QED Reloaded: Towards a Pluralistic Formal Library of Mathematical Knowledge

MICHAEL KOHLHASE and FLORIAN RABE
Jacobs University Bremen
http://kwarc.info

Proposed in 1994, the "QED project" was one of the seminally influential initiatives in automated reasoning: It envisioned the formalization of "all of mathematics" and the assembly of these formalizations in a single coherent database. Even though it never led to the concrete system, communal resource, or even joint research envisioned in the QED manifesto, the idea lives on and shapes the research agendas of a significant part of the community

This paper surveys a decade of work on representation languages and knowledge management tools for mathematical knowledge conducted in the KWARC research group at Jacobs University Bremen. It assembles the various research strands into a coherent agenda for realizing the QED dream with modern insights and technologies.

1. INTRODUCTION

Even though short-lived and ultimately unsuccessful, the QED project and manifesto [Ano94] of 1994 were enormously influential on automated reasoning. The QED manifesto urged the automated reasoning community to work towards a universal, computer-based database of all mathematical knowledge, strictly formalized in logic and supported by proofs that can be checked mechanically. The QED database was intended as a communal resource that would guide research and allow the evaluation of automated reasoning tools and systems. This database was never realized – not even started as a concrete collection or system – but the discussion about it influenced a whole generation of researchers and practitioners in the automated reasoning and formal methods communities. Indeed the QED idea – if not the system and project – is very much alive in the community today. On the twentieth anniversary of the QED manifesto we survey the state of formalization and proving and the chances of realizing the twenty-year-old dream after all.

We find that many of the problems that prevented the QED project from succeeding were already part of the initial discussion on the QED mailing list. The

The work presented here has partially been funded by the DFG under grants KO 2428/13-1, KO 2428/9-1, and RA-1872/3-1. The intuitions and results presented here summarize the efforts of the KWARC group at Jacobs University Bremen and a number of external collaborators as listed in the various cited publications. The MathHub.info system was built in collaboration with Mihnea Iancu and Constantin Jucovschi.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
©2014 by the authors

pertinent topics were *i*) the choice of the *root logic*, its calculi and proof formats, in particular regarding types and partial functions, *ii*) the question whether mathematics truly needed formalization and consequently the acquisition of funding. While the latter point was probably the main practical obstacle, the former was and still is arguably the biggest theoretical stumbling block for a *universal* library of formalized mathematics. There was a consensus that a root logic was needed, i.e., a logic in which all of mathematics would be formalized. But most groups only advocated the logic of their own system as the most feasible candidate. The discussion ended in a proposal by Bob Boyer – the driving force behind QED – to use Primitive Recursive Arithmetic [Sko67] (PRA), a quantifier-free formalization of the natural numbers. But while PRA is a canonical choice of root logic due to its extreme simplicity, it was generally regarded as insufficiently expressive for *practical* formalization. For example, in the recollection of the first author, Peter Andrews expressed this sentiment with the sarcastic comment that "if PRA, then we should gödelize it first!".

Actually, Andrews' position in the discussion was much more constructive than this quote suggests and has motivated much of the authors' research. In one post [And94] to the QED mailing list, he advocated to "accept diversity": a library organization that accepts theorems and proofs in *any* logic after that logic has itself been submitted to the database. The reasoning behind this is that the various logical languages then – and currently – in use have evolved for good reasons: Every such system caters to a different aspect of formalization or area of application. However, at the time of QED the development of logical frameworks [Pfe01], which can give a theoretical basis for this vision, had just begun.

Since then, the plurality of logic designs and of the corresponding theorem proving technologies has become even more formidable. Today there are about half-a-dozen libraries with $\sim 10^5$ formalized significant mathematical theorems, such as the ones of Mizar [TB85], Coq [Coq14], and the HOL systems [Har96]. These include deep theorems such as the Odd-Order Theorem [Gon+13] or the Kepler Conjecture [Hal05]. But each library is based on a separate root logic, all of which are mutually incompatible, which leads to duplication of work and missed opportunities for knowledge transfer.

In his post [And94], Andrews argues for another so far-unrealized goal: "I envision the QED library as a vast database and *a large number of associated utilities* for accessing, displaying, translating, manipulating, and using the items in the database" (emphasis ours). This places the utilities at the same level as the formalizations and not at the level of the individual systems contributing to the QED database. This makes sense if we consider the need for additional tools that integrate formalization systems with symbolic computation and large scale knowledge management systems – a very real need as evidenced by the formation of the CALCULEMUS and MKM (Mathematical Knowledge Management) communities, respectively, in the last two decades.

In 2007, Freek Wiedijk identified two further reasons for the failure of QED [Wie07]. Firstly, even though the root logics were adopted because of their expressivity, they still lack *practical* expressivity in comparison with conventional mathematics. He gives four deep but non-problematic mathematical theorems from calculus, abstract algebra, category theory, and set theory and concludes that no

system allows proving all of them based on natural definitions and notations. This is compatible with the observation that each root logic has particular strengths and no single root logic can be satisfactory for all applications.

Secondly, the large libraries that have been developed lack library organization strategies to keep them coherent in the presence of many contributors. Wiedijk observes that today's large libraries are either well-curated and integrated or have a large number of contributors. It is possible to develop management support for libraries such as refactoring and dependency tracking. But if these features were provided generically by the QED database, it would significantly ease the implementation burden on each system.

Contribution and Overview. We begin with a thorough review of the state of the art in Section 2. We describe the existing tools, their root logics, and their libraries and focus on the integration problems posed by the plurality of root logics.

Our contribution is a novel design that we propose for a **universal community-driven QED database**. Contrary to all existing large QED-style databases, ours is systematically **pluralistic**: We accept the diversity of logics and build it into the representation language and its semantics from the beginning. This is the key step to resolve the problem of incompatible root logics.

A pluralistic QED database must include the definitions of the various root logics (and their semantics) and be able to understand formalizations in any of them. Therefore, the main design choice of our database is to use a meta-language that allows representing both the logics and the formalizations. We use our OMDoc/Mmt language as this meta-language, which we present in Section 3.

In Section 4, we describe the architecture of our MathHub.info system. It uses an API-style implementation of our meta-language (cf. Section 3.2) as the central component and complements it with scalable user management, repository manager, a web interface for trans-library navigation, and a mathematical search engine. MathHub.info subsumes our envisioned QED database but goes beyond it by including also informal mathematical documents.

We are convinced that these advances are crucial stepping stones towards building a QED-like resource in a future community effort. Therefore we have started to build a nucleus of a QED-inspired database (Section 5) that embraces logical plurality. Presently, MathHub.info already serves several major mathematical libraries including those of 2 QED-style proof assistants. Section 5.1 describes how we can represent the root logics and the libraries uniformly in OMDoc/Mmt and MathHub.info, and we discuss the most promising methods to obtain to integrate existing libraries within our system in Section 5.2. Section 6 concludes the article.

2. STATE OF THE ART IN FORMALIZATION AND DEDUCTION

The systematic formalization of mathematical knowledge and its semantics go back at least to the seminal work by Russell and Whitehead [WR13]. The use of computer systems started in the 1950s and 1960s focusing on designing foundations that combine human and machine-friendliness. *Automated theorem proving*, going back to ideas by Newell, Simon, and Davis, has been most successful for first-order predicate logic and related languages. For more expressive languages that more adequately model mathematical knowledge, best results were reached in the au-

tomated *verification of human-written proofs*, going back to ideas McCarthy, de Bruijn, Milner, and Martin-Löf.

Formalization pays off almost exclusively at large scales due to the high level of theoretical understanding and practical investment that they require from both developers and users. Therefore, today's successful projects are built on double-digit person years of investment. Even though this would naturally call for a community effort, the last few decades have seen increasing specialization into **isolated, mutually incompatible systems** and **incompatible overlapping libraries of formal knowledge**.

Moreover, during that time the advances in computer and internet technology have dramatically changed our expectations regarding scalability. Many requirements have become critical that are not anticipated in the designs of formal systems such as system interoperability and massively-multi-user collaboration.

2.1 Deduction Systems

Foundations. The notion of a foundation comes from the debate about the foundations of mathematics at the beginning of the last century, which tried to find a logical basis for mathematics and came up with first-order logic + set theory as an answer. *Philosophically*, a foundation should be as simple as possible, since it has to be accepted wholesale because we cannot determine the consistency of the foundation itself. *Practically*, a foundation should be expressive enough to allow structurally adequate and convenient formalizations.

The diverse group of foundations in today's QED-motivated systems are the result of trying to optimize this trade-off. All of these systems are based on a **fixed foundation**, i.e., a fixed root logic in which all formalizations in that system are stated. Most of these derive either from constructive type theories, higher-order logic, or first-order set theory.

The constructive type theories are mostly based on Martin-Löf type theory [ML74] or the calculus of constructions [CH88] and make use of the Curry-Howard correspondence [CF58; How80] to treat propositions as types. Systems include Nuprl [Con+86], Agda [Nor05], Coq [Tea], and Matita [Asp+06]. The second group of systems go back to Church's higher-order logic [Chu40] and use the LCF architecture [Mil72] to represent theorems as objects of an abstract type. Systems include HOL4 [**HOL4:on**], ProofPower [Art], Isabelle/HOL [NPW02], and HOL Light [Har96]. Notably, only Mizar [TB85] and Isabelle/ZF [PC93] are based on variants of axiomatic set theory and thus most similar to the foundation of conventional mathematics.

