

**Version:** 1.0  
**Date:** November 2002

## **Mathematical Problem Ontology: Initial Draft**

Olga Caprotti<sup>1</sup> and Mike Dewar<sup>2</sup>

<sup>1</sup>RISC-Linz    <sup>2</sup>NAG

**Deliverable D05 (Public)**

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Mathematical Problem Ontology</b>	<b>2</b>
2.1	Common Metadata . . . . .	3
2.2	Problem Context . . . . .	3
2.3	Find . . . . .	3
2.4	Prove . . . . .	4
2.5	Decide . . . . .	5
2.6	Lookup . . . . .	5
<b>3</b>	<b>Relation to Other MONET Ontologies</b>	<b>6</b>

## 1 Introduction

This document describes one part of the MONET architecture [3] 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”, or “decide whether 1234567891 is prime”. Because these questions arise in the context of web services, we will consider an XML language in which to express mathematics and additional metadata that further specify requirements for the problem (such as accuracy, solution domain or arithmetic). In fact, this additional metadata is to be found in the mathematical service ontology (see MPDL in [2]) managed by the broker.

The exact syntax by which a client expresses in XML an instance of a problem should be coordinated with the syntax used for the documents that make up other parts of the MONET architecture, e.g. the mathematical problem description library, the algorithms library and more generally the MSDL language.

## 2 Mathematical Problem Ontology

For the time being, a mathematical problem consists of a *computational directive* to be performed on one or more OpenMath objects. Additionally the problem may contain extra information that is mathematically relevant for solving the problem.

One possible way to markup a problem is a very straightforward, for instance the following fragment of XML

```
<matpro>
  <description>The factors of the polynomial P </description>
  <input name="P">
    <OMOBJ>
      <OMA><OMS cd="polyr" name="" />x^4-4x^2</OMA>
    </OMOBJ>
  </input>
  <find>
    <output name="F">
      <OMOBJ><OMA><OMS cd="poly" name="squarefree" />
        <OMV name="P" />
      </OMOBJ>
    </output>
  </find>
</matpro>
```

represents the problem of finding the squarefree factorization, named  $F$ , of the input polynomial  $P = x^4 - 4x^2$ .

A more compact way does not make the input polynomial explicit in the problem description:

```
<matpro>
  <description>The factors of the polynomial P </description>

  <find>
    <output name="F">
      <OMOBJ><OMA><OMS cd="poly" name="squarefree"/>
        <OMA>x4-4x2</OMA>
      </OMOBJ>
    </output>
  </find>
</matpro>
```

## 2.1 Common Metadata

The metadata which is used to describe in a human-readable way the problem to be solved is common to all kind of directives. This includes for instance an optional description of the problem in natural language or XML (e.g. docbook, xhtml).

## 2.2 Problem Context

A mathematical problem typically lives in a certain context, e.g. a certain 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 by the broker in order to supply the necessary knowledge.

Contexts should be given in one of the available 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 should be designed so that it is extensible by other namespaces.

The context can be intended as part of the input to the problem or can be a constraint to the solution of the problem (in the sense of pre- or post- conditions). This is modelled on the way the mathematical problem descriptions will be structured [2].

## 2.3 Find

A problem to find involves a certain unknown which has to be computed/evaluated given certain data. The input and the output could be subject to conditions or constraints. There are finer distinctions in terms of how to connotate *find* when the intention is *solve* (find an assignment of) or *compute* (evaluate for certain values of the input parameters) or *simplify* (find a normal form for, or a simpler form with respect to a reduction  $R$ ). We can in future decide to allow finer distinctions between these various flavours.

In terms of OpenMath, it seems at first plausible to expect that the content model of a `find` directive is an application. However there are cases in which it is also possible to have a binding object. For instance, to choose the variable acting as unknown in an equation with parameters, it is enough to universally quantify the parameters in a solve problem. To manipulate a first-order formula, it is enough to issue a simplify request that generates, say a disjunctive normal form, or a skolemized version (the OpenMath CDs might then contain a measure of complexity in the definition of these symbols).

