Formal Representation Issues in an Open Mathematical Knowledge Base — Extended Abstract —

Michael Kohlhase
School of Computer Science
Carnegie Mellon University
Pittsburgh, USA
http://www.cs.cmu.edu/~kohlhase

July 9, 2001

Abstract

For the logical formulation of mathematical concepts, we propose a methodology for developing representation formalisms for mathematical knowledge bases. Concretely we propose to equip knowledge bases with a hierarchy of logical systems that are linked by *logic morphisms*. These mappings relativize formulae and proofs and thus support translation of the knowledge to the various formats currently in use in deduction systems. On the other hand they define higher language features from simpler ones and ultimately serve as a means to found the whole knowledge base in axiomatic set theory.

1 Introduction

Around 1994, an anonymous group of authors put forward the "QED Manifesto" [QED95], which advocates building up an open mathematical knowledge base (and supporting software systems) as a kind of "human genome project" for the deduction community (see `http://www.mcs.anl.gov/qed` for an overview of the QED activities). Unfortunately, the vision has failed to catch on in spite of a wave of initial interest. In our view this is due to the lack of supporting software, as well as to the ensuing debate on the "right" logical formalism. We will not concentrate on the motivation of a knowledge base system – this has been more thoroughly and more elaborately in the QED Manifesto [QED95], which we presuppose as a background reference to the discussion.

There are several attempts to build such knowledge base systems currently under way. The first that needs to be mentioned is the Automath project [dB80] (the precursor of all logical frameworks and formal mathematical knowledge base systems), and the MIZAR project [Rud92], that has compiled a very large body of formalized mathematics. Moreover, semi-automated reasoning systems usually store large amounts of mathematical data in a file-oriented library storage mechanism (see e.g. [Imp, Isa, PVS]). All in all, even if successful, these projects have only yielded mathematical content, and not an open web-based infrastructure for mathematical knowledge bases.

In the last years we have embarked on an infrastructure project to develop an open (not tied to a particular mathematical software system) mathematical knowledge base (using contemporary database and communication technology) system. The MBase system is a web-based, distributed knowledge base for mathematics that is universally accessible through MathWeb [FK99, FHJ+99]. The current implementation is still under development, it consists of the MBase server, which acts as a MathWeb service which communicates with other services through a system of *mediators*. The primary interface format of MBase is OMDoc [Koh00b, Koh01], an

XML-based representation language for MBase content. Since this is an extension of the emerging OpenMath standard [CC98] for web-based mathematics, its syntax is logic-independent. So the mediators can first do the logic-transformation, then generate the OMDoc representation, and then create the concrete input syntax of the respective reasoning system by invoking a standard XML style sheet processor with a specialized XSL style sheet.

Currently, connections to the theorem proving system ΩMEGA [BCF+97], INKA [HS96], PVS [ORS92], λ*Clam* [RSG98], TPS [ABI+96] and CoQ [Tea] systems are supported. Access for humans is possible via specialized knowledge base clients (e.g. browser services that display the knowledge base content), specialized query engines.

In this paper we will not describe the MBase system (see [FK00, KF00] for an overview or go to`http://www.mathweb.org/mbase` for a demo), but the problems of integrating formal knowledge from the theorem provers connected to MBase. In general we will follow the approach of theory and language interpretations, which allow to build a two-dimensional hierarchy of theories in MBase.

2 Logics, Morphisms and MBase Languages

The logical language supported by MBase is a polymorphically typed, sorted record λ-calculus modeled after the mathematical everyday language (often called "mathematical vernacular", e.g. [dB94]). It is a joint generalization of the ML-polymorphic λ-calculus with kinds as used in ISABELLE and Hashimoto & Ohori's polymorphic record calculus [Oho95]. Records allow a clean formalization of mathematical structures, such as groups or fields, polymorphism is needed to reuse definitions and theorems in the knowledge base and ensure a modular structure of the theory. Finally the mechanism of "kinds" adds to the practical expressivity of the polymorphism and is used in many theorem proving systems (λ*Clam*, ISABELLE,...). Finally, the MBase logic supplies the infrastructure for sorted λ-calculi (see section [KF00]). Conceptually, sorts are unary predicates (corresponding to often-used sets in mathematics) that are treated specially in the inference procedures (sorted matching and unification). This added structure leads to a more concise representation and a more guided search. For clients that cannot manipulate sorts, types, records, or higher-order quantification, the mediators built into MBase can relativize these language features away, retaining the intended meaning.

