Towards A Mathematical Services Description Language

Olga Caprotti
Wolfgang Schreiner
Research Institute for Symbolic Computation
Johannes Kepler University, A-4040 Linz, Austria
{ocaprott,schreine}@risc.uni-linz.ac.at

February 26, 2002

Abstract

As a foundation for a framework for brokering mathematical services distributed among networked servers, we propose a language that describes the mathematical problems solved by the services. This mathematical service description language extends WSDL, the web service description language. Servers register their problem solving capabilities expressed in this language with a "semantic broker" to which clients submit corresponding task descriptions. The broker (possibly in cooperation with a deduction system) determines the suitable services and returns them to the client for invocation. This mechanism thus hides from the client the actual implementation of mathematical services and focuses on the semantical aspects.

Keywords: online mathematical services, web service description language, openmath.

1 Introduction

In the recent years, mathematical software systems, i.e., computer algebra systems like Maple, Mathematica, or GAP, automated theorem provers like Otter, PVS, or Theorema, or systems for numerical scientific computing like Matlab are slowly migrating from being standalone desktop tools to be used by a human to computational engines used as "mathematical services" by other programs. There are numerous examples in the literature of scientific software needing access to computer algebra methods [22], of computer algebra systems requiring advanced reasoning capabilities [14, 13] or vice-versa of automated theorem provers exploiting computer algebra systems for their system solving facilities [5, 17]. Some systems provide built-in communication protocols for interfacing to third-party software, whereas others have to be accessed via the standard input/standard output interface. Frequently however these approaches suffer from the inadequate specification of the functionality of a system. The result may take surprising forms, a challenge for the interpretation capabilities of the client software sometimes forced to mimic a human user [21].

Furthermore the Internet and the World Wide Web have further extended the range of possibilities by enabling networked access to mathematical services hosted in distant locations. Clients may use these services via remote access protocols [25, 19, 8]. The more networked mathematical services become available, the less it is desirable that each speaks its own language, therefore common mathematical communication protocols are required [3, 16]. Since more and more clients are programs rather than humans, mechanisms to convey some understanding on the meaning of the offered services need to be worked out.

The technologies for accessing mathematical services over the network focus mainly on the problem of portable data representation (OpenMath [12], MathBus [26], MathWeb [1], JavaMath [19]). iamc does provide a protocol for the exchange of meta-data about the names of services offered and about accepted data formats, but it does not yet explicitly address the problem how a client can learn about the service semantics. Only the MathWeb activities [16] deal with the semantic level focusing on the integration of automated theorem provers.

Thus, while the basic communication protocols for mathematical data have emerged, there has been only little work on the higher protocol levels that make mathematical services actually usable for automated tools without human insight. In particular, additional protocol levels have to be designed in order to support programming of mathematical applications that employ mathematical services distributed over the Internet. The final goal of a mathematical service description language is to be able to write such programs from an abstract point of view, without knowing about the actually available implementations. In this paper we advocate a mathematical service description format that abstracts from the details of an implementation and focuses on the actual *problem* being solved. The paper begins with a section on the state of the art of semantic web standards and mathematical communication. Then follows the core section in which the design of a language for describing mathematical services is presented. The final sections outlines briefly some of the technology that can support the implementation of the design.

2 State of the Art

There are several standard languages and protocols relevant to our purposes, from general ones for describing Web resources to specific ones developed for interacting mathematical services. Since it is important to build on existing standards for enhancing compatibility, we shortly overview the state of the art.

2.1 Semantic Web Standards

The World Wide Web in its trend towards a Semantic Web is becoming a network in which the resources from being machine-readable are turned to machine-understandable, namely they can be processed automatically rather than just displayed, are more usable for humans, and can be used to offer a variety of services.

