OMDoc: An Infrastructure for OpenMath Content Dictionary Information

Michael Kohlhase

FB Informatik, Universität des Saarlandes, Saarbrücken
http://www.ags.uni-sb.de/~kohlhase

Abstract

The OpenMath framework for transmitting mathematical objects over the Internet relies on the concept of Content Dictionaries (CDs) to define the semantics of mathematical objects. This is an essential measure for establishing a meaningful communication amongst mathematical software systems (and humans).

Currently, the infrastructure for conceiving, administering, viewing CDs is limited to a file-based almost flat repository. In this paper, we propose to use the OMDoc extension of the OpenMath Xml encoding as an infrastructure to express and manipulate content dictionary information. OMDoc extends OpenMath by adding support for document markup (making the CDs more readable to the human user) and structured specification (making them more explicit, formal, and allow the user to reuse, and inherit CD information in a flexible, but well-defined way).

1 Introduction

It is plausible to expect that the way we do (conceive, develop, communicate about, and publish) mathematics will change considerably in the next ten years. The Internet plays an ever-increasing role in our everyday life, and most of the mathematical activities will be supported by mathematical software systems like computer algebra systems, theorem provers, mathematical knowledge bases, visualisation tools, etc; we will jointly call them *mathematical services*, if they are available on the Internet. These will be connected by a commonly accepted distribution architecture. From the experience with our MathWeb [11, 12] and similar experiments [7, 3] we can distinguish four kinds of problems that have to be overcome for realizing the vision of creating a world wide web of cooperating mathematical services. We will review them briefly and point out their relation to the OpenMath effort.

Syntax This is the very problem addressed by the OpenMath Xml encoding. With its imminent wider acceptance, and the emerging set of support tools, this problem will soon be solved.

Protocol/Control The problems of low-level communication and common control protocols have to be decided upon in the concrete application. OpenMath does not make a contribution here, but empirically, a very wide range of such protocols and architectures can be flexibly modelled by agent-oriented programming [15, 16]; in MathWeb we have experimented with the current de-facto standard Kqml [9] with good results. While a large-scale evaluation of this approach is still missing, we will not pursue this in this paper and refer the reader to [11, 2].

Semantics For the integration of systems it is crucial to specify concisely and without ambiguity the meaning of the exchanged formulae, i.e. there is the problem of establishing a semantics for the communicated mathematical objects. Otherwise the results of the integrated system can be arbitrary[1]. This is well-known as the so-called *ontology problem* in distributed artificial intelligence, the accepted solution to this is to either take recourse to a common set of concepts (the ontology) or to negotiate a private ontology for the communication. OpenMath recognises this and offers the mechanism of "content dictionaries": machine-readable, but normally *informal* definitions of the mathematical concepts involved.

Context The context problem is a variant of the semantics problem, i.e. in the communication of two mathematical software systems (or more generally agents) it is advantageous to maintain a sense of shared context or state. For instance, the state can be used to refer back to (parts of) previous formulae that are kept in the so-called context. Of course it is possible to eliminate context/state from the communication by retransmitting the relevant parts of the context, but this can lead to an exponential increase in costs. As a consequence almost all mathematical software systems use some form of context for the communication with the user. The OpenMath community has repeatedly discussed the concept of "dynamic CD" for this, but has not reached a conclusion yet.

All in all we see that the concept of content dictionaries in OpenMath addresses important problems (semantics and context) in the communication of mathematical objects over the Internet. Before we go on, let us briefly review the structure of CDs (see figure 1 for an example); they are basically collections of *symbol declarations*[2], without explicit structuring mechanism (the `CDUses` element in CDs, which

[1]Recall the recent incident of the NASA Mars mission, where NASA specified metric units but the contractor used pounds and inches (as a result the probe crashed on Mars.)