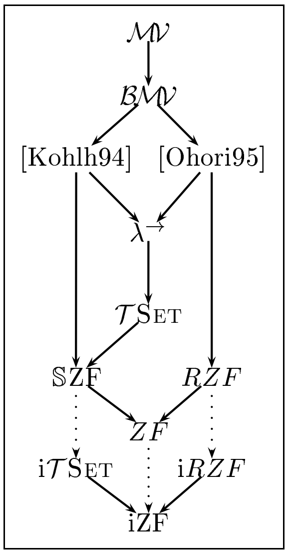

Figure 1: Hierarchy

We will use a variant of the theory interpretation approach proposed in [Far93] for **relativization** mappings, that can be used to transport meanings and proofs between logical formalisms. In fact, we can build a whole hierarchy of representation languages (see Figure 1), where relativizations can be used to arrive at various representation formalisms for mathematics, down to axiomatic (Zermelo-Fraenkel) set theory. Before we formally define the notion of relativization by the concept of **logical morphism** in the next section, let us discuss the consequences for the architecture of MBase.

The defining intuition for logic morphisms is that

> **Logic Morphisms Transport Proofs**: Let $\mathcal{F}:\mathcal{S} \longrightarrow \mathcal{S}'$ be a logic morphism and $\mathbf{A}$ an $\mathcal{S}$-theorem, then $\mathcal{F}(\mathbf{A})$ is an $\mathcal{S}'$-theorem.

This already suggests the logical structure of a mathematical knowledge base: Orthogonal to the usual theory hierarchy (induced by theory interpretation morphisms; we will not go into in this article, see [Far93]), there is a hierarchy of logical system induced by logic morphisms. In Figure 1, we have specified some of the logical systems we will discuss in this article.

Mathematical knowledge can be specified in any of the logical systems; it can be queried and retrieved in any logical system that is downward accessible from this one. Furthermore, communication of mathematical software systems is possible by way of the "least common denominator logic". This may seem as a severe restriction of applicability of the approach, but it is not since the set of logical systems and morphisms in the hierarchy is not necessarily fixed:

- A new logical system can be incorporated by specifying a logic morphism to any of the existing systems.
- A new logic morphism can be added, if it is consistent with the information already present in the structure, i.e. if it is *redundant*.

Of course these hierarchy extensions generate proof obligations (determining the logic morphism property and redundancy), which will have to be supported in a system like MBase. We leave a discussion of this to another article.

The practical usefulness of a language hierarchy will depend very much on the existence of such redundant morphisms. In particular for the "least-common-denominator" problem between languages $\mathcal{L}$ and $\mathcal{L}'$ we can have two kinds of situations:

- If there is a good and well-understood way to translate formulae from language $\mathcal{L}$ to $\mathcal{L}'$, then we can implement this as a redundant logic morphism in MBase bypassing the need of an intermediate "communication logic". Moreover, making the logic morphism available in MBase will allow other users to use it.
- If there is no such translation, or if it is very domain-specific, then (of course) logic morphisms will not help (only further research into the semantic relation between the logics and possible translations will).

In [KF00] we have described how this approach can be used to build up the logical hierarchy in Figure 1, in particular to build up typed λ-calculus ($\lambda^{\rightarrow}$) from axiomatic set theories like ZF [Fra22], and thus ground the hierarchy of representation languages in set theories.

3 Theory Hierarchies and the Development Graph

Orthogonal to the hierarchy of logical langauges, there is the theory hierarchy. Traditionally, mathematical knowledge has been partitioned into so-called **theories**, often centered about certain mathematical objects like groups, fields, or vector spaces. Theories have been formalized as collections of

- **signature declarations** (the symbols used in a particular theory, together with optional typing information).
- **axioms** (the logical laws of 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**, which is used to specify the intended behavior of programs. There, the concept of a theory (specification) is much more elaborated to support the structured development of specifications. Without this structure, real world specifications become unwieldy and unmanageable. Therefore MBase supports the structured specification of theories building upon the technical notion of a **development graph** [Hut99], since this supplies a simple set of primitives for structured specifications and also supports management of theory change. Furthermore, it is logically equivalent to a large fragment of the emerging Casl standard [CoF98] for algebraic specification (see [AHMS00]).

The main idea is that 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 which acts much like the language morphism discussed in the last section. For instance, given a theory of monoids using the symbols `set`, `op`, `neut` (and axioms stating the associativity, closure, and neutral-element axioms of monoids), a theory of groups can be given by importing the monoid theory and simply stating an axiom for the existence of inverses. Similarly, 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 $\{M \mapsto R^*, \circ \mapsto *, e \mapsto 1\}$, where R^* is R without 0 (as defined in the theory of monoids).

Following Hutter's development graph [Hut99], we can use the knowledge about theories to establish so-called **inclusion morphisms** that establish the source theory as included (modulo renaming by a morphism) in the target theory. This information can be used to add further structure to the theory graph and help maintain the knowledge base with respect to changes of individual theories. In effect, any axiom in the source theory there must be a theory in the target theory (i.e. provable). This information can be used to trace dependency information in the knowledge base and and support management of theory change (e.g. to determine what theorems have to be re-proven if we change a definition).