The foundations of the PVS system [ORS92] includes a variant of higher-order logics but with a significantly extended type system. The IMPS system [FGT93] is based on a variant of higher-order logic with partial functions. The foundation of ACL2 [KMM00] is an untyped language based on Lisp.

Monolithic vs. Homogeneous vs. Heterogeneous Methods. In principle – and according to mathematical folklore – all of mathematics can be formalized *monolithically* in a giant collection of first-order formulas based on a small collection of set-theoretic axioms. In practice – e.g. for a QED-like project, the monolithic approach is infeasible, since it lacks the structuring devices needed for coping with

the size and complexity of the collection, with development workflows, and the constantly changing presentation of mathematical knowledge.

Therefore, conventional mathematics uses what the authors call the **heterogeneous method**. Going back to the works by Bourbaki [Bou64], this method focuses on defining *theories* and stating every result (definitions, theorems, notations, etc.) in the smallest possible theory. This allows using *theory morphisms* to move results between theories in a truth-preserving way [FGT92]. Consequently, while all mathematics can be reduced to first principles, it is usually carried out in highly abstracted settings that hide the foundation.

Yet, virtually all formalizations in QED-motivated systems are based on the **homogeneous** method, which models mathematical knowledge using only conservative extensions (e.g., definitions, theorems) of the fixed foundation. For this purpose, most systems support complex conservative extension principles, such as type definitions in the HOL systems, provably terminating functions in Coq or Isabelle/HOL, or provably well-defined indirect definitions in Mizar.

Many QED systems support heterogeneous reasoning in some way, e.g., using theories and locales in Isabelle, parametric theories in PVS, modules and records in Coq, structures in Mizar. IMPS [FGT93] even uses the heterogeneous method systematically as the central design choice.

Moreover, many homogeneous formalizations employ the heterogeneous method implicitly. Here, first one introduces a new construct (e.g., the real numbers) by defining it in terms of existing ones (e.g., Cauchy sequences). Then one expands the definition while proving the characteristic theorems about the new construct. Finally, afterwards only those theorems are used, and the definitions are never expanded again. In particular, it becomes irrelevant, which out of several possible definitions (e.g., Dedekind cuts) was used. Indeed, despite the huge philosophical and practical differences between the foundations of QED systems, their libraries routinely use implicit heterogeneity to abstract from the foundational details and to create high-level settings for practical formalizations. However, while the heterogeneous method makes this abstraction explicit by introducing a theory (e.g., the second-order theory of the real numbers), it remains implicit in homogeneous formalizations.

Incompatibility. The combination of fixed foundation and homogeneous method has become the dominant approach in the QED area. This stands in contrast to the success of the heterogeneous method in conventional mathematics as well as in software engineering and algebraic specification. In particular, the latter fields systematically use formal module systems to build large theories out of little ones, e.g., in SML [Mil+97] and ASL [SW83], respectively.

Therefore, a lot of expensive formalization work is needed just to build the setting of interest (e.g., the real numbers) as a conservative extension of the fixed foundation. Adding insult to injury, the subsequent formalizations (e.g., of theorems about the real numbers) are actually less valuable than systematically heterogeneous formalizations would be: It becomes virtually impossible to reuse theorems within a library, let alone across libraries. Therefore, almost all current systems are mutually incompatible, with only a few ad hoc translations between them (e.g., [KW10; KS10]).

Moreover, while a fixed foundation is reasonable for an individual deduction system, it is counter-productive for a universal library of mathematics because different domains may require different foundations. Similarly, different communities involved in the formalization effort may favor different foundations (maybe only because of the formalization styles or workflows they support). Thus the library level of QED would profit from **foundational pluralism**, i.e., the ability to support multiple foundations in a single universal library.

2.2 Formal Libraries

Overview. All major deduction systems provide support for managing formalizations in "libraries" and collect some kind of library of formalizations. Often a certain basic library is loaded upon startup, and the user can load additional libraries on demand. The library mechanism can be decentralized with users developing and/or hosting individual libraries or centralized with a committee collecting and possibly curating the library.

A very sensitive issue here is backwards compatibility, i.e., the question whether a library is still readable after upgrading the main system. Only for centralized libraries, this can be guaranteed by the system developers. For example, Isabelle updates turned out to be one of the major problems in the L4 verification project [Kle+10] (7 years, 390000 lines of Isabelle/HOL).

The Isabelle and the Mizar groups maintain one centralized library each – the "Archive of Formal Proof" [AFP] and the "Mizar Mathematical Library" [MizLib], respectively. The Coq group maintains a similar set of contributions. These libraries contain individual formalizations with relatively few interdependencies.

Highly-integrated libraries are usually found as part of a single formalization project whose size required the development of a separate library. Even though these libraries started as auxiliary devices, they are valuable results in their own right – maybe even more valuable than the primary formalization. Examples are Tom Hales's formalizations in HOL Light for the Kepler conjecture [Hal05] and Georges Gonthier's work in Coq for the recently proved Feit-Thompson theorem [**Gonthier+:mcpoot13**]. John Harrison's formalizations in his HOL Light system [**Harrison:hltp11**] and the NASA PVS library [**PVSlibraries:on**] have a similar flavor although they were not motivated by a single theorem but by a specific application domain. The latter is one of the biggest decentralized libraries, whose maintenance is disconnected from that of the system.

All of the above, use the homogeneous method with implicit heterogeneity. Gonthier's Coq work in the Mathematical Components project [**MathComponents:on**] does so most systematically by using record types as an analogue to theories.

In principle, highly integrated libraries can be developed best with the heterogeneous method. This was pursued in the IMPS library [**ImpsKB:on**], (by the authors) in the LATIN logic library [Cod+11], and in the TPTP library [SS98] of challenge problems for automated theorem provers. However, none of these enjoys the level of interactive theorem proving support developed for individual fixed foundations.

The OpenTheory format [Hur09] offers some support for heterogeneity in order to allow moving theorems between systems for higher-order logic (specifically HOL Light, HOL4, and ProofPower). It provides a generic representation format for

proofs within higher-order logic that makes the dependency relation (i.e., the operators and theorems used by a theorem) explicit and thus permits heterogeneity. The OpenTheory library comprises several theories that have been obtained by manually refactoring exports from HOL systems.

Library Integration. There are several facets of library integration. Firstly, one can *refactor a single library* to increase reuse through modularity, sharing, and inheritance. Typically, this amounts to using the heterogeneous method. Secondly, one can *align two libraries*. Here alignments are special relations that pair each concept of one library with the corresponding concept in the other one. Alignment relations may range from "subsumes the intent of" to "are equivalent and proofs can be shared". Thirdly, one can *merge two libraries*. Usually, this requires translating the libraries into a common language and then identifying and eliminating overlap between them.

No strong tool support is available for any of these facets. The state-of-the-art for refactoring a single library is manual ad hoc work by experts, maybe supported by simple search tools (often text-based). Merging libraries can hardly be attempted because the state-of-the-art is still short of satisfactory translations into common languages.

This is despite the large need for more integrated and easily reusable large libraries. For example, Alan Bundy's research group is working on evolution of higher-order ontologies for physics. They need a way to integrate the libraries HOL light, which are the best existing ones for real analysis, with Isabelle/HOL, which provides better user interface and integration with external solvers than HOL Light. Similarly, in Tom Hales's Flyspeck project [Hal05], his proof of the Kepler conjecture is formalized in HOL Light. But it relies on results achieved using Isabelle's reflection mechanism, which cannot be easily recreated in HOL Light. And these are only the integration problems between two systems using the same root logic!

Library Exports. In most cases, integration attempts falter already when trying to access the library in the first place. The libraries consist of text files in **source languages** optimized for fast and convenient writing by human users. These differ from the internal languages of the respective deduction systems, which correspond more closely to the documented foundation of the system.

Consequently, highly non-trivial algorithms for parsing, type reconstruction, and theorem proving have been developed to build the corresponding data structures inside the tools. This has the effect that for each library, there is essentially only a single system able to read its surface language: the respective deduction system. In particular, there are usually no separate translators from surface to system languages: deduction systems are typically realized as read-evaluate-print interfaces to the foundation, optimized for batch-processing input files, and appear to the outside as monolithic black boxes. Said surface language parsing facilities are usually tightly integrated and practically inseparable from the deductive core of the system. Thus using the respective deduction for parsing and then exporting libraries in easily machine-readable formats – system languages is the only way to obtain machine-level access to deduction system libraries.

Many deduction systems do not specify machine-processable system languages, let alone an exporter. And even where they do – for example, Mizar, HOL Light,

and Coq provide idiosyncratic, system-near exports – the exports have two problems.

Firstly, the use of the homogeneous method means that the export contains the elaborated system language and not the high-level surface structure that would be more valuable for reuse. For example, in the case of Mizar, it proved notoriously difficult [DW97; BK07; Urb06] until the proposers obtained an export [Ian+13] that they could actually make use of in their applications.

Secondly, the system languages and export facilities/code quickly become out-of-date as new features are added to the main system. The only exception are exports that are actively maintained by the main developers, but this is rarely the case. Even the Mizar XML export has gotten somewhat out-of-sync recently although the XML export was tightly integrated with (and thus essential for) the main system.

Library Imports. Even deduction systems that have an export facility usually do not have a corresponding import facility that would read the exported library back into the system, let alone convert the library into the structured surface language originally used for writing it. Admittedly, an import at this level creates little value for an isolated system. To let users interoperate with libraries of other systems however this is exactly what is needed – the surface language level is where they think, author, and update. The complexity of the conversion task is one of the reasons we see so little interoperability of systems.