The Extensible Markup Language XML [7] takes care of the syntactical aspect by providing a universal structured format for documents and data. An XML Document Type Definition (DTD) or better an XML Schema, describes the syntax of a class of documents which becomes in this way machine-readable by any tool that can fetch the DTD. XML has applications in various domains, for instance MathML, the Mathematical Markup Language [18] recommended by the Web Consortium is XML designed for mathematical data. XML serves as an anchor for various related W3C standards and recommendations ranging from styling language to data transport standards.

A large portion of the commercial world, including Microsoft and IBM, has agreed on an XML grammar for describing network services as collections of endpoints capable of exchanging messages. The Web Services Description Language (WSDL) [10] specification is currently a W3C note, however there are several tools and programming interfaces for developers to quickly setup services that use the WSDL protocol. WSDL conveys to the client the meta-data required to set up the communication with the server, this includes for instance the type of physical communication protocol to be used to setup the connection (SOAP 1.1, HTTP GET/POST, and MIME are supported bindings). The exchanged messages can be described using XML Schema or any other type definition language. WSDL itself does not prescribe how to publish or publicize such service descriptions. There are however other specification languages that take care of the registration aspects, for instance the Universal Description, Discovery and Integration language defines one framework for cataloging and dispatching WSDL descriptions for creating directories of Web Services.

2.2 Mathematical Communication

Communicating mathematics over the Web is a very active field of study which encompasses various different efforts: from designing languages to act as standard for the representation of mathematics, to defining protocols for bringing interactive computations online. We will consider here those languages and protocols relevant to our purposes.

We mentioned earlier MathML, a W3C recommendation for capturing both the structure and the content of mathematical notation [18]. While this approach has evolved from a focus on the presentation of mathematical objects, the OpenMath standard has been explicitly designed to represent meaning [11]. Both MathML and OpenMath aim to enhance mathematical communication by enabling mathematical data to be served, received, and processed electronically.

OpenMath in particular is equipped for conveying mathematical concepts arising from all areas of mathematics [11, 9] and can be used e.g. to express formal mathematical objects like theorems and proofs, as well as the usual mathematical expressions handled by computer algebra packages. OpenMath, for instance, can be used to represent preconditions and postconditions of algorithms, since these can be seen as mathematical formulae in a well-defined logical setting. The OpenMath representation of mathematics relies on a small set of "expression trees" constructors (application, binding, attribution and error), on some basic objects (byte-arrays, strings, integers, IEEE floats, variables), and on the usage of symbols defined in Content Dictionaries (CDs). These are publicly available, and extensible, collections of mathematical definitions, a sort of XML dictionary of mathematics [23].

The OpenMath Document Specification OMDoc [20] is an XML DTD for mathematical documents based on OpenMath. It has been designed to represent formal mathematical knowledge as a graph of related concepts, in which theorems are linked to definitions, and are collected/organized in mathematical theories. OpenMath Documents are for instance an exported format for the specifications expressed in the Common Algebraic Specification Language CASL. OMDoc adds a pragmatic layer on top of OpenMath; on the lowest level, however OpenMath attribution objects can be used to represent the same type of information.

The standardization of data formats alone is not sufficient for the electronic communication of mathematics since computations are an essential aspect of mathematics on a computer. Protocols and Application Program(ming) Interfaces (APIs) by which mathematical computational services can be offered and invoked are needed and have been developed.

The Internet Accessible Mathematical Computation project IAMC [25] defines (after the model of the HyperText Transfer Protocol HTTP) the application layer protocol MCP by which a computational server and a client can communicate with each other. A server can be referenced by a client via a URL; the subsequent client-server dialogue comprises computation and control requests from the client, corresponding responses from the server, and also server requests to the client that solicit information from the end user. The protocol allows the server to offer different data formats and also to send informal service descriptions, e.g.

```
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)
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)
```

IAMC is thus mainly designed to allow human users to access remote services via (typically Web-based) clients.