In MBase and in its communication representation OMDoc, the theory structure is represented by with the OpenMath content dictionary mechanism. OpenMath **content dictionaries** are XML-based representations of signature information (names and types of mathematical concepts expressed as OpenMath **symbols**); they contain symbol names and so-called mathematical properties, that restrict the semantics of the concepts, but no principled treatment of concept definitions or logical entailment of mathematical properties. The mathematical theories described in this section are a superset of the information in OpenMath content dictionaries, so we view them as a drop-in replacement, and directly use the OpenMath reference mechanism for mathematical objects in OMDoc and MBase, while keeping the richer theory structure we have described in this section for theory maintenance and exploration (see [Koh00a] for a discussion).

4 Practical Representation Issues

In this section we will talk more about the practical problems encountered in a series of experiments of connecting the theorem proving systems Ωmega [BCF+97], InKa [HS96], Pvs [ORS92], λ*Clam* [RSG98], Tps [ABI+96] and CoQ [Tea] systems are supported. to the MBase system by equipping them with an OMDoc interface.

The first observation in the interpretation is that even though the systems are of relatively different origin, their representation languages share many features

- Tps and Pvs are based on a simply typed λ-calculus, and only use type polymorphism in the parsing stage, whereas Ωmega and λ*Clam* allow ML-style type polymorphism.
- Ωmega, InKa and Pvs share a higher sort concept, where sorts are basically unary predicates that structure the typed universe.
- Pvs and CoQ allow dependent- and record types as basic representational features.

but also differ on many others

- InKa, Pvs, and CoQ explicitly support inductive definitions, but by very different mechanisms and on differing levels.
- CoQ uses a constructive base logic, whereas the other systems are classical.

At one level, the similarities are not that surprising, all of these systems come from similar theoretical assumptions (most notably the Automath project [dB80]), and inherit the basic setup (typed λ calculus) from it. The differences can be explained by differing intuitions in the system design and in the intended applications.

Following recent work on the systemization and classification of λ-calculi [Bar92], we have started to ground these languages in language hierarchy like the one shown in Figure 1. The structural similarities between theories and logical languages and their structuring morphisms allow to re-use the OMDoc/MBase theory mechanism for language definition: The logical symbols and language constructs can be defined just like other (object-level) symbols/concepts. As a consequence, the development of the OMDoc interface to the theorem provers mentioned above included the specification of the representation language as a theory (which could be used as an integrated documentation). The structured theory mechanism can now be used to re-use and inter-relate the various representation formats between the theorem provers. For instance the simply typed λ-calculus can be factored out (and thus shared) of the representation languages of all of the theorem proving systems above[1]. This makes the exchange of logical formulae via the OMDoc format very simple, if they happen to be in a suitable common fragment: In this case, the common (OpenMath/OMDoc) syntax is sufficient for communication.

On the other hand, the experiments have shown that the basic assumptions in the MBase/OMDoc/OpenMath framework will have to be extended to fully accommodate the current practice in theorem proving representations. For example, Pvs allows the specification of parameterized theories. This feature is mainly used to compensate for the lack of type polymorphism in the representation language, but is more general than this. If the number of theory instances (i.e. symbols, where the formal theory parameters are instantiated with concrete values) is finite, the theory can be accommodated in the development graph model, by generating finitely many explicit instance theories and the appropriate inclusion morphisms. However Pvs also allows to quantify over variables that are later used to instantiate theory parameters; in this case the number of theory instances is potentially infinite, and cannot be directly be represented in MBase/OMDoc. Unfortunately this problem cannot simply be fixed by adding additional representation concepts to OMDoc, since it breaks a fundamental assumption in OpenMath, namely that theories can always explicitly be represented (as content dictionaries), and that this can be done ahead of using them. Thus extending MBase/OMDoc to this form of mathematical practice will reconciling even something as fundamental as the OpenMath standard with mathematical practice. Note that the concept of parametric theories cannot easily be dismissed as representational aberrations; the work on so-called functors in the Theorema project `http://www.theorema.org` views parametric theories as the principal building blocks and successfully uses this higher-order structure to guide theorem proving.

5 Conclusions

We have presented some ideas how to cope with the complexities of formal representations in open mathematical knowledge bases that arises from the fact that such systems must accept input from widely differing representation systems.

We have proposed a logic-based approach based on theory and logic morphisms to structure the (in practice rather large) set of theories and representation languages into structured (inheritance) graphs, and we have shown how this structure can be exploited to generate mediators that translate mathematical knowledge between mathematical software systems. While this picture is compelling in theory, its practical promise has still to be realized by an implementation and larger experiments.