There is however, a middle ground: if the export supports "source references", i.e. if every fragment in the system language representation of the library contains pointers back to the exact range of the source – in surface language – that was translated to that fragment, then at least conflict markers or notifications of changes can be localized in the source – and thus be integrated into the source management facilities of the library system.

Library Translations. For importing fragments of a source library into a different (target) library, we need an export facility for the source library, an import facility for the target library, and additionally one of two things: a logic translation into the target system or a logical framework that can handle libraries in multiple logics. Both are rare. Therefore, there are only a few examples of bridging libraries between two large deduction systems.

A small number of library bridges have been realized using ad-hoc logic translations, typically in special situations. [KW10] translates from HOL Light [Har96] to Coq [Coq14] and [OS06] to Isabelle/HOL. Both translations benefit from the well-developed HOL Light export and the simplicity of the HOL Light foundation. [KS10] translates from Isabelle/HOL [NPW02] to Isabelle/ZF [PC93]. Here import and export are aided by the use of a logical framework to represent the logics. The Coq library has been imported into Matita, aided by the fact that both use very similar foundations. The OpenTheory format [Hur09] facilitates sharing between HOL-based systems but has not been used extensively.

The second approach requires a logical framework in which the source logic can be represented. Then it is straightforward to map the source library to the library mechanisms of the framework. The framework can serve as a uniform intermediate data structure via which other systems import libraries. This approach was used by the proposers in [Ian+13] for Mizar and in [KR14] for HOL Light. These used

the logical framework LF [HHP93] and made the libraries available to knowledge management services. Another example is the recent Dedukti system [BCH12], which imports, e.g., Coq and HOL Light into a similar logical framework, namely LF extended with rewriting.

The Library Integration Problem. Even when an export-import pair is available, it is usually still very difficult to integrate libraries due to what we dub the *library integration problem*.

The prevalent use of the homogeneous method means that concepts that are interesting to reuse in a different library are usually defined as conservative extensions. Their properties are proved as theorems in the single fixed foundation rather than as axioms in a dedicated theory. Any heterogeneity remains implicit in the export and cannot be reconstructed during the importing. This is disastrous for integration in the typical case where different libraries use different definitions (e.g., Dedekind cuts vs. Cauchy sequences). Thus, a logic translation will usually not map the real numbers of one system to the real number of another system.

This problem would not exist if all libraries were heterogeneous: For example, if two deduction systems used explicit theories of the real numbers that abstract from the precise (and possibly different) definitions, the theorems proved in these theories could be moved across systems easily.

Very little work exists to address this problem. In [OS06], some support for library integration was present: Defined identifiers could be mapped to arbitrary identifiers ignoring their definition. No semantic analysis was needed because the translated proofs were rechecked by the importing system anyway. This approach was revisited and improved in [KK13], which systematically mapped aligned the concepts of the basic HOL Light library with their Isabelle/HOL counterparts and proved the equivalence in Isabelle/HOL. The was further improved in [GK14] by using machine learning to identify large sets of further alignments.

The OpenTheory format [Hur09] provides representational primitives that, while not explicitly using theories, effectively permit heterogeneous developments in HOL. The bottleneck here is manually refactoring the existing homogeneous libraries to make use of heterogeneity. We recently sketched a partial solution aimed at overcoming the integration problem in [RKS11].

2.3 Logical Frameworks

In response to the plurality of logical systems, logical frameworks have been introduced. These are formalisms in which the syntax and semantics of logics (and similar languages) can be defined. [Pfe01] gives an overview.

LF [HHP93] is a logical framework based on the dependently-typed λ-calculus. It uses the judgments-as-types methodology. In particular, logic definitions usually use a declaration $\texttt{proof} : \texttt{form} \to \texttt{type}$ such that $\texttt{proof}\, F$ is the type of proofs of F. Twelf [PS99] is the most mature concrete implementation of LF. It includes a theorem prover for meta-theorems, i.e., theorems *about* the defined logics. A wide variety of logics have been defined in Twelf, e.g., in the LATIN library [Cod+11].

Dedukti [BCH12] implements LF modulo rewriting. By supplying rewrite rules (whose confluence Dedukti assumes) in addition to an LF theory, users can give more elegant logic encodings. A number of logic libraries have been exported to

Dedukti, which is envisioned as a universal proof checker.

Isabelle [Pau94] implements intuitionistic higher-order logic, which (if seen as a pure type system with propositions-as-types) is rather similar to LF. Isabelle includes an LCF-style interactive theorem prover and a tactic language for proving object theorems, i.e., theorems *within* the defined logic. Despite being logic-independent, most of the proof support in Isabelle is optimized for individual logics defined in Isabelle, most importantly Isabelle/HOL and Isabelle/ZF.

All logical frameworks allow represent logics as theories. This allows using the heterogeneous method to build logics from small components and to express logic translations as theory morphisms in the logical framework. However, only the LATIN library [Cod+11] systematically employs this approach.

2.4 Mathematical Knowledge Management

Mathematical knowledge, research, and applications are distributed globally, and mathematical knowledge is highly interlinked by explicit and implicit references and citations. Therefore, a computer-supported management system should support global interlinking and management algorithms have to scale up to large (global) data sets. Yet, virtually all current systems for formalized mathematics – the deduction systems surveyed above – operate under the implicit assumption that all knowledge is locally available and loaded into main memory – i.e. at local scale.

On the other hand, there is research and the development of representation languages for web-scale – i.e. global scope – mathematics. OpenMath [Bus+04] and MathML [Aus+10] provide general definitions of the concrete and abstract syntax of mathematical objects. Both can be customized by content dictionaries, which introduce additional primitives and describe their semantics. OMDoc [Koh06] extends these with abstract and concrete syntax for mathematical documents. In particular, these system can act as the base of a new breed of system languages for mathematical libraries. Indeed they have been used as system-level interchange formats for mathematical/symbolic systems (e.g., the use of OpenMath in the SCIEnce project [HR09]), as a basis for integrating mathematics with the semantic web (e.g., in the MONET FP6 project and the HELM/MoWGLI FP6 project), or as markup languages for web browsers (e.g., by the integration of MathML into HTML5). They are the basis of generic assistant systems such as MathWebSearch [KŞ06] for search or ActiveMath [Mel+03] for user-adaptive learning. Many mathematical assistant systems from symbolic computation or (to a much smaller extent) formal deduction can use them for export or import of their knowledge.

MKM for Deduction Systems. As a rough general rule, systems for formalized semantics fare badly on knowledge management challenges like large scale collaborative projects, change and distribution management, and integrated development environments. Retro-fitting formal deduction systems with knowledge management support has proved very expensive and unsatisfactory. In fact, because these systems have initially focused on soundness and efficiency at small scales, large scale knowledge management has proved very difficult to add as an afterthought, often prohibitively so. Moreover, developers' resources are stretched thin already by developing (and maintaining) their system at all. Therefore, the gradual migration towards new designs that overcome these problems is extremely difficult. It is no

coincidence that Matita [Asp+06] – one of the few major new systems of the last decade – is the most MKM-friendly among them.

More concretely, most proof assistants come with some support for searching statements (usually a plain text search of the source files or a search for identifiers with certain properties after loading the library) and ad hoc change management (usually based on file-modified timestamps). Most systems are able to export their library as browsing-oriented HTML, using styles, cross-references, and visibility management.

These services tend to be low-level services based on data structures that are close to the plain text source (e.g., text-based search) or high-level services based on the volatile in-memory data structures (e.g., searching for identifiers with certain types). The systems often lack persistent high-level representations of the library (e.g., in OMDoc) that could serve as the basis for generic large scale MKM services that can be developed and run independent of the deduction system. If such representations exist, such as Coq's binary or Mizar's XML files, their MKM potential is not fully exploited, often due to a lack of resources among the experts and a lack of documentation for outsiders.

One of the most advanced system-specific solutions is the automated reasoning service for HOL Light [KU13], which uses machine learning to select from a large knowledge base of theorems those that can help to prove an open proof obligation automatically. One of the most advanced system-independent MKM projects in this area is [Ala+11], which develops a general Wiki infrastructure that is applied to Mizar and Coq. For Isabelle, a partial reimplementation in a different programming language (Scala instead of ML) permitted a tighter coupling between deduction kernel and MKM applications (in this case authoring support) [Wen12].

3. THE OMDOC/MMT LANGUAGE AND SYSTEM

We have developed the OMDoc representation format [Koh06] as a web-scalable representation format for mathematical knowledge and documents that can cover formal and informal content. OMDoc extends formula representation standards like OpenMath [Bus+04] and MathML [Aus+10] with a representational infrastructure for statements, heterogeneous theory development, and documents. In particular, this makes it a suitable representation format for theorem prover libraries – and their documentation which are – when present – informal mathematics addressed to humans not machines.

3.1 MMT: A Meta-Language for Pluralistic Mathematical Libraries

In the last five years we re-developed the formal core of OMDoc in the MMT language [RK13; HKR12; Rab14b], formally defining the semantics that were left informal in OMDoc, greatly extending expressivity, and clarifying the representational primitives. OMDoc/MMT (or MMT for short) introduces a **foundation-independent** approach: It maximizes genericity and heterogeneity by avoiding any commitment to a particular foundation. This introduces a **globally scalable Module system** for Mathematical Theories that abstracts from and mediates between different foundations and maximizes the reuse of concepts, tools, and formalizations. As such, MMT provides a uniform solution to the root logic controversy of QED.