JavaMath is an API [19] mainly oriented towards the development of Java-based clients that access mathematical services using OpenMath as an exchange language. Within a computational session for a specific mathematical engine, the programmer may send a native command, retrieve an OpenMath object or native output as a result of a computation, and assign an OpenMath object to a local variable. While this approach allows fast prototyping of applications, it still lacks the framework for the engines to provide and define a semantically "certified" service.

MathWeb is a more established project [1] which defines a software bus for accessing mathematical services based on a CORBA-like model where a central broker is in charge of routing and of supplying authentication information to mathematical services. These are provided by theorem provers and recently also by computer algebra systems. However, this approach is only sufficient for relatively closed projects with fixed and well-known services. Recent work has explored the use of KQML as a language of communicating mathematical "agents" can mutually offer and access services [3, 16].

Summarizing, the technologies for accessing mathematical services over the network focus mainly on the problem of portable data representation and there has been only little work on the higher-level protocols towards machine-understable mathematical services. In the next section we propose a layered mathematical service description language designed to support abstract programming of mathematical applications employing mathematical services of various nature.

3 Description of Mathematical Services

From a pragmatic point of view, the description of a mathematical service is intended to facilitate the discovery of a certain computational facility, either abstractly from a specific mathematical question or more informally by a name of an algorithm or via any other item that might identify a piece of software. Keeping this in mind, the overall design of the description language is structured in layers which proceed from an abstract view, namely the problem addressed by the mathematical service, all the way down to the hostname-port address of the service as depicted in Figure 1.

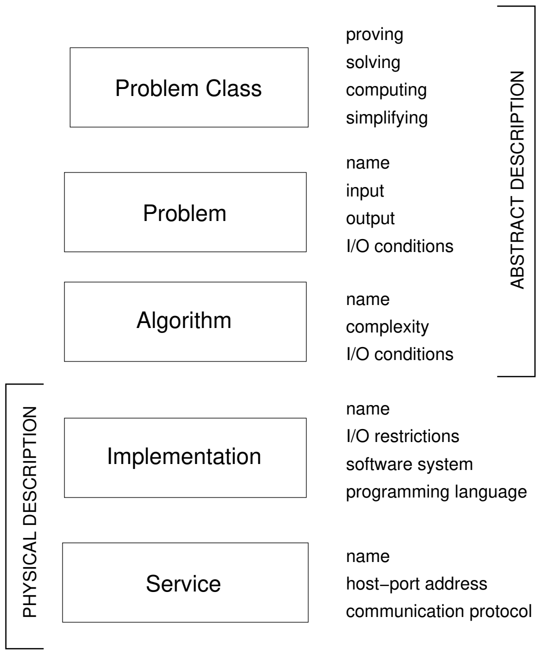

Figure 1: Mathematical Service Description Language Layers

Problem Class We begin by attempting a classification/taxonomy of problem classes. This first coarse distinction is intended as a guideline on how to further specify a specific problem instance. The underlying idea is that a *computing* problem might have a different description from a *proving* problem. For instance, stating that a problem belongs to the class of *simplification* problems, amounts to choosing an equivalence relation $\equiv$ and a *reduction* relation since these are key components of the problem class definition.

A *problem class* description defines the structure of classes of problems (parametrized by a logical system L and a theory T of L) including for instance

- *Computing*, namely the problem of carrying out operations in a domain, in a theory T i.e., given a term f with a free variable x and a value a in a properly chosen domain, return the value of f with a substituted for x;

- *Simplifying*, namely the problem of obtaining simpler yet equivalent objects in a domain, i.e., given a term t and a set of simplification rules R, return a term $t' = t$ that satisfies some "simplicity" condition,

- *Solving*, i.e., given a set of formulas $F[\vec{x}]$ in a particular logic, find (some or all) substitutions $\vec{a}$ in a chosen domain that make $F[\vec{a}]$ true in a certain theory;

- *Proving*, i.e., given a closed formula F in a particular logic, determine whether the formula is true in a particular domain or not. In particular, for some logic, this is equivalent to returning a proof.