**Note of the author** I find it quite interesting that an example of “solve” is a SAT problem where the input is a propositional formula and the result is a boolean assignment for the propositional variables. At first I thought I might classify such a problem under a prove problem (for prove satisfiable) but it seems more appropriate to have it as a solve since it returns an object and not a proof. Basically, a good hint to where a problem lies is give by the expected result of the problem.

## 2.4 Prove

A problem to prove returns a proof of the validity of an assertion  $A$  (an OpenMath objects representing a proposition, `boolean` in STS) or a counter example in case the assertion is false. The broker however does not certify how the service will behave in case of false assertions. When uncertain of the validity of the assertion, the client may first issue a *decide* problem and only later ask for the proof.

A proving problem has a directive `prove` which clearly indicates that the expected result is a proof. The result 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. In the MONET architecture however, one can also expect to receive a proof that uses a third party encoding such as OMDoc.

```
<matpro>
  <description>IsPrime </description>
  <prove logic="ICC|FOL" calculus="ND">
    <output name="F">
      <OMOBJ><OMA><OMS cd="numberth1" name="isprime"/>
        <OMI>1234567891</OMI>
      </OMA>
    </OMOBJ>
  </output>
</prove>
</matpro>
```

The logic and the calculus chosen for the proof can be used by the broker to select the appropriate service (different logical system will use different CDs). Within the OpenMath ontology we will identify a set of CDs that are suited for the representation of proofs and possibly as alternative also indicate a fragment of OMDoc (OMDoc2 is designed modularly).

The values of the calculus attribute could be the names of the CDs or CDGroups that define calculi, see [1] for a survey.

## 2.5 Decide

A problem to decide returns a boolean answer about the validity of an assertion (an OpenMath object representing a proposition, of type `boolean` in STS) subject to decidability. The broker however does not certify how the service will behave in case of undecidability.

```
<matpro>
  <description>Primality Test</description>

  <decide>
    <output name="F">
      <OMOBJ><OMA><OMS cd="numbertheory1" name="isprime"/>
        <OMI>1234567890</OMI>
      </OMA>
    </OMOBJ>
  </output>
</decide>
</matpro>
```

As we have seen above, the notion of context discussed in Section 2.2 is particularly relevant for this class of problems.

## 2.6 Lookup

How does the lookup activity fit in this ontology? Is it an object of the ontology or is it just a broker functionality which MONET will support?

Here are some example from Arjeh Cohen in order to form an opinion.

- Give me some information on a mathematical notion (could this generalize to a *define* or *explain* directive). First inclination: OpenMath CD. Second: how do we relate OM CDs to further information, without compromising the ‘unambiguity’ of the CD symbols?

This leads to a data base search.

(e.g. factorization, use google, find biggest number factored, maybe want to say: *integer factorization*...forming a bit of context for the notion we are looking for, or look up Weisstein’s encyclopedia online at <http://www.ericweisstein.com/encyclopedias/books/Factorization.html>

- What does the series 1,2,4,8,16,... mean? Does it have a name? (hint: use Sloane’s integer number sequence) This asks for ‘inductive’ information rather than the usual deductive.

**2.6.0.1 Comment from MCD:** I’m not convinced that lookup is a directive in the same way the “prove”, “find” etc. are. I can find an integral by looking up its value in a suitable database, decide whether an integer is a Carmichael number by trying to look it up in a suitable list etc... I think its more a solution strategy than anything else.

### 3 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 query ontology is directly dependent on the mathematical problem ontology since it deals with queries that involve mathematical problems.

**Mathematical Service Ontology** The mathematical service ontology [2] 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 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.

### References

- [1] W. Bibel and E. Eder. Methods and calculi for deduction. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming - Vol 1: Logical Foundations*. Oxford, Clarendon Press, pages 67–182. 1993.
- [2] The MONET Consortium. Mathematical Service Description Language: Initial Draft. Technical Report Deliverable, The MONET Consortium, November 2002. Available from <http://monet.nag.co.uk>.
- [3] The MONET Consortium. MONET Architecture Overview. Technical Report Deliverable, The MONET Consortium, November 2002. Available from <http://monet.nag.co.uk>.