On the practical side we have presented the results of a series of experiments of building mediators (in this case translators to the OMDoc representation format) by hand for specific systems. We have seen that a unified representation format can be used to pinpoint representation language similarities and enable system communication, but that some fundamental problems remain.

References

[ABI+96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem-proving system for classical type theory. *Journal of Automated Reasoning*, 16:321–353, 1996.

[1] We will make the language definitions of the theorem provers available only shortly

[AHMS00] Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy and D. Bert, editors, *Proceedings Workshop on Algebraic Development Techniques, WADT-99*. Springer, LNCS 1827, 2000.

[Bar92] Henk P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford University Press, 1992.

[BCF+97] 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, editor, *Proceedings of the 14th Conference on Automated Deduction*, number 1249 in LNAI, pages 252–255, Townsville, Australia, 1997. Springer Verlag.

[CC98] Olga Caprotti and Arjeh M. Cohen. Draft of the Open Math standard. The Open Math Society, `http://www.nag.co.uk/projects/OpenMath/omstd/`, 1998.

[CoF98] Language Design Task Group CoFI. Casl — the CoFI algebraic specification language — summary, version 1.0. Technical report, `http://www.brics.dk/Projects/CoFI`, 1998.

[dB80] Nicolaas Govert de Bruijn. A survey of the project AUTOMATH. In R. Hindley and J. Seldin, editors, *To H.B. Curry: Essays in Combinator Logic, Lambda Calculus and Formalisms*, pages 579–606. Academic Press, 1980.

[dB94] N. G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In R. P Nederpelt, J. H. Geuvers, and R. C. de Vrijer, editors, *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, pages 865 – 935. Elsevier, 1994.

[Far93] William M. Farmer. Theory interpretation in simple type theory. In *HOA'93, an International Workshop on Higher-order Algebra, Logic and Term Rewriting*, volume 816 of *LNCS*, Amsterdam, The Netherlands, 1993. Springer Verlag.

[FHJ+99] 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.

[FK99] Andreas Franke and Michael Kohlhase. System description: MATHWEB, an agent-based communication layer for distributed automated theorem proving. In Harald Ganzinger, editor, *Proceedings of the 16th Conference on Automated Deduction*, number 1632 in LNAI, pages 217–221. Springer Verlag, 1999.

[FK00] Andreas Franke and Michael Kohlhase. System description: MBASE, an open mathematical knowledge base. In David McAllester, editor, *Automated Deduction – CADE-17*, number 1831 in LNAI, pages 455–459. Springer Verlag, 2000.

[Fra22] Adolf Abraham Fraenkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, 86:230–237, 1922.

[HS96] Dieter Hutter and Claus Sengler. INKA - The Next Generation. In M.A. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th Conference on Automated Deduction*, number 1104 in LNAI, pages 288–292, New Brunswick, NJ, USA, 1996. Springer Verlag.

[Hut99] Dieter Hutter. Reasoning about theories. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1999.

[Imp] The imps online theory library. Internet interface at `http://imps.mcmaster.ca/theories/theory-library.html`.

[Isa] The isabelle online theory library. Internet interface at `http://www4.informatik.tu-muenchen.de/~isabelle/library`.

[KF00] 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.

[Koh00a] Michael Kohlhase. OMDoc: An infrastructure for OPENMATH content dictionary information. *Bulletin of the ACM Special Interest Group on Symbolic and Automated Mathematics (SIGSAM)*, 34(2):43–48, 2000.

[Koh00b] Michael Kohlhase. OMDoc: An open markup format for mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/omdoc`.

[Koh01] Michael Kohlhase. OMDoc: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In Eugenio Roanes Lozano, editor, *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2000*, number 1930 in LNAI. Springer Verlag, 2001.

[Oho95] Atsushi Ohori. A polymorphic record calculus and its compilation. *ACM Transactions on Programming Languages and Systems*, 17(6):844–895, 1995.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. PVS: a prototype verification system. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 748–752, Saratoga Spings, NY, USA, 1992. Springer Verlag.

[PVS] Pvs libraries. `http://pvs.csl.sri.com/libraries.html`.

[QED95] The QED manifesto. Internet Report `http://www.rbjones.com/rbjpub/logic/qedres00.htm`, 1995.

[RSG98] Julian D.C. Richardson, Alan Smaill, and Ian M. Green. System description: Proof planning in higher-order logic with λ*clam*. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in LNAI. Springer Verlag, 1998.

[Rud92] Piotr Rudnicki. An overview of the mizar project. In *Proceedingsof the 1992 Workshop on Types and Proofs as Programs*, pages 311–332, 1992.

[Tea] Coq Development Team. *The Coq Proof Assistant Reference Manual*. INRIA. see `http://coq.inria.fr/doc/main.html`.