In the logic broker architecture [2], a uniform way of specifying abstractly a service employs the notion of consequence relation [4]. A server providing a certain service S, is expected to register a *specification* L_S of the service S that includes the specification of the associated *consequence relation* $(L, \vdash)$ where L is a (finite? enumerable?) set of sentences, and $\vdash \subseteq 2^L \times L$ is such that, for all $\Gamma, \Delta \in 2^L$, $\alpha, \beta \in L$, $\alpha \in \Gamma$ implies $\Gamma \vdash \alpha$ (inclusion), $\Gamma \vdash \alpha$ implies $\Gamma \cup \Delta \vdash \alpha$ (monotonicity), and $\Gamma \vdash \alpha$ and $\Delta \cup \{\alpha\} \vdash \beta$ imply $\Gamma \cup \Delta \vdash \beta$ (Cut). Because this notion of consequence can be used for characterizing several known logics (including Linear Logic and non-monotonic logics) and in general, for semantics-independent classification of standard connectives (via equations on consequence relations), it is enough for formally specifying services provided by reasoning systems which are the focus of the logic broker architecture. Our approach aims at describing not only reasoning services (hence it must accommodate a description of consequence relations) but also computational ones.

Problem A *problem* is a concrete instance of a problem class. The specific format of the problem instance is defined by the problem class it belongs to, eventually with additional (optional) fields that enhance its formalization. The name of a problem description may be used by a task to refer to a particular problem in a direct way; this allows a broker to identify corresponding services easily.

Additionally, if we consider that mathematical services as intended in this paper are computational services provided by mathematical software, it is possible to formally describe the problem that is solved as inspired by formal methods. According to [15],

> the term formal methods refers to the use of techniques from logic and discrete mathematics in the specification design and construction of computer systems and software.

The goal of formal methods is to be able to mechanically check an argument without relying on human intuition. Formal methods have been used to determine consistency of descriptions or properties of a design, to clarify and facilitate communication of requirements, to satisfy standards or provide certification data. Formal methods heavily rely on formal languages in which notation and techniques are derived from logic and are supported by mechanized proof methods.

The motivation for including a formal definition is one of usage. The formal definition of a problem description can be used by a broker (possibly in collaboration with a deduction system) to identify services that satisfy a description given in terms of the content of a problem. A specification of a problem expresses constraints on what is to be computed in a style à la Floyd-Hoare by describing input and output parameters $\vec{x}$ and $\vec{y}$, their domains, and eventually a precondition $p(\vec{x})$ and a postcondition $q(\vec{x}, \vec{y})$. Notice that in order to formally define a problem, we will require a knowledge base of mathematics to use as reference for the mostly used concepts. Our preferred choice in this regard involves OpenMath.

For instance, a task requesting an integer GCD computation can be satisfied by the broker using a service advertising extended GCD computation. Although the names of the problems do not match, formal manipulation of the specification of the service allows to satisfy the request in an alternative way by reasoning on the preconditions and postconditions of the problem.

Algorithm An *algorithm* is a solution strategy for a problem (possibly, by strengthening the precondition or weakening the postcondition, of a more special one). Apart from referring to the solved problem, an algorithm description may contain information about its asymptotic time complexity and memory requirements, i.e., about its general dynamic behavior. Potentially, also a description of the algorithm in a formal algorithmic language may be added for reasoning about additional properties. Apparently, there may be multiple algorithms for a given problem.

This layer captures the abstract computational essence of a mathematical service, which often is the interface of an algorithm. Requiring a certain algorithm for solving a problem is the natural mathematical activity it aims to support.

Implementation An *implementation* is a particular realization of an algorithm as executable software, possibly with additional constraints on the input and additional possibilities for the output (error cases, implementation restrictions, etc). Apart from referring to the algorithm applied (or at least to the problem solved), an implementation description may contain additional information about the underlying software system, about its absolute runtime (relative to the machine speed) and its absolute memory requirements. Sometimes it is interesting to know the author of a certain implementation, and the date or version. This kind of meta-data is the object of this description layer.