MMT separates large scale concerns, which are addressed foundation-independently by MMT, from small scale concerns, which are addressed by individual foundations and tools. Thus, foundation-specific development can focus on the logical core of the foundation instead of spending resources on ad hoc large scale support. Dually, large scale support can be developed generically (and often more easily) at the MMT level.

More concretely, MMT **integrates successful representational paradigms**

- the logics-as-theories representation from frameworks like LF,
- the structured theories from module systems in software engineering or algebraic specification languages,
- the categories of theories and the reuse along theory morphisms from the heterogeneous method,
- the Curry-Howard correspondence from type theoretical foundations,
- the URIs as globally unique logical identifiers from OPENMATH,
- the standardized XML-based interchange syntax of OMDOC.

and makes them available in a single, coherent representational system for the first time. The combination of these features is based on a small set of carefully chosen, orthogonal primitives in order to obtain a simple and extensible language design.

The central concept of MMT is that of a **theory**, which is a named list of declarations. Most importantly, the declaration of a **constant** introduces a new name possibly with additional attributes such as type, definiens, or notation. Relative to a theory, **objects** are formed as syntax trees with binding.

These three concepts are sufficient to naturally represent virtually all formal languages.

- All languages are represented uniformly as MMT theories. This includes foundations, logical frameworks, logics and type theories, signatures and theories, set theories.
- All operators and symbols of a language are represented as MMT constants. This includes the sort, constant, function, and predicate symbols of logics, the base type, type operators, and term constructors of type theories, the concepts, relations, and individuals of ontologies, as well as – via the Curry-Howard correspondence – the judgments, inference rules, axioms, and theorems of calculi.
- All composed expressions are represented as objects. This includes formulas, derivations and proofs, terms, types, kinds, and universes, etc.

All MMT theories are related via **theory morphisms**, which are used to uniformly describe representation theorems between theories. These include translations, functorial representations, implementations, and models. They are also used as the semantics of **import** declarations within theories, which uniformly represent all aspects of building theories modularly, including inheritance and instantiation.

A key innovation of MMT is the **meta-theory** relation between theories. Let us write M/T to express that we work in the object language T using the meta-language M. For example, most of mathematics is carried out in `FOL/ZFC`, i.e., first-order logic is the meta-language, in which set theory is defined. `FOL` itself might be defined in a logical framework such as LF [HHP93], and within ZFC, we can define the language of natural numbers, which yields `LF/FOL/ZFC/Nat`.

In MMT, all of these languages are represented as theories, each of which may be reused as the meta-theory of another one. Crucially, the meta-theory indicates

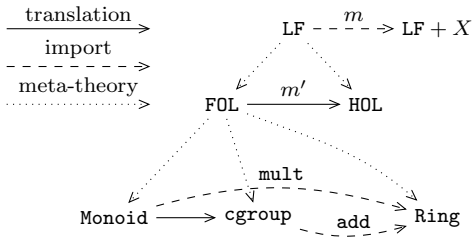


Fig. 1. Meta-Theories in Mmt

both to humans and to machines how a theory is to be understood. For example, interpretations of ZFC must understand FOL, and the typing relation of FOL is inherited from LF. The diagram of Mmt theories in Fig. 1 gives an example, where LF + X represents some extension of LF with additional features. Note that the meta-theory relation, language translations, and imports are represented uniformly as Mmt theory morphisms.

We can see Mmt as the last step in a **progression towards more abstract formalisms** as indicated below. In conventional mathematics, domain knowledge is expressed directly in ad hoc notation. Logic provided a formal syntax and semantics for this notation. Logical frameworks provided a formal way to define this syntax and semantics. Now Mmt adds a meta-level, at which we can design logical frameworks. That makes Mmt very robust against future language developments: We can develop LF + X without any change to the Mmt infrastructure and can easily migrate all results obtained within LF.

Mathematics	Logic	Logical Frameworks	Foundation-Independence
			Mmt
		logical framework	logical framework
	logic	logic	logic
domain knowledge	domain knowledge	domain knowledge	domain knowledge

Last but not least, we want to remark that Mmt supports the heterogeneous method of formalization explicitly and externally – rather than implicitly and internally like the many of the deduction systems build on the homogeneous method. This has two great advantages for our cause of building a universal community-driven QED database. The first practical: the foundation (root logic) becomes a parameter to formalization, which enables an inclusive "bring-your-own-foundation" (BYOF) approach to community building. The second is foundational: the various root logics become unencumbered by concerns of building module systems and can therefore become much weaker – as weak as the mathematical domain allows, which makes consistency of the root logic much less of an issue and thus the whole QED enterprise philosophical palatable to more communities.

3.2 Foundation-Independent Knowledge Management

We have invested heavily into building a tool suite for supporting the creation and life-cycle management of OMDoc/Mmt-encoded libraries of mathematical knowledge. Mmt comes with an API [Rab13b] that provides MKM support for

these libraries. This has led to a series of experiments demonstrating the feasibility of foundation-independent tool support.

Exploiting the small number of primitives in the MMT language, the MMT API provides a simple scalable implementation of MMT and MMT-based functionality (written in the functional and object-oriented language Scala [OSV07]). With the generality of MMT, one might think that it is not possible to implement deep, meaningful functionality at the MMT level. Indeed, implementing any result requires first generalizing it to the MMT level. However, practical experience has shown that *many results can be generalized* to the MMT level. For each result, this may require a substantial research effort, which samples existing results for specific foundations and recovers them as special cases of a general principle. But it is doubly rewarding: Besides yielding a general result, the abstract level of MMT provides a more focused view on a concept and often yields clearer intuitions.

As a consequence, all MMT-level functionality is implemented foundation-independently, and foundation-specific aspects (if any) are supplied by plugins via an abstract plugin-API. Crucially, experience shows that the *vast majority of the implementation is foundation-independent*. For example, the MMT API comprises $> 30,000$ lines of code, and the LF plugin, which provides in particular typing and proof rules for LF, comprises $< 2,000$ lines. Thus all functionality can be made available to individual formal systems at small cost.

Logical results implemented at the MMT level include notations and parsing, module system and theory transformations, type reconstruction and simplification, as well as a (so far very basic) theorem prover. For example, the foundation-independent aspects of type reconstruction include lookup of identifiers, implicit arguments, solving for meta-variables, constraint delay, and error reporting. The LF plugin only supplies ~ 10 rules of a few lines each, which correspond directly to the statement of the rules on paper – advanced aspects such as modularity and unsolved meta-variables remain transparent to the plugin developer.

Knowledge management results implemented at the MMT level include project management and build system [Hor+11], IDE (build by integrating MMT with the jEdit text editor) [Rab14a], interactive browsing using HTML+presentation MathML and JavaScript [GLR09], change management [IR12], as well as indexing, querying, and search [Rab12; KMP12; KI12].

Plugin interfaces allow the convenient import/export of content in other formats. MMT URIs serve as identifiers throughout the implementation and abstract from physical storage units such as file systems or versioned repositories. MMT maintains a catalog that maps URIs to physical locations. MMT content is loaded into memory only when needed, and the distribution of content over physical storage and networks remains transparent to MMT-based services.

For example, the MMT-based foundation-independent IDE is realized as a plugin to the jEdit text editor. Fig. 2 shows a definition of propositional logic with meta-theory LF. An intentionally introduced error was detected by type reconstruction and highlighted. Note how the sidebar shows the abstract syntax tree of the theory: The types of the variables x and y were inferred and are displayed in the syntax tree even though type reconstruction for the whole expression failed. Other features include hyperlinks, type inference tooltips for selected subexpres-

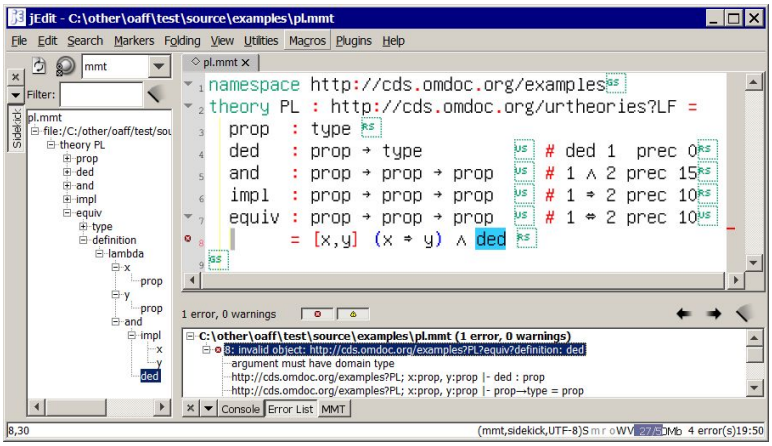


Fig. 2. MMT IDE based on jEdit

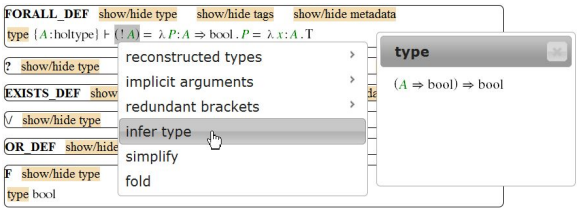


Fig. 3. Type Inference in the MMT Web Server

sions, context-sensitive auto-completion, or interactively solving for missing subexpressions based on the expected type. MMT's design makes it possible to realize this advanced functionality using only $< 1,000$ lines of glue code between jEdit and MMT.