[2]The actual term used is `CDDefinition`, which is somewhat misleading, since e.g. the property of being commutative does not really *define* addition.

suggests a simple inclusion mechansim only specifies which examples are used in FMP tags or examples).

```
<CD>
 <CDName>arith1</CDName>
 <CDReviewDate>2000-09-01</CDReviewDate>
  ...
 <CDUses>
  <CDName>alg1</CDName>
   ...
  <CDName>relation1</CDName>
 </CDUses>
 <Description>
  This CD defines symbols for common arithmetic functions.
 </Description>

 <CDDefinition>
  <Name>plus</Name>
   <Description>
    An nary commutative function plus.
   </Description>
   <CMP>Addition is commutative, i.e. a+b=b+a</CMP>
   <FMP>∀ x,y.x + y = y + x</FMP>[3]
 </CDDefinition>
 ...
</CD>
```

Figure 1: An OpenMath Content Dictionary

Symbol declarations contain a description of the symbol together with a set of "*commented mathematical properties*" (CMP) and "*formal mathematical properties*" (FMP) in the form of OpenMath objects. As we see in figure 1, these can be arbitrary properties of the symbol (e.g. commutativity, but not associativity in the concrete example of arith1.ocd).

Note that this way of specifying the meaning of symbols is at best a partial solution, since the OpenMath framework does not offer any support for ensuring consistency, conciseness, or manipulation of CDs. In short, CDs are not *primary objects* in OpenMath:

- They are machine-readable (OpenMath specifies an Xml document type definition), but not machine-understandable, since the only part of their content that is fully formalised is administrative information like review- and expiry dates.
- They are not objects intended for communication themselves, but as background references defining the meaning of symbols, in particular for implementing phrase-books. OpenMath does provide an encoding of CDs as OpenMath objects, but there are no phrase-books or applications using that.

As a consequence, their content is largely informal (only humans can interpret them anyway). Unfortunately, OpenMath misses out on the opportunity to serve also as a standard for the theorem proving and software engineering (algebraic specification communities and program verification), where the precise (and machine-understandable, i.e. formal) specification of meaning is essential.

Changing this situation, by providing support for making the semantics of mathematical objects explicit or even formal is in some sense a much larger task, which involves formalising the mathematical theories behind them. We will show that an extension of OpenMath (called OMDoc), can be used as an infrastructure to support this task. In fact we claim (and we will substantiate this in section 6) that OMDoc is a good drop-in replacement for CDs in OpenMath.

[3]We have used the mathematical notation instead of the OpenMath representation to conserve space here.

2 OMDoc: OpenMath Documents

OMDoc extends the OpenMath standard to encompass mathematical documents (see [19]). This is suitable for our task since almost all of mathematics (specifications and properties of mathematical objects) is currently communicated in document form (publications, letters, e-mails, talks,...). As these documents have a complex structure of their own (often left implicit by typographic conventions), the specific task to be solved in the extension to OpenMath is to provide a standardised infrastructure for this as well.

As a consequence, OMDoc provides two sorts of markup devices; for

microstructure of mathematical texts this largely comprises the general pattern "definition, theorem, proof" that has long been considered paradigmatic of mathematical documents like textbooks and papers. Furthermore OMDoc supports auxiliary items like explanatory text, cross-references, exercises, applets, etc. See [21] for details. In a nutshell, OMDoc uses specialised Xml elements for all of these which may contain text representations (in form of CMPs) and formal versions (in the form of FMPs) of the mathematical content.)

macrostructure in terms of mathematical theories. For this, OMDoc techniques from the field of software engineering (see e.g. [22] for an introduction to algebraic specification), which focuses around the structured specification of structured formal theories of the behaviour of software and hardware.

In this paper, we will presuppose an intuition about the microstructure of mathematical texts, but concentrate on the treatment of mathematical theories (see section 3).

OMDoc is developed in the MathWeb project (see http://www.mathweb.org) to serve as

- a *communication standard* between mathematical services [11, 12].
- a data format that supports the *controlled refinement* from informal presentation to formal specification of mathematical objects and theories. Basically, an informal textual presentation can first be marked up, by making its discourse structure[4] explicit, and then formalising the textually given mathematical knowledge in logical formulae (by adding FMP elements; see [21] for details.
- a basis for *individualised (interactive) books*. OMDoc documents can be generated from the MBase service (see section 6) making use of the discourse structure information encoded in MBase. This application is joint work with Arjeh Cohen's OpenMath project at Eindhoven.

[4]classifying text fragments as definitions, theorems, proofs, linking text, and their relations; we follow the terminology from computational linguistics here.

We are currently evaluating OMDoc in the development of a user-adaptive interactive book including proof explanation based on IDA [5]. The OMDoc representation supports the formalisation of (parts of) the mathematical knowledge in IDA and makes it accessible to the ΩMEGA [4] mathematical assistant system, which can prove some of the problems either fully automatically (by proof planning) or in interaction with the user. This newly developed formal data (it is not present in IDA now) will enable the reader to read and experiment with the proofs behind the mathematical theory, much as she can in the present version with the integrated computer algebra system GAP [24].