Service A *service* is a set of implementations running on a particular machine; its description contains information about the underlying machine (contact address, computational performance, communication performance, memory equipment, ...), about the actual communication protocol for invoking and controlling the service (including the choice of synchronous or asynchronous communication, the possibilities for querying the status of a computation, the potential of aborting a computation, etc), for querying the status of the machine, and about the way in which service requests are handled (one at a time, in parallel up to some limit, etc). This description is mostly achieved using WSDL [10].

Summarizing, every description has a unique resource identifier (URI) and may embed a formal definition. Less sophisticated applications may associate a fixed meaning with a URI; this is analogous to the standard set of OpenMath CDs which informally agree on the meaning of every symbol. More advanced applications may refer to the formal definition to learn about unknown entities; this is analogous to KQML-based approaches for describing mathematical services. While the first strategy makes the description framework quickly usable, the formal later ultimately unleashes its power; combinations of both strategies (formal descriptions relying on informally defined entities) are also possible.

4 Technology

In this section we look at some of the technological issues involved in the design of the mathematical service description language presented so far.

In order to have a running service it would suffice to limit the description to the service layer, and in fact most available client-server architectures work at this level and hide the location details and the objects' encoding to the client. This is the level of WSDL. However, the additional descriptions are essential for enhancing the network of mathematical services by intelligent functionalities.

Our experiments with WSDL[1] have shown that although it is very convenient to setup a service using WSDL, and to generate automatically clients for a service described by a WSDL specification, in order to reason about the service, more information is needed, for instance on the type of the arguments. Currently we are investigating the possibility of using the extensibility fields of WSDL to convey additional meta-data, in particular on the type of the input and output (beyond the primitive types provided by XML schemas[2]).

As WSDL teaches, a specific XML language defined by a collection of XML schemata [24, 6] is a suitable language for describing the functionality of mathematical services. Each problem class can be associated with a specific namespace defined by an XML schema, so for instance there is a schema for a computing problem. This schema acts as a specification of the basic skeleton structure of the problem instance. Problem descriptions will also be defined by XML schemas, as to associate a namespace to each specific problem, for instance a real root isolation problem could be described using the schema identified by the URI `http://www.mathbroker.net/computing/algebra/realrootisolation`. Schemas for problem instances include the schema of the problem class they belong to so that the class requirements on the descriptions are satisfied.

As example, a fragment of the computing problem of real root isolation could be described in XML syntax roughly as in Figure 2. The descriptions of mathematical objects are given in two ways, directly embedded using OpenMath syntax like `<OMOBJ><OMV name="y"/></OMOBJ>` or as a reference to a URI where mathematical concepts can be retrieved from a knowledge base like MBase (e.g. `http://mbase.mathweb.org:8080/mbase/`). The formal definition of a problem (e.g. pre/post-conditions) may be encoded in OpenMath, but other XML representations of mathematics should also be possible.