Another example is the MMT web server. Fig. 3 shows a fragment of the HOL Light library as imported into MMT in [KR14]. It shows how the context menu is used to interactively call type inference on the selected subexpression in the definition of the universal quantifier (which is written ! in HOL Light). Other interactive features include folding subexpressions, hiding/showing inferred types, implicit arguments, and redundant brackets, or retrieving the definition of a symbol or of an included theory.

The web server can be run as a dedicated server or locally. For example, it can be run from within the jEdit IDE, in which case browser and editor become connected, e.g., for synchronous navigation.

4. THE MATHHUB.INFO ARCHIVE AND PORTAL

The MMT system described in the last section can be used to give individual users access to a mathematical library and supports their knowledge management workflows. But a full-scale, global QED database requires user/rights management, distributed revision control, and Web 3.0 features (e.g., discussions and user-generated

annotations). For that purpose, we introduce the MathHub.info system. It is realized as an instance of the Planetary system [Koh12], which we have substantially extended in the course of the work reported here.

4.1 Architecture

MathHub.info has four main components:

i) a versioned *backend* holding the libraries,
ii) MMT as the kernel tool understands the libraries provides semantic services for them,
iii) a *frontend* that makes the libraries and the semantic services available to users.
iv) a *plugin architecture* for instrumenting document presentations with localized semantic services, and

We use best-of-breed open source systems for the components going beyond MMT. For the backend, we use the GitLab repository manager [GL], an open-source alternative to GitHub, which can be adapted as a backend for MathHub.info. For the frontend, we use we use the Drupal container management system.[1] For the document instrumentation system we use our JOBAD system, a JavaScript framework that embeds semantic services into HTML documents and thus make them "active" (interactive and user-adaptive, see [**GLR:WebSvcActMathDoc09**; Koh12] for details). Figure 3 shows JOBAD in action: it links fragments of the formula presentations with computations in the MMT system and makes both available to the user embedded in the document.

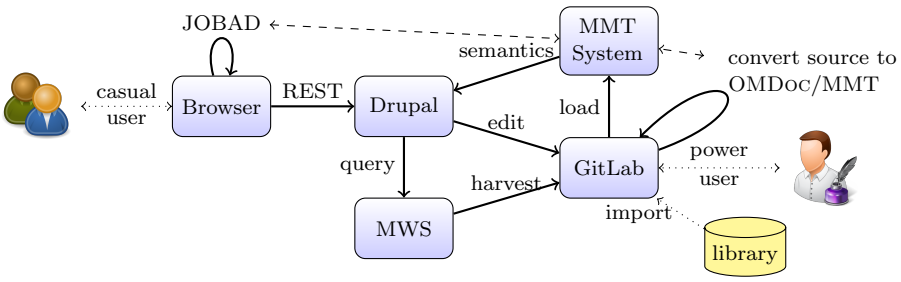


Fig. 4. The MathHub.info Architecture

Figure 4 shows the detailed architecture. Here GitLab provides distributed versioned storage of the libraries and organizes them into repositories owned by users and groups. And Drupal supplies uniform theming, discussion forums, and a plugin infrastructure for adding interface functionality. Both systems provide user management, but we automatically synchronize the users and permissions between them.

This has the advantage that we can combine two methods for accessing the contents of MathHub.info: *i*) an online, web-based workflow for the casual user, and *ii*) an offline authoring workflow based on git working copies for power users and

[1] Drupal and similar systems self-describe as "content management systems", but they actually only manage the documents and their metadata – essentially document containers – without changing their internal structure.

bulk edits. Users can fork or pull the relevant repositories from GitLab, edit them, and submit them back to MathHub.info either via a pull request to the repository masters or a direct commit/push. As the content is often highly interlinked and distributed across multiple interdependent repositories, we have developed tool support for managing multiple working copies across repository borders.

The interactive functionalities in MathHub.info are based on the OMDoc/Mmt representation of the libraries, but authors and users have to interact with them in the respective source language of the library. Both the source and OMDoc/Mmt representations are versioned in GitLab and the respective source representations must be converted into OMDoc/Mmt by language-specific custom exporters. Correspondingly library import is managed by MathHub.info at the level of the GitLab repository.

Then we can dedicate a specific git working copy together with an Mmt instance to a user or a group that shares permissions. Thus, the Mmt instance sees (and takes into account for its services) only the documents accessible to the group. If an authenticated user edits MathHub.info content, the changes are committed under his name into the specific working copy. This makes it easy to cope with multiple synchronous users, for which MathHub.info uses separate working git clones and Mmt instances.

4.2 Content

MathHub.info is intended as a portal and archive for both QED-style formalized mathematics and *flexiformal* mathematics, i.e., informal or partially formalized documents. To deal with flexiformal mathematical content, we have also extended the Mmt API to allow informal fragments and to degrade gracefully in their presence. For example, Mmt's type reconstructions works on those subexpressions for which type information is available, and Mmt's change management for those documents for which dependency information is available.

We are currently hosting a nucleus of formal and flexiformal libraries and their OMDoc/Mmt representations to develop and evaluate the functionality. Concretely, these are

i) SMGloM [**Kohlhase:dmesmgm14**], a flexiformal glossary containing about 1500 definitions and notation written in sTeX [Koh08] (semantically annotated LaTeX).
ii) about 6500 flexiformal sTeX files containing teaching materials (slides, course notes, problems, and solutions) in computer science,
iii) the LATIN logic library [Cod+11] with about 1000 modules,
iv) the TPTP library [SS98] of over 20,000 theorem proving challenge problems,
v) the Mizar Mathematical Library [TB85] of about 1,000 articles and 50,000 statements, and
vi) the HOL Light Library of about 200 files and over 15,000 statements. ()

In the future we want to open the portal up for user-supplied content, eliciting documents from mathematics and nearby disciplines. We anticipate that MathHub.info may be attractive to authors because it *i*) offers free private repositories, *ii*) allows transforming mathematical papers into hosted, searchable HTML5 documents, *iii*) allows adding interactivity by semantic annotations in a stepwise fashion.

But ultimately, we are interested in a communal resource, in which the document

sources are available for inspection, re-use and semantic analysis. Therefore, the free private repositories are in what we call *public escrow*: They are private as long as the user actively requests them to be. Otherwise, they are published under a copyleft license of the user's choice.

Even though the MathHub.info system is more general than necessary for a QED-style database of formal mathematics, it is well-suited for it. In fact, we expect that the inclusion of informal documents will support stepwise formalization workflows and allow interlinking conventional mathematical texts with their formalizations.

4.3 Global Services in MathHub.info

Like the deduction systems surveyed above, the MMT system provides its services based on data structures in main memory – essentially a must for reasoning and knowledge management services – even though the OMDOC/MMT language supports global scope and linking via its XML-based syntax and MMT URIs. Essentially, the memory consumption of the MMT system adapts dynamically to the scope of the respective user's field of attention.

The MathHub.info system can also integrate services that have a global view, i.e., services that range over all the libraries in MathHub.info. Such services usually operate on a special, reduced representation of content – e.g., the dependency graph. This representation is often produced by mapping a translation service provided by the MMT system over the library; this is very efficient because many of the MMT-based algorithms are systematically incremental, i.e., they can translate a single file or theory without loading its dependencies.

A good example of a global-scope service is the MathWebSearch service (MWS; [KŞ06]), a formula search engine. It consists of a web service that harvests formulae from all MathHub.info content and submits them to a specialized substitution tree index, which can be efficiently queried by the user (cf. Figure 4). In contrast to local services, which load content into memory on demand, MWS must keep the whole index in memory [KMP12].

In MathHub.info, we are using an extension (♭search [KI12]) of MWS, which makes use of the modular structure in OMDOC/MMT encoded libraries: ♭search uses the MMT system to flatten all theories (essentially copying over all inherited statements) creating a monolithic knowledge space, which can then be indexed in the regular MWS engine. In this way, we can search the exponentially larger knowledge space induced by the modular theories hosted in MathHub.info.

Figure 5 shows a query for the associativity of integer addition, and shows a hit in the theory IntArith, which inherits the associativity axiom from the theory AbelianGroup. This alleviates a common concern with the heterogeneous method: that the space-efficient representation given by reuse of theories in modular structures hinders accessibility of contents. By using MMT and ♭search, the heterogeneous structure of formalizations can remain transparent to casual users.

5. QED RELOADED

With the MMT representation format and system and the MathHub.info infrastructure, we have almost all the critical pieces for a modern incarnation of the QED database. In this section, we show how they can work together to form a pluralistic

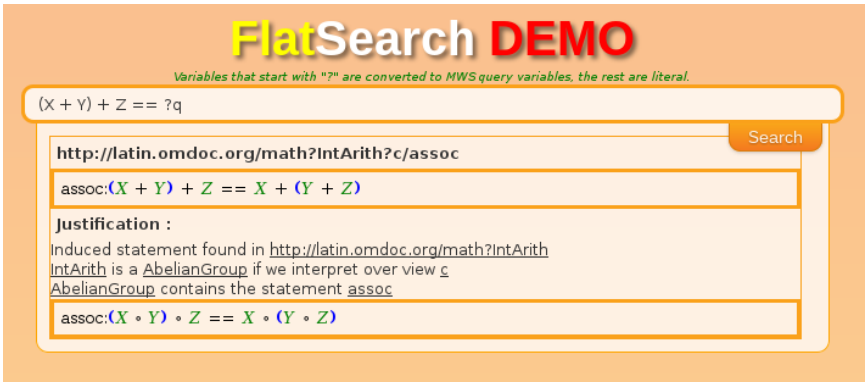


Fig. 5. ♭search on a Theory Graph similar to the one in Figure 1

formal library system of mathematical knowledge. Of course, actually filling this infrastructure with the QED content remains a community effort.

