

</msdl:classifications>
<msdl:problem name='NewYorkTimes'>
<msdl:header/>
<msdl:body>
  <msdl:output name='S'>
    <OMOBJ><OMA><OMS cd="list1" name="list"/>
      <OMI>1</OMI>
      <OMI>2</OMI>
      <OMI>4</OMI>
      <OMI>8</OMI>
      <OMI>16</OMI>
    </OMA>
  </OMOBJ>
</msdl:output>
</msdl:body>
</msdl:problem>
</mpl:mproblem>
<mql:mqol>

```

One should not confuse a `lookup` directive with a lookup strategy used for solving an `evaluate` or `decide` directive. A lookup request returns a name for a mathematical object, results known about it, and possibly services that deal with it.

2.6 Common Metadata

The metadata which is used to describe in a human-readable way the problem to be solved is common to all kinds of directive. This includes for instance an optional description of the problem in natural language or XML (e.g. `docbook`, `xhtml`) which can be given as content of a `documentation` element.

3 XML Representation

The XML representation of a problem in a query is defined by the element `mproblem` in the query namespace `http://monet.nag.co.uk/monet/mql`. This element is a container for the elements `msdl:classifications` and `msdl:problem` in the MSDL namespace. Both elements have to appear once. The `msdl:classifications` element may be empty. If no `msdl:directive-type` child exists, then the default directive is set to `evaluate`.

The `msdl:problem` element must contain an `msdl:output` child element which either has content (apart from the `msdl:signature` child) or it has a name attribute that can be used in a `msdl:post-condition`. The schema does not enforce this constraint.

The schema allows for XML from arbitrary namespaces to be included as content of the elements. For instance, the following is a problem query which uses Content-MathML:

```

<mql:mqol xmlns:mql="http://monet.nag.co.uk/monet/mql"
          xmlns:msdl="http://monet.nag.co.uk/monet/msdl"

```

```

xmlns:mml="http://www.w3.org/1998/Math/MathML">

<mql:query>
  <mql:mproblem>
    <msdl:directive-type>evaluate</msdl:directive-type>
    <msdl:problem name='mml_sin'>
      <msdl:header/>
      <msdl:body>
        <msdl:output name='d'>
          <mml:math>
            <mml:apply><mml:sin/><mml:pi/></mml:apply>
          </mml:math>
        </msdl:output>
      </msdl:body>
    </msdl:problem>
  </mql:mproblem>
</mql:query>

```

4 More Examples

A MONET problem in a query may be stated as content either of an `msdl:output` element or of a `msdl:post-condition` on the output. Here below is the primality decision problem:

```

<msdl:directive-type>
  http://monet.nag.co.uk/monet/msdl#decide
</msdl:directive-type>
<msdl:problem name='primality_decision'>
  <msdl:header/>
  <msdl:body>
    <msdl:output name='d'/>
    <msdl:post-condition>
      <om:OMOBJ>
        <om:OMA>
          <om:OMS cd="relation1" name="eq"/>
          <om:OMV name="d"/>
          <om:OMA><om:OMS cd="numbertheory1" name="isprime"/>
            <om:OMI>1234567891</om:OMI>
          </om:OMA>
        </om:OMA>
      </om:OMOBJ>
    </msdl:post-condition>
  </msdl:body>
</msdl:problem>

```

Usage of the `msdl:directive-type` is in certain cases necessary in order for the broker to disambiguate the request. In particular, the `prove` directive directive is the only difference to

the previous problem query fragment:

```

<msdl:directive-type>
  http://monet.nag.co.uk/monet/msdl#prove
</msdl:directive-type>
<msdl:problem name='primality_proof'>
  <msdl:header/>
  <msdl:body>
    <msdl:output name='d' />
    <msdl:post-condition>
      <om:MOBJ>
        <om:OMA>
          <om:OMS cd="relation1" name="eq" />
          <om:OMV name="d" />
          <om:OMA><om:OMS cd="numbertheory1" name="isprime" />
            <om:OMI>1234567891</om:OMI>
          </om:OMA>
        </om:OMA>
      </om:MOBJ>
    </msdl:post-condition>
  </msdl:body>
</msdl:problem>

```

5 Relation to Other MONET Ontologies

The Mathematical Problem Ontology is just one of the ontologies developed within the MONET project. Here we try to relate it to other relevant ontologies in the MONET architecture.

Query Ontology The mathematical problem ontology is part of the query ontology [6]. A MONET query might just boil down to a mathematical problem.

Mathematical Service Ontology The mathematical service ontology [5] has to describe services that can be matched against the query issued by the clients. A broker has to detect whether a service published using the mathematical service ontology language provides a solution to a problem described using the mathematical problem ontology language.

Explanation Ontology The explanation ontology [7] is concerned with how the broker or planner motivates the choice of a service as answer to a certain problem. The meta information provided in the mathematical problem description that was used for matching against the mathematical service descriptions has to be traced for producing the explanation to the client.

6 Conclusion

The languages described in this document are designed to form an extensible basis for describing mathematical problems within the MONET framework. Documented W3C XML schemas defining these languages have been produced.

References

- [1] Arjeh Cohen and Henk Barendregt. Electronic Communication of Mathematics, interacting Computer Algebra Systems and Proof Assistants. *Journal of Symbolic Computation*, 2001.
- [2] Arjeh Cohen, Hans Cuypers, and Hans Sterk. *Algebra Interactive*. Springer-Verlag, 1999.
- [3] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, September 1991.
- [4] The MONET Consortium. MONET Architecture Overview. Technical Report Deliverable D04, The MONET Consortium, December 2002. Available from <http://monet.nag.co.uk>.
- [5] The MONET Consortium. Mathematical Service Description Language: Final version. Technical Report Deliverable D14, The MONET Consortium, March 2003. Available from <http://monet.nag.co.uk>.
- [6] The MONET Consortium. The MONET Mathematical Query Ontology. Technical Report Deliverable D13, The MONET Consortium, March 2003. Available from <http://monet.nag.co.uk>.
- [7] James Davenport. The Mathematical Explanation Ontology: draft. Technical Report Deliverable D07, The MONET Consortium, March 2003. Available from <http://monet.nag.co.uk>.
- [8] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.