In the context of this project, we have developed sophisticated mechanisms to translate OMDoc representations to output formats like LaTeX, HtML, MathMl, and the communication formats of mathematical services including theorem provers and computer algebra systems. This allows to browse OMDocs on the web and to produce nice printed output.

We also have first authoring tools for OMDoc that simplify generating OMDoc documents for the working mathematician. There is a simple OMDoc mode for `emacs`, and a LaTeX style [20] that can be used to generate OMDoc representations from LaTeX sources and thus help migrate existing mathematical documents. A second step will be to integrate the LaTeX to OpenMath conversion tools developed in the OpenMath Esprit project. Michel Vollebregt has built a program that traverses an OMDoc and substitutes various representations for formulae (including the Mathematica, GAP, and Maple representations) with the corresponding OpenMath representations.

3 Mathematical Theories in OMDoc

Traditionally, mathematical knowledge has been partitioned into so-called **theories** (see [10, 8] for an introduction), often centred around certain mathematical objects like groups, fields, or vector spaces. Theories have been formalised as collections of

- signature declarations (the symbols used in a particular theory, together with optional typing information).
- axioms (the logical laws defining the theory).
- theorems; these are in fact logically redundant, since they are entailed by the axioms.

In software engineering a closely related concept is known under the label of an (algebraic) specification, that is used to specify the intended behaviour of programs. There, the concept of a theory (specification) is much more elaborate to support the structured development of specifications. Without this structure, real world specifications become unwieldy and unmanageable.

In OMDoc, we support this structured specification of theories; we build upon the technical notion of a development graph [14], since this supplies a simple set of primitives for structured specifications and also supports the management of theory change. Furthermore, it is logically equivalent to a large fragment of the emerging Casl standard [6] for algebraic specification (see [1]).

All specification languages support mechanisms for specifying signature and axiom information, in particular, most also support abstract data types as a convenient shorthand for sets of inductively defined objects and recursive functions on these. We will subsume these under the label of simple theories and discuss their representation in OMDoc in the next section. After that we will use section 5 to discuss the issue of structuring and reusing theories by importing material from other theories.

4 Simple Theories

Theories are specified by the `theory` element in OMDoc. Since signature and axiom information are particular to a given theory, the `symbol`, `definition`, and `axiom` elements must be contained in a theory as sub-elements.

```
<theory xml:id="monoid">...
 <symbol name="monoid">
  <commonname xml:lang="en">monoid</commonname>
  <commonname xml:lang="de">Monoid</commonname>
  <commonname xml:lang="it">monoide</commonname>
  <signature system="simply-typed">
     set[any] -> (any -> any -> any) -> any -> bool
  </signature>
 </symbol>...
</theory>
```

Figure 2: An OMDoc symbol declaration

`symbol` This element specifies the symbols for mathematical concepts, such as 1 for the natural number "one", + for addition, = for equality, or `group` for the property of being a group. The `symbol` element has an `id` attribute which uniquely identifies it (in an OMDoc document).

This information is sufficient to allow referring back to this symbol as an OpenMath symbol. For instance the symbol declaration in figure 2 gives rise to an OpenMath symbol that can be referenced as `<OMS cd="monoid" name="monoid"/>`.

If the document containing this `symbol` element is stored in a data base system, the OpenMath symbol could be looked up by its common name. The type information specified in the `signature` element characterises a monoid as a three-place predicate (taking as arguments the base set, the operation and a neutral element).

`definition` Definitions give meanings to (groups of) symbols (declared in `symbol` elements elsewhere) in terms of already defined ones. For example the number 1 can be defined as the successor of 0 (specified by the Peano axioms). Addition is usually defined recursively, etc.

The OMDoc `definition` element supports several kinds of definition mechanisms specified in the `type` attribute, currently:

`simple` The `FMP` contains an OpenMath representation of a logical formula that can be substituted for the symbol specified in the `item` attribute of the definition.

`inductive` The formal part is given by a set of recursive equations whose left and right hand sides are specified by the `pattern` and `value` elements in `requation` elements. The termination proof necessary for the well-definedness of the definition can be specified in the `just-by` attribute of the definition.