5.1 Underpinning QED with Formalized Foundations

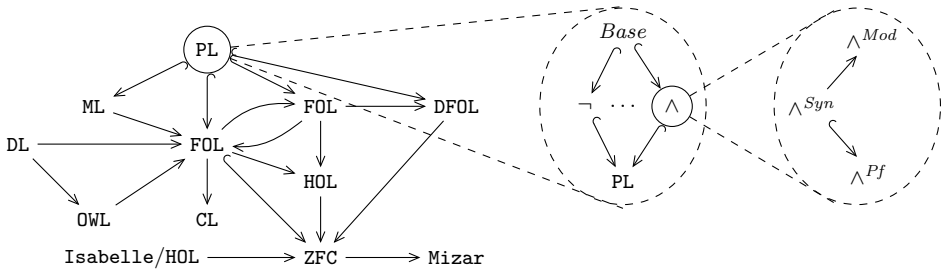


Fig. 6. A Fragment of the LATIN Atlas

Representing Foundations. The LATIN project [Cod+11] built a heterogeneous, highly integrated library of formalizations of logics and related languages as well as translations between them. It represented logics as MMT theories with the logical framework LF as the meta-theory. Fig. 6 shows a high-level view of a fragment of the resulting diagram of MMT theories. The left side shows some of the logics, the middle zooms into the modular definition of propositional logic, and the right side zooms in further to show the definition of conjunction as a triple of syntax, proof theory, and model theory.

The formalized **logics** include propositional, first-order, sorted first-order, common, higher-order, modal, description, and linear logics. Type theoretical features, which can be freely combined with logical features, include the λ-cube, product and union types, as well as base types like booleans or natural number. In many cases alternative formalizations are given (and related to each other), e.g., Curry- and Church-style typing, or Andrews and Prawitz-style higher-order logic. The logic **morphisms** include the relativization translations from modal, description, and

sorted first-order logic to unsorted first-order logic, the negative translation from classical to intuitionistic logic, and the translation from first to sorted first- and higher-order logic.

All representations systematically exploit modularity and form a single highly interconnected diagram of MMT theories. Every logical principle, e.g., as conjunction, the universal quantifier of first-order logic, or the extensionality principle of higher-order logic, is formalized in a separate module. Thus, logics can be composed modularly from the individual features using the MMT module system. For example, logic of Isabelle [Pau94] can be obtained by combining the modules for Church-style typing, simple function types, a boolean type, implication, typed universal quantification, and typed equality, as well as corresponding theories for the proof theory and corresponding theory morphisms for the model theory.

The LATIN library also allows representing model theoretical semantics as theory morphisms from a logic into a theory representing a foundation [Rab13a]. A paradigmatic example was published as [HR11]. The **foundations** in LATIN include Zermelo-Fraenkel set theory, Church's higher-order logic, and Mizar's formalized set theory [IR11]. These representations can also double as a documentation layer for the foundations.

Representing a foundation in a logical framework can be very difficult. Often it requires a deeper understanding of the logic and its implementation than published in the literature. Moreover, LF and related logical frameworks are not strong enough yet to represent all features of the typical foundations of QED systems. Already Mizar's soft type system cannot be represented elegantly. Similarly, inductive and record types as well as subtyping and implicit coercions are notoriously difficult to represent in logical frameworks.

Here a crucial strength of MMT is to allow plurality even at the logical frameworks level. Therefore, we can gradually design the better logical frameworks necessary to represent the various foundations and integrate them into the overall QED library.

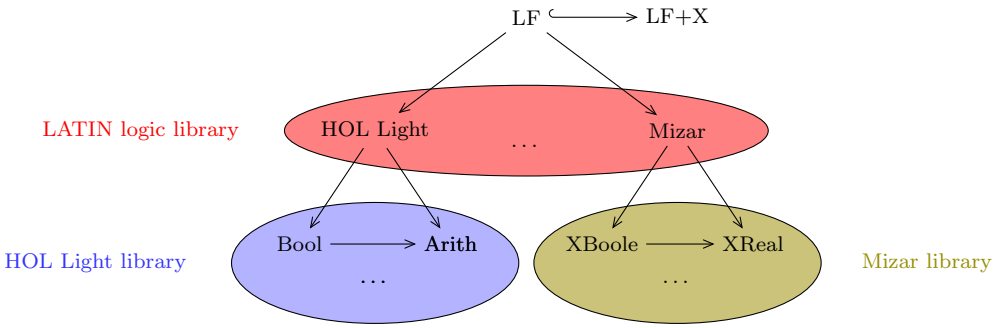


Fig. 7. Representing Libraries in MMT

Representing Libraries. After representing a foundation in an appropriate logical framework, we can convert the respective libraries into OMDOC/MMT. At this point, we have mostly completed such conversions for Mizar [Ian+13] and HOL Light [KR14], resulting in the MMT diagram of Fig. 7.

In our experience, the specification of these exports is relatively simple. Usually, the only exception are unusual idiosyncrasies language features. However, the implementation and maintenance of these exports is huge and can even be impossible without refactoring the system. Therefore, substantial collaboration from within the respective developer community is indispensable.

But if our proposal for a pluralistic QED library gains traction, it will become a maturity signal for a deduction system to support this conversion. Additionally, this would provide a forum for independent proof verification.

5.2 Integrating QED Libraries across Foundations

The pluralistic QED library described above still suffers from the library integration problem. However, we believe it is a prerequisite for attacking the problem in the first place. In fact, we might even say that we need it just to develop the tools for surveying the extent of the problem. We propose two methods for attacking the problem.

Alignments. We subscribe to the analogy that considers conventional mathematics as akin to (informal) software specification and formalized mathematics as akin to implementations. Consequently, the different foundations correspond to the different programming languages used to implement a specification.

Now the flexiformal MathHub.info system that we suggest for QED already permits representing both the formalizations and the conventional mathematics. In fact, we regard the flexiformal glossary SMGloM mentioned in Section 4.2 as the kernel of a collection of informal reference specifications: SMGloM is a theory-graph structured set of theories, which contain flexiformal definitions for standard mathematical concepts and objects [**Kohlhase:dmesmgm14**]. As such, they abstract away from many of the foundational choices we have to make in formalization. Thus, we can link each concept in a formalization to the glossary concept that it "implements".

Then we can define two symbols in two libraries to be **aligned** if they are linked to the same concept in the glossary. Alignment will have to be a very complex relation. For example, HOL Light has a type of booleans, which is aligned with two concepts in Mizar: the type first-order propositions and the 2-element set of booleans. Conversely, Mizar's number 0 is aligned with two numbers in HOL Light: the natural and the real number 0.

Therefore, alignments will not induce formal translations between libraries. But they will still allow a variety of important services, such as:

- Mathematician can use common mathematical syntax to search all QED libraries up to alignment.
- Formalizers can look for useful theorems formalized in other systems. This may even allow to automatically transport proof ideas, e.g., by using a heuristic that uses alignments to translate essential lemmas.
- Libraries of one system can be presented in the notations of another.

Interface Theories. The method of alignments can be deepened to what we call **interface logics and interface theories**. Following the above analogy, these correspond to the use of formal specifications in software engineering.

Interface logics will be the minimal logics needed to state a problem and thus will be much simpler than common foundations. This is no coincidence: Whereas foundations are designed to be simple and logically very expressive (because they should be fixed and implemented once and for all), interface logics should be as inexpressive as possible even if that makes them more complex. While giving a logic translation between any two typical foundations can be prohibitively difficult, a weak interface logic can be easily imported into all of them.

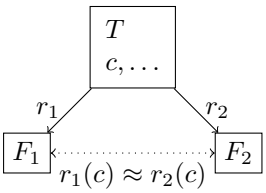
For example, axiomatic set theory, higher-order logic, and constructive type theory are common foundations, but we can specify the real numbers using a much weaker logic as an interface, e.g., a fragment of second-order logic. The LATIN logic library already constitutes a large heterogeneous library of interface logics.

Similarly, interface theories will use interface logics as their meta-theories and differ from QED-style formalizations by excluding any definitions and proofs. For example, the interface theory for the real numbers will declare derive operations such as exponentiation and their properties. But it will not commit to a concrete definition and will not prove the properties.

We propose building a community-curated library of interface theories for the typical domains of computer science and mathematics. In many cases, interface theories can be extracted from existing formalizations. Moreover, we project that mathematicians will be eager to join this effort. Indeed, in some sense, the creation of interface theories is the essence of mathematics, and we can already see this effect in our experiences with the glossary.

We can leverage interface logics and theories for the integration of QED in two ways. Firstly, deduction and symbolic computation procedures can refer to them to declare their scope. Accordingly, proof assistant can use them to describe the scope of open proof obligations. It is then straightforward to build mediator systems that match up problem producers with problem solvers.