[1]See `http://perseus.risc.uni-linz.ac.at:8080/openmath/index.html`

[2]More specifically, one could define an XML schema to type, say OpenMath recursive polynomials.

```
<problem xsi="http://www.mathematics.net/problem/computing/algebra/realroot">
     <class>
     <computingProblem xsi="http://www.mathematics.net/problem/computing">
     <name>Real root isolation</name>
     <comment encoding="xhtml">Real root isolation of a polynomial in Q[x]</comment>
     <input>
     <parameter encoding="openmath"><OMOBJ><OMV name="p"/></OMOBJ></parameter>
     <domain encoding="openmath"><OMOBJ><OMA><OMS name="PolynomialRingR" cd="polyr"/>
                                 <OMS cd="setname1" name="Q"/><OMV name="x"/>
                                 </OMA></OMOBJ></domain>
     </input>
     <output>
     <parameter encoding="openmath"><OMOBJ><OMV name="l"/></OMOBJ></parameter>
     <domain encoding="openmath"><OMOBJ><OMA><OMS name="list" cd="list"/>
                                <OMA><OMS name="isolating_interval" cd="real_roots"/>
                                <OMS  cd="setname1" name="Q"/></OMA></OMA><OMOBJ>
     </domain>
     </output>
     <pre><domain="FOL" link="http://mbase.mathweb.org:8080/mbase">
           ...p is square-free...</domain>
     </pre>
     <post><domain="FOL" encoding="openmath">
           ...the elements of l are disjoint open isolating
            intervals for the roots of p... </domain>
     </post>
     </computingProblem>
     </class>
```

Figure 2: Real Root Isolation Computing Problem

By a similar extension mechanism, it is possible to develop schemas for the specification of algorithms, implementations, and services.

5 Conclusion and Work Plan

We have described the design of a language for the description of mathematical services in a distributed networked environment. The language is structured in layers, each one capturing a particular aspect of a mathematical problem solution, implementation and service details. This structured approach has the advantage of allowing a modular development of the language by starting from the bottom layer, which is in part covered by existing standard languages, and then adding onto it.

The final goal of the description language is to support the electronic processing of mathematical services in the scenario where the Net is a global problem solving framework where each computation can be accessed/found by a multitude of entry points.

Our current work is the exact definition of the XML schemas that will correspond to the layers, the identification of some case studies that should serve as a guideline, and the study of available online knowledge bases of mathematics to be used as reference. We have begun with analyzing available web services standards such as WSDL and related ones. Our experiments have convinced us that WSDL alone does not suffice if we want to eventually be able to reason about mathematical services, to streamline them and to do mathematics online.

References

[1] Andreas Franke and Michael Kohlhase. System Description: MathWeb, an Agent-Based Communication Layer for Distributed Automated Theorem Proving. In Harald Ganzinger, editor, *16th Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 217–221. Springer, 1999.

[2] A. Armando and D. Zini. Interfacing Computer Algebra and Deduction Systems via the Logic Broker Architecture. In *Proceedings of the XXth Calculemus symphosium*, St. Andrews, Scotland, 6-7 August 2000.

[3] Alessandro Armando, Michael Kohlhase, and Silvio Ranise. Communication Protocols for Mathematical Services based on KQML and OMRS. In Manfred Kerber and Michael Kohlhase, editors, *Symbolic Computation and Automated Reasoning: The CALCULEMUS-2000 Symposium*, St Andrews, Scotland, August 6–7, 2000. A. K. Peters, Natick, MA.

[4] Arnon Avron. Simple consequence relations. *Information and Computation*, 92(1):105–140, 1991.

[5] C. Ballarin, K. Homann, and J. Calmet. Theorems and Algorithms: An Interface between Isabelle and Maple. In A.H.M. Levelt, editor, *Internat. Symposium on Symbolic and Algebraic Computation (ISSAC'95)*, pages 150–157, Montreal, Canada, July 10 - 12, 1995. ACM Press.

[6] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes. W3C candidate recommendation, World Wide Web Consortium, October 2000. `http://www.w3.org/TR/2000/CR-XMLschema-2-20001024`.

[7] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0. W3C recommendation, World-Wide Web-Consortium, October 2000. `http://www.w3.org/TR/REC-XML`.

[8] O. Caprotti, A. M. Cohen, H. Cuypers, M. Riem, and H. Sterk. Using OpenMath Servers for Distributing Mathematical Computations. In

Wei Chi Yang, Sung-Chi Chu, and Jen-Chung Chuan, editors, *ATCM 2000: Proceedings of the Fifth Asian Technology Conference in Mathematics*, pages 325–336, Chiang-Mai, Thailand, December 17–21 2000. ATCM Inc., USA.