`implicit` Here, the `FMP` elements contain a set of logical formulae that uniquely determines the value of the symbols that are specified in the `items` slot of the definition. Again, the necessary proof of unique existence can be specified in the `just-by` attribute.

`obj` This can be used to directly give the concept defined here as an OpenMath object, e.g. as a group representation generated by a computer algebra system.

Figure 3 gives an example of a (simple) monoid definition.

For a description of abstract data types see [21].

```
<definition name="mon.d1" item="monoid" type="simple">
 <CMP>
  A structure (M,*,e), is called a monoid,
  if (M,*) is a semi-group and e a unit for *.
 </CMP>
</definition>
```

Figure 3: A Definition of a monoid

5 Complex Theories and Inheritance

Not all definitions and axioms need to be explicitly stated in a theory; they can be inherited from other theories, possibly transported by signature morphism. The inheritance information is stated in an `imports` element.

`imports` This element has a `from` attribute, which specifies the theory which exports the formulae.

For instance, given a theory of monoids using the symbols `set`, `op`, `neut` (and `axiom` elements stating the associativity, closure, and neutral-element axioms of monoids), a theory of groups can be given by the theory definition using `import` in figure 4.

```
<theory xml:id="group">
 <imports xml:id="group.import" from="monoid" type="global"/>
 <axiom><CMP> Every object in
  <OMOBJ><OMS cd="monoid" name="set"/></OMOBJ> has an inverse.
 </CMP></axiom>
</theory>
```

Figure 4: A theory of groups based on that of monoids

`morphism` The morphism is a recursively defined function (it is given as a set of recursive equations using the `requation` element, described above). It allows to carry out the import of specifications modulo a certain renaming. With this, we can e.g. define a theory of rings given as a tuples $(R,+,0,-,*,1)$ by importing from a group $(M,\circ,e,i)$ via the morphism $\{M\mapsto R,\circ\mapsto +,e\mapsto 0,i\mapsto -\}$ and from a monoid $(M,\circ,e)$ via the morphism $\{M\mapsto R^*,\circ\mapsto *,e\mapsto 1\}$, where R^* is R without 0 (as defined in the theory of monoids).

`inclusion` This element can be used to specify applicability conditions on the import construction. Consider for instance the situation given in figures 5 and 6, where the theory of lists of natural numbers is built up by importing from the theories of natural numbers and lists (of arbitrary elements). The latter imports the element specification from the parameter theory of elements, thus to make the actualisation of lists to lists of natural numbers, all the symbols and axioms of the parameter theory must be fulfilled by the natural numbers. For instance if the parameter theory specifies an ordering relation on elements, this must also be present in theory `Nat`, and have the same properties there. These requirements can be specified in the `inclusion` element of OMDoc. Due to lack of space, we will not elaborate this and refer the reader to [14, 21].

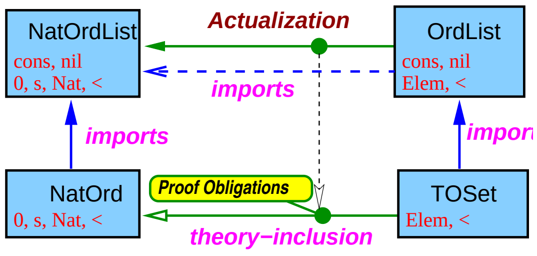

Figure 5: A Structured Specification of Lists

6 A System Infrastructure for OMDocs and Content Dictionaries

So far we have only taken a look at the OMDoc representation format for theories. To see the infrastructural advantage over the OpenMath CD format, let us compare e.g. the theories `nat-list.thy` given in figure 6 and the theory of rings alluded to on page 4 with hypothetical equivalent CDs.

readability We can use all the presentation mechanisms developed in the context of the IDA project mentioned in section 2; this allows to browse CD information as structured texts (as opposed to CD representations in Xml, which is supported by the OpenMath technology) on the web and to produce nice printed output.

definitions The definitions, axioms, and theorems given in these theories can be expressed as `CMP`s and `FMP`s in `CDDefinitions` in the CD.