Secondly, interfaces can provide a way to translate between libraries. Consider the diagram on the right where an interface theory T is implemented by two foundation F_1 and F_2. The correctness of these implementations is witnessed by two theory morphisms $r_1 : T \to F_i$.



In particular, for a T-symbol c, we say that $r_1(c)$ and $r_2(c)$ are **formally aligned** via (c, r_1, r_2). Typically, r_i maps symbols to symbols so that it induces a partial inverse r_i^{-1}. As suggested in [RKS11], we can then compose r_1^{-1} with r_2 to translate partially from F_1 to F_2. This partial translation will only be defined for concepts that are expressible in the interface theory and thus in particular exclude proofs.

A major difference in the analogy between software specification and formalization is that the former only has to show that an implementation is *correct*. In the latter, we also have to ask whether it is *conservative*. We define r_i to be **conservative** if for any T-formula A, the language F_i proves $r_1(A)$ only if T proves A. Trusted library translations can now be obtained as follows: If r_1 is conservative, then every F_1 proof of $r_1(A)$ guarantees the truth of $r_2(A)$ in F_2.

However, even if we assume the consistency F_1, it is still a major theoretical

challenge to prove the conservativity of r_1. This challenge has received surprisingly little attention in the QED community so far because each individual QED system implicitly assumes that its own foundation defines truth.

6. CONCLUSION & OUTLOOK

To assess the feasibility of actually starting work on the QED project, 20 years after its announcement, we have surveyed the advances in formalized libraries. We find that while there have been massive improvements in individual deduction systems and their respective libraries, they remain insular and non-interoperable, leading to duplication of work, sub-standard re-formalizations, and missed opportunities for sharing. Moreover, the current crop of deduction systems have not been designed for interoperability or global-scale libraries.

We contend that in the current situation, where we see a oligopoly of at least half-a-dozen major deduction systems and libraries[2], we will only reach critical mass and make progress towards an QED-inspired library if we take Peter Andrews' suggestion to "accept diversity" seriously and design a powerful, joint, pluralistic library system that federates the various libraries, provides mathematicians with a unified view on the library, and provides comprehensive library management functions based on that view.

We have presented a body of work conducted with the aim of building such a pluralistic, global-scale library for mathematical knowledge. This work has been conducted independently from the more established automated deduction and formal methods community under the headings of "logical frameworks" (for logical plurality) and "mathematical knowledge management" (for global scale). We view this work as missing parts needed for establishing a QED library that deserves its name.

We contend that 20 years after the QED manifesto the time is ripe for putting together the advances in deduction systems, logical frameworks, and MKM to create a QED library system and organize a community effort to work towards realizing this vision. Now, as then, the greater automated reasoning community has much to gain from such an effort. Especially, when some of its spin-offs are becoming standard tools in industry.

As a final word of caution relativizing what was said here, we point out that the QED effort will hardly be able to keep up with the $\sim 1.2 \cdot 10^5$ articles published annually in research mathematics alone. Even if we assume that most of these articles are not important enough to be formalized, we foresee that we will have to generalize the methods presented in this paper to include partial formalizations. Fortunately, our research shows that (contrary to complete mechanical proof verification) many aspects of the infrastructure and knowledge management can be generalized well to the flexiformal setting. We conjecture that this will allow small-step formalizations that will be more efficient and flexible than the current big-step formalization process.

[2]In [Wie07] Wiedijk compares surface and system languages of deduction system and finds that there is no inherent winner, and we do not expect this to change in the near future.

References

[AFP] AFP. *Archive of Formal Proofs*. URL: http://afp.sf.net (visited on 12/20/2011).

[Ala+11] J. Alama, K. Brink, L. Mamane, and J. Urban. "Large Formal Wikis: Issues and Solutions". In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, J. Urban, and F. Rabe. Springer, 2011, pp. 133–148.

[And94] Peter B. Andrews. *Accept Diversity*. 1994. URL: http://mizar.org/qed/mail-archive/volume-2/0199.html.

[Ano94] Anonymous. "The QED Manifesto". In: *Automated Deduction*. Ed. by A. Bundy. Springer, 1994, pp. 238–251.

[Art] R. Arthan. *ProofPower*. http://www.lemma-one.com/ProofPower/.

[Asp+06] A. Asperti, C. Sacerdoti Coen, E. Tassi, and S. Zacchiroli. "Crafting a Proof Assistant". In: *TYPES*. Ed. by T. Altenkirch and C. McBride. Springer, 2006, pp. 18–32.

[Aus+10] Ron Ausbrooks, Stephen Buswell, et al. *Mathematical Markup Language (MathML) Version 3.0*. W3C Recommendation. World Wide Web Consortium (W3C), 2010. URL: http://www.w3.org/TR/MathML3.

[BCH12] M. Boespflug, Q. Carbonneaux, and O. Hermant. "The λΠ-calculus modulo as a universal proof language". In: *Proceedings of PxTP2012: Proof Exchange for Theorem Proving*. Ed. by D. Pichardie and T. Weber. 2012, pp. 28–43.

[BK07] Grzegorz Bancerek and Michael Kohlhase. "Towards a Mizar Mathematical Library in OMDoc Format". In: *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*. Ed. by R. Matuszewski and A. Zalewska. Vol. 10:23. Studies in Logic, Grammar and Rhetoric. University of Białystok, 2007, pp. 265–275. URL: http://kwarc.info/kohlhase/papers/trybook.pdf.

[Bou64] N. Bourbaki. "Univers". In: *Séminaire de Géométrie Algébrique du Bois Marie - Théorie des topos et cohomologie étale des schémas*. Springer, 1964, pp. 185–217.

[Bus+04] Stephen Buswell, Olga Caprotti, et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: http://www.openmath.org/standard/om20.

[CF58] H. Curry and R. Feys. *Combinatory Logic*. Amsterdam: North-Holland, 1958.

[CH88] T. Coquand and G. Huet. "The Calculus of Constructions". In: *Information and Computation* 76.2/3 (1988), pp. 95–120.

[Chu40] A. Church. "A Formulation of the Simple Theory of Types". In: *Journal of Symbolic Logic* 5.1 (1940), pp. 56–68.

[Cod+11] M. Codescu, F. Horozal, et al. "Project Abstract: Logic Atlas and Integrator (LATIN)". In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, F. Rabe, and J. Urban. Springer, 2011, pp. 289–291.

[Con+86] R. Constable, S. Allen, et al. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, 1986.

[DW97] Ingo Dahn and Christoph Wernhard. "First Order Proof Problems Extracted from an Article in the Mizar Mathematical Library". In: *Proceedings of the International Workshop on First order Theorem Proving*. Ed. by Ulrich Furbach and Maria Paola Bonacina. RISC-Linz Report Series 97-50. Johannes Kepler Universität Linz, 1997, pp. 58–62.

[FGT92] W. Farmer, J. Guttman, and F. Thayer. "Little Theories". In: *Conference on Automated Deduction*. Ed. by D. Kapur. 1992, pp. 467–581.

[FGT93] W. Farmer, J. Guttman, and F. Thayer. "IMPS: An Interactive Mathematical Proof System". In: *Journal of Automated Reasoning* 11.2 (1993), pp. 213–248.

[GK14] T. Gauthier and C. Kaliszyk. "Matching concepts across HOL libraries". In: *Intelligent Computer Mathematics*. Ed. by S. Watt, J. Davenport, et al. Springer, 2014, pp. 267–281.

[GL] *GitLab*. URL: http://gitlab.org (visited on 02/24/2014).

[GLR09] J. Gičeva, C. Lange, and F. Rabe. "Integrating Web Services into Active Mathematical Documents". In: *Intelligent Computer Mathematics*. Ed. by J. Carette, L. Dixon, C. Sacerdoti Coen, and S. Watt. Springer, 2009, pp. 279–293.

[Gon+13] G. Gonthier, A. Asperti, et al. "A Machine-Checked Proof of the Odd Order Theorem". In: *Interactive Theorem Proving*. Ed. by S. Blazy, C. Paulin-Mohring, and D. Pichardie. 2013, pp. 163–179.

[Hal05] T. Hales. "Introduction to the Flyspeck Project". In: *Mathematics, Algorithms, Proofs*. Ed. by T. Coquand, H. Lombardi, and M. Roy. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[Har96] J. Harrison. "HOL Light: A Tutorial Introduction". In: *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design*. Springer, 1996, pp. 265–269.

[HHP93] R. Harper, F. Honsell, and G. Plotkin. "A framework for defining logics". In: *Journal of the Association for Computing Machinery* 40.1 (1993), pp. 143–184.

[HKR12] F. Horozal, M. Kohlhase, and F. Rabe. "Extending MKM Formats at the Statement Level". In: *Intelligent Computer Mathematics*. Ed. by J. Campbell, J. Carette, et al. Springer, 2012, pp. 64–79.

[Hor+11] F. Horozal, A. Iacob, et al. "Combining Source, Content, Presentation, Narration, and Relational Representation". In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, F. Rabe, and J. Urban. Springer, 2011, pp. 212–227.

[How80] W. Howard. "The formulas-as-types notion of construction". In: *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*. Academic Press, 1980, pp. 479–490.

[HR09] Peter Horn and Dan Roozemond. "OpenMath in SCIEnce: SCSCP and POPCORN". In: *MKM/Calculemus Proceedings*. Ed. by Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt. LNAI 5625. Springer Verlag, July 2009, pp. 474–479. ISBN: 978-3-642-02613-3.