[9] Olga Caprotti and Arjeh Cohen. On the role of OpenMath in interactive mathematical documents. *Journal of Symbolic Computation*, 2001. To appear.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, 15 March 2001.

[11] OpenMath Consortium. The OpenMath Standard, February 2000. O. Caprotti, D. P. Carlisle and A. M. Cohen Eds.

[12] Stephane Dalmas, Marc Gaetano, and Stephen Watt. An OpenMath 1.0 Implementation. In Wolfgang Küchlin, editor, *ISSAC'97 International Symposium on Symbolic and Algebraic Computation*, pages 241–248, Maui, Hawaii, July 21–23, 1997. ACM Press, New York.

[13] Martin Dunstan, Tom Kelsey, Steve Linton, and Ursula Martin. Lightweight Formal Methods for Computer Algebra Systems. In O. Gloor, editor, *International Symposium on Symbolic and Algebraic Computation (ISSAC'98)*, pages 80–87, Rostock, Germany, August 13–15, 1998. ACM Press.

[14] Martin Dunstan, Tom Kelsey, Ursula Martin, and Steve Linton. Formal Methods for Extensions to CAS. In *FM'99 — Formal Methods: World Congress on Formal Methods in the Development of Computing Systems*, volume 1709 of *Lecture Notes in Computer Science*, pages 1758–1777, Toulouse, France, September 20–24, 1999. Springer.

[15] Judy Crow et al. *NASA Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems, Volume II: A Practitioner's Companion*, volume 2. NASA Office of Safety and Mission Assurance, 1998.

[16] A. Franke, S. Hess, Ch. Jung, M. Kohlhase, and V. Sorge. Agent-Oriented Integration of Distributed Mathematical Services. *Journal of Universal Computer Science*, 5(3):156–187, March 1999. Special issue on Integration of Deduction System.

[17] J. Harrison and L. Théry. Extending the HOL Theorem Prover with a Computer Algebra System to Reason About the Reals. In J. J. Joyce

and C.-J. H. Seger, editors, *Higher Order Logic Theorem Proving and its Applications: 6th International Workshop, HUG'93*, volume 780 of *Lecture Notes in Computer Science*, pages 174–184, Vancouver, B.C., Canada, August 11-13, August 11-13 1993. Springer.

[18] Patrick Ion and Robert Miner. Mathematical Markup Language (MathML) Version 2.0. W3C proposed recommendation, World Wide Web Consortium, January 2001. http://www.w3.org/TR/2001/PR-MathML2-20010108/.

[19] JavaMath — Internet Accessible Mathematical Services, February 2001. http://javamath.sourceforge.net.

[20] M. Kohlhase. OMDoc: Towards an Internet Standard for the Administration, Distribution and Teaching of Mathematical Knowledge. In *AISC '2000, Artificial Intelligence and Symbolic Computation, Theory, Implementations and Applications*, Madrid, Spain, July 17–19, 2000.

[21] The Maple/PVS Project, January 2001. http://www-theory.dcs.st-and.ac.uk/info/example.

[22] Symbolic/Extended Symbolic Math Toolbox, February 2001. http://www.mathworks.com/products/symbolic.

[23] OpenMath Society. http://www.openmath.org.

[24] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema Part 1: Structures. W3C candidate recommendation, World Wide Web Consortium, October 2000. http://www.w3.org/TR/2000/CR-XMLschema-1-20001024.

[25] Paul Wang. Design and Protocol for Internet Accessible Mathematical Computation. In Sam Dooley, editor, *ISSAC'99 International Symposium on Symbolic and Algebraic Computation*, pages 291–298, Vancouver, Canada, July 28–31, 1999. ACM, New York.

[26] Richard Zippel. The MathBus. In *Workshop on Internet Accessible Mathematical Computation at ISSAC'99*, Vancouver, Canada, July 28–31, 1999. http://icm.mcs.kent.edu/research/iamc99proceedings.html.