In fact, we have developed Xsl style sheets that transform OMDoc theories (see section 4) into CDs and back [18]. This in particular allows to migrate all existing CDs into our proposed format.

redundancy Note that in the translation back into CD format we lose the information which indicates which of the properties are defining ones, and which just serve illustration purposes (they are technically redundant, and add no further specification, since they are implied by the defining properties).

reuse both theories reuse other theories, by structured inclusion. The ring theory makes use of the `monoid` and `group` theories importing material via a renaming (`CDUses` does not allow this), and `natlist.thy` uses the parameterised theory `list` instantiating it with natural

```
<theory xml:id="Param">
 <symbol name="Elem" type="sort"/>
 <symbol name="ord"/>
 <axiom ... ord is a partial order on Elem ... /axiom>
</theory>

<assertion xml:id="geq-ord" theory="nat-thy">
 <CMP><OMOBJ><OMS name="geq" cd="nat"/></OMOBJ> is a
  partial order on <OMOBJ><OMS name="nat" cd="nat"/><OMOBJ>
 </CMP>
<assertion>

<theory xml:id="List">
 <imports xml:id="List.im" type="global" from="Param"/>
 <symbol name="List-sort" type="sort"/>
 <symbol name="cons"/><symbol name="nil"/>
 <symbol name="ordered"/>
</theory>

<theory xml:id="nat-list.thy">
 <imports xml:id="nat-list.im-nat"
          type="global" from="nat-thy"/>
 <imports xml:id="nat-list.im-Element"
          type="local"  from="List">
  <morphism xml:id="elem-nat">
   <requation>
    <pattern><OMS cd="Param" name="Elem"/></pattern>
    <value>
     <OMOBJ><OMS cd="nat.thy" name="Nat"/></OMOBJ>
    </value>
   </requation>
  </morphism>
 </imports>
 <inclusion item="elem-nat-incl"/>
</theory>

<axiom-inclusion xml:id="elem-nat-incl"
                 from="nat.thy"
                 to="Param" by="ord-nat">
 <morphism xml:id="elem-nat-incl-morph"
           base="elem-nat"/>
</axiom-inclusion>
```

Figure 6: A theory of Lists of Natural Numbers

numbers. CDs for these theories would have to duplicate multiply used theories, which can lead to combinatorial explosion, especially in the case of parametric theories.

Another piece of infrastructure which can be used for providing OpenMath infrastructure for managing content dictionary information is the MBase system, a MathWeb service that acts as a distributed mathematical knowledge base. We will not present the MBase system here (see [13, 17]), but only look at some applications. MBase can be used to

- collect and integrate multiple CDs (in OMDoc representation), in particular it can be used as a web-based CD server.
- generate personalised sub-documents or linearisations of the structured data based on a user model. This supports browsing and manipulating CD information in large systems that reference a great number of CDs.
- answer high-level queries about the existence of specific symbols, e.g. in cases where the actual CD a symbol is defined in is unknown.

7 Conclusion

We have argued that for the specification and formalisation of the meaning of mathematical symbols we need more support than is currently available from OpenMath content dictionaries. We have proposed to use the OMDoc format – an extension to OpenMath that allows to represent the semantics and structure of various kinds of mathematical documents – as the basis for an infrastructure for conceiving, administering, viewing content dictionary information. As a document-centred format, OMDoc gives better readability than content dictionaries. With its structuring devices inspired by methods from the field of software engineering it allows for more concise and reusable specifications.

In the meantime, OMDoc is adopted as a data interchange format for theorem provers and program verification systems (including Isabelle, Ωmega, λ*Clam*, Imps, and InKa). All of these provide a substantial amount of formalised mathematical theories, that can be used as CD information in the OpenMath setting.

Of particular importance in this respect is the bridge between the Casl standard (Common Algebraic Specification Language) [6] currently under development in Saarbrücken and Bremen. This software engineering standard attempts to solve a similar goal as the OpenMath standard: the communication of meaningful mathematical objects over the net. While OpenMath has concentrated on standardising the representation of mathematical objects and web-based communication, Casl has concentrated on standardising the specification of (mathematical) theories (i.e. content dictionary information).

OMDoc borrows from both standards and combines the ideas in one system, thus it is well-suited as a basis for solving the semantics-problem mentioned in the introduction. Together with the MBase system, which can act as an interactive OMDoc/CD server (thus making CD information dynamic), OMDoc can eventually serve as a basis for the context problem mentioned there. This will however require a tighter integration with the protocol/control layer, which we intend to study in the near future taking [2] as a basis.

References

[1] Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy and D. Bert, ed.S, *Proceedings Workshop on Algebraic Development Techniques, WADT-99*. Springer, LNCS 1827, 2000.