[HR11] F. Horozal and F. Rabe. "Representing Model Theory in a Type-Theoretical Logical Framework". In: *Theoretical Computer Science* 412.37 (2011), pp. 4919–4945.

[Hur09] J. Hurd. "OpenTheory: Package Management for Higher Order Logic Theories". In: *Programming Languages for Mechanized Mathematics Systems*. Ed. by G. Dos Reis and L. Théry. ACM, 2009, pp. 31–37.

[Ian+13] M. Iancu, M. Kohlhase, F. Rabe, and J. Urban. "The Mizar Mathematical Library in OMDoc: Translation and Applications". In: *Journal of Automated Reasoning* 50.2 (2013), pp. 191–202.

[IR11] M. Iancu and F. Rabe. "Formalizing Foundations of Mathematics". In: *Mathematical Structures in Computer Science* 21.4 (2011), pp. 883–911.

[IR12] M. Iancu and F. Rabe. "Management of Change in Declarative Languages". In: *Intelligent Computer Mathematics*. Ed. by J. Campbell, J. Carette, et al. Springer, 2012, pp. 325–340.

[Jeu+12] Johan Jeuring, John A. Campbell, et al., eds. *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM) (Bremen, Germany, July 9–14, 2012). LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012. ISBN: 978-3-642-31373-8.

[KI12] Michael Kohlhase and Mihnea Iancu. "Searching the Space of Mathematical Knowledge". In: *DML and MIR 2012*. Ed. by Petr Sojka and Michael Kohlhase. in press. Masaryk University, Brno, 2012. ISBN: 978-80-210-5542-1. URL: http://kwarc.info/kohlhase/papers/mir12.pdf.

[KK13] C. Kaliszyk and A. Krauss. "Scalable LCF-style proof translation". In: *Interactive Theorem Proving*. Ed. by S. Blazy, C. Paulin-Mohring, and D. Pichardie. Springer, 2013, pp. 51–66.

[Kle+10] G. Klein, J. Andronick, et al. "seL4: formal verification of an operating-system kernel". In: *Communications of the ACM* 53.6 (2010), pp. 107–115.

[KMM00] M. Kaufmann, P. Manolios, and J Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, 2000.

[KMP12] Michael Kohlhase, Bogdan A. Matican, and Corneliu C. Prodescu. "MathWebSearch 0.5 – Scaling an Open Formula Search Engine". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM) (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring, John A. Campbell, et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 342–357. ISBN: 978-3-642-31373-8. URL: http://kwarc.info/kohlhase/papers/aisc12-mws.pdf.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[Koh08] M. Kohlhase. "Using LaTeX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304.

[Koh12] Michael Kohlhase. "The Planetary Project: Towards eMath3.0". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM) (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring, John A. Campbell, et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 448–452. ISBN: 978-3-642-31373-8. arXiv: 1206.5048 [cs.DL].

[KR14] C. Kaliszyk and F. Rabe. "Towards Knowledge Management for HOL Light". In: *Intelligent Computer Mathematics*. Ed. by S. Watt, J. Davenport, et al. Springer, 2014, pp. 357–372.

[KS10] A. Krauss and A. Schropp. "A Mechanized Translation from Higher-Order Logic to Set Theory". In: *Interactive Theorem Proving*. Ed. by M. Kaufmann and L. Paulson. Springer, 2010, pp. 323–338.

[KU13] C. Kaliszyk and J. Urban. "Automated Reasoning Service for HOL Light". In: *Intelligent Computer Mathematics*. to appear. Springer, 2013.

[KW10] C. Keller and B. Werner. "Importing HOL Light into Coq". In: *Interactive Theorem Proving*. Ed. by M. Kaufmann and L. Paulson. Springer, 2010, pp. 307–322.

[KŞ06] Michael Kohlhase and Ioan Şucan. "A Search Engine for Mathematical Formulae". In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*. Ed. by Tetsuo Ida, Jacques Calmet, and Dongming Wang. LNAI 4120. Springer Verlag, 2006, pp. 241–253. URL: http://kwarc.info/kohlhase/papers/aisc06.pdf.

[Mel+03] Erica Melis, Jochen Büdenbender, et al. "Knowledge Representation and Management in ActiveMath". In: *Annals of Mathematics and Artificial Intelligence* 38.1–3 (2003), pp. 47–64.

[Mil+97] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML,* Revised edition. MIT Press, 1997.

[Mil72] R. Milner. "Logic for computable functions: descriptions of a machine implementation". In: *ACM SIGPLAN Notices* 7 (1972), pp. 1–6.

[MizLib] *Mizar Mathematical Library*. URL: http://www.mizar.org/library (visited on 09/27/2012).

[ML74] P. Martin-Löf. "An Intuitionistic Theory of Types: Predicative Part". In: *Proceedings of the '73 Logic Colloquium*. North-Holland, 1974, pp. 73–118.

[Nor05] U. Norell. *The Agda WiKi*. http://wiki.portal.chalmers.se/agda. 2005.

[NPW02] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Springer, 2002.

[ORS92] S. Owre, J. Rushby, and N. Shankar. "PVS: A Prototype Verification System". In: *11th International Conference on Automated Deduction (CADE)*. Ed. by D. Kapur. Springer, 1992, pp. 748–752.

[OS06] S. Obua and S. Skalberg. "Importing HOL into Isabelle/HOL". In: *Proceedings of the 3rd International Joint Conference on Automated Reasoning*. Ed. by N. Shankar and U. Furbach. Vol. 4130. Lecture Notes in Computer Science. Springer, 2006.

[OSV07] M. Odersky, L. Spoon, and B. Venners. *Programming in Scala*. artima, 2007.

[Pau94] L. Paulson. *Isabelle: A Generic Theorem Prover*. Vol. 828. Lecture Notes in Computer Science. Springer, 1994.

[PC93] L. Paulson and M. Coen. *Zermelo-Fraenkel Set Theory*. Isabelle distribution, ZF/ZF.thy. 1993.

[Pfe01] F. Pfenning. "Logical frameworks". In: *Handbook of automated reasoning*. Ed. by J. Robinson and A. Voronkov. Elsevier, 2001, pp. 1063–1147.

[PS99] F. Pfenning and C. Schürmann. "System Description: Twelf - A Meta-Logical Framework for Deductive Systems". In: *Lecture Notes in Computer Science* 1632 (1999), pp. 202–206.

[Rab12] F. Rabe. "A Query Language for Formal Mathematical Libraries". In: *Intelligent Computer Mathematics*. Ed. by J. Campbell, J. Carette, et al. Springer, 2012, pp. 142–157.

[Rab13a] F. Rabe. "A Logical Framework Combining Model and Proof Theory". In: *Mathematical Structures in Computer Science* 23.5 (2013), pp. 945–1001.

[Rab13b] F. Rabe. "The MMT API: A Generic MKM System". In: *Intelligent Computer Mathematics*. Ed. by J. Carette, D. Aspinall, et al. Springer, 2013, pp. 339–343.

[Rab14a] F. Rabe. "A Logic-Independent IDE". In: *Workshop on User Interfaces for Theorem Provers*. Ed. by C. Benzmller and B. Woltzenlogel Paleo. 2014.

[Rab14b] F. Rabe. "How to Identify, Translate, and Combine Logics?" In: *Journal of Logic and Computation* (2014). to appear.

[RK13] F. Rabe and M. Kohlhase. "A Scalable Module System". In: *Information and Computation* 230.1 (2013), pp. 1–54.

[RKS11] F. Rabe, M. Kohlhase, and C. Sacerdoti Coen. "A Foundational View on Integration Problems". In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, F. Rabe, and J. Urban. Springer, 2011, pp. 107–122.

[Sko67] Thoralf Skolem. "The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent vari- ables ranging over infinite domains". In: 3rd printing, 1997. Source books in the history of the sciences series. Cambridge, MA: Harvard Univ. Press, 1967, pp. 302–333. ISBN: 0-674-32450-1.

[SS98] G. Sutcliffe and C. Suttner. "The TPTP Problem Library: CNF Release v1.2.1". In: *Journal of Automated Reasoning* 21.2 (1998), pp. 177–203.

[SW83] D. Sannella and M. Wirsing. "A Kernel Language for Algebraic Specification and Implementation". In: *Fundamentals of Computation Theory*. Ed. by M. Karpinski. Springer, 1983, pp. 413–427.

[TB85] A. Trybulec and H. Blair. "Computer Assisted Reasoning with MIZAR". In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence*. Ed. by A. Joshi. Morgan Kaufmann, 1985, pp. 26–28.

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

[Urb06] Josef Urban. "XML-izing Mizar: making semantic processing and presentation of MML easy". In: *Mathematical Knowledge Management, MKM'05*. Ed. by Michael Kohlhase. LNAI 3863. Springer Verlag, 2006, pp. 346 –360.

[Wen12] M. Wenzel. "Isabelle/jEdit - A Prover IDE within the PIDE Framework". In: *Intelligent Computer Mathematics*. Ed. by Johan Jeuring, John A. Campbell, et al. Springer, 2012, pp. 468–471.

[Wie07] F. Wiedijk. "The QED Manifesto Revisited". In: *From Insight to Proof, Festschrift in Honour of Andrzej Trybulec*. 2007, pp. 121–133.

[WR13] A. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1913.

[Coq14] Coq Development Team. *The Coq Proof Assistant: Reference Manual*. Tech. rep. INRIA, 2014.

[The] The HOL4 development team. http://hol.sourceforge.net/.