[2] Alessandro Armando and Michael Kohlhase. Communication protocols for mathematical services based on kqml and omrs. In Manfred Kerber and Michael Kohlhase, eds., *CALCULEMUS-2000, Systems for Integrated Computation and Deduction*, St. Andrews, Scotland, 2000. AKPeters. forthcoming.

[3] Alessandro Armando and Daniele Zini. Towards Interoperable Mechanized Reasoning Systems: the Logic Broker Architecture. In A. Poggi, ed, *to appear on the Proceedings of the AI*IA-TABOO Joint Workshop 'From Objects to Agents: Evolutionary Trends of Software Systems'*, Parma, Italy, May 29–30, 2000.

[4] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann,

and V. Sorge. ΩMEGA: Towards a mathematical assistant. In William McCune, ed, *Proc. of the 14th Conference on Automated Deduction*, no. 1249 in LNAI, pp. 252–255, 1997. Springer Verlag.

[5] Arjeh Cohen, Hans Cuypers, and Hans Sterk. *Algebra Interactive!* Springer Verlag, 1999. Interactive Book on CD.

[6] Language Design Task Group CoFI. Casl — the CoFI algebraic specification language — summary, version 1.0. Technical Report, `http://www.brics.dk/Projects/CoFI`, 1998.

[7] Louise A. Dennis, Graham Collins, Michael Norrish, Richard Boulton, Konrad Slind, Graham Robinson, Mike Gordon, and Tom Melham. The prosper toolkit. In *Proceedings of the Sixth International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS-2000*, 2000.

[8] William Farmer. An infrastructure for intertheory reasoning. In McAllester [23], pp. 115–131.

[9] T. Finin and R. Fritzson. KQML — a language and protocol for knowledge and information exchange. In *Proceedings of the 13th Intl. Distributed Artificial Intelligence Workshop*, pp 127–136, Seattle, WA, USA, 1994.

[10] William Farmer, Josuah Guttman, and Xavier Thayer. Little theories. In D. Kapur, ed, *Proc. of the 11th Conference on Automated Deduction*, vol. 607 of *LNCS*, pp 467–581, Saratoga Spings, NY, USA, 1992. Springer Verlag.

[11] Andreas Franke, Stephan M. Hess, Christoph G. Jung, Michael Kohlhase, and Volker Sorge. Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, 5:156–187, 1999.

[12] Andreas Franke and Michael Kohlhase. System description: MATHWEB, an agent-based communication layer for distributed automated theorem proving. In Harald Ganzinger, ed, *Proc. of the 16th Conference on Automated Deduction*, no 1632 in LNAI, pp 217–221. Springer Verlag, 1999.

[13] Andreas Franke and Michael Kohlhase. System description: MBASE, an open mathematical knowledge base. In McAllester [23], pp. 455–459.

[14] Dieter Hutter. Reasoning about theories. Techical Report Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1999.

[15] Nicholas R. Jennings and Michael J. Wooldridge, eds. *Agent Technology : Foundations, Applications, and Markets*. Springer, Berlin, 1998.

[16] N. R. Jennings and M. Wooldridge. *Handbook of Agent Technology*, chapter Agent-Oriented Software Engineering. AAAI/MIT Press, 2000. forthcoming.

[17] Michael Kohlhase and Andreas Franke. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Comutation; Special Issue on the Integration of Computer algebra and Deduction Systems*, 2000. forthcoming.

[18] Michael Kohlhase. The OMDoc repository. Internet page at `http://www.mathweb.org/omdoc`.

[19] Michael Kohlhase. OMDOC: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *Proceedings AISC'2000*, 2000. forthcoming.

[20] Michael Kohlhase. Creating OMDOC representations from LaTeX. Internet Draft available at `http://www.mathweb.org/ilo/omdoc/latex2omdoc.ps`, 2000.

[21] Michael Kohlhase. OMDOC: Towards an OPENMATH representation of mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/ilo/omdoc`.

[22] J. Loeckx, H.-D. Ehrig, and M Wolf. *Specification of Abstract Data Types.* Teubner, Chichester, New York, Brisbane, 1996. ISBN 3-519-02115-3.

[23] David McAllester, ed. *Automated Deduction – CADE-17*, no. 1831 in LNAI. Springer Verlag, 2000.

[24] Martin Schönert et al. *GAP – Groups, Algorithms, and Programming.* Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1995.