Neuantrag auf Sachbeihilfe

OAF: An Open Archive of Formalizations

Acronym: OAF

July 21, 2015

Michael Kohlhase
Jacobs University

Florian Rabe
Jacobs University

Contents

0 General Information (for the ELAN system only)

0.1 Joint Proposal; Applicants (Antragsteller)

	Prof. Dr. Michael Kohlhase Professor of Computer Science 13. September 1964	Dr. Florian Rabe ???? ????
	Work Address (Dienstanschrift):	
	Jacobs University Campus Ring 1, 28757 Bremen Tel: +49 421 200 3140 Fax: +49 421 200 493140 m.kohlhase@jacobs-university.de	Jacobs University Campus Ring 1, 28757 Bremen Tel: +49 421 200 31?? Fax: +49 421 200 4931?? m.kohlhase@jacobs-university.de
	Private Address (Privatanschrift):	
	Mühlental 5, 28717 Bremen Tel: +49 421 6396849	None of your business Tel: that neither

0.2 Topic (Thema)

Ein offenes Archivsystem für Formalisierungen

0.3 Research area and field of work (Fachgebiet und Arbeitsrichtung)

Scientific discipline: Computer Science
Fields of work: Knowledge Management

0.4 Anticipated total duration (Voraussichtliche Gesamtdauer)

3 years; initial proposal

0.5 Application period (Antragszeitraum)

36 Months starting 1. September 2013

0.6 Summary

Mathematical knowledge is at the core of science and engineering and a major factor in innovation in developed societies. Its quantity is currently growing faster than our ability to organize and utilize it. Machine support via symbolic (software) systems would greatly enhance the potential of mathematical knowledge, but is predicated on the existence of libraries of formalized background knowledge. Thus, machine support is hampered by the costliness of formalization. Even worse, symbolic systems and their libraries are non-interoperable because they are based on differing foundations, and much work is spent on re-development of basic libraries that could be more productively invested in covering new areas. Moreover, the ensuing plurality of library formats forces implementors to spend time on library organization features instead of perfecting the core functionality of their systems.

The proposed OAF project tackles these interoperability and plurality problems by developing an open archive for formalizations, a common and open infrastructure for managing and sharing formalized mathematical knowledge such as theories, definitions, and proofs. The OAF infrastructure is designed to be scalable with respect to both the size of the knowledge base and the diversity of logical foundations. In particular, the OAF system will be based on a uniform foundation-independent representation format for libraries, which allows formalizing the logical foundations alongside the library and thus acts as framework for aligning libraries.

This will resolve two major bottlenecks in the current state of the art. It will provide a permanent archiving solution that not all systems and user communities can afford to maintain separately. And it will establish a standardized and open library format that serves as a catalyst for comparison and thus evolution of systems.

Symbolic system developers will be able to delegate library management by exporting their libraries into the OAF and developers of mathematical knowledge management (MKM) systems will be able to develop high-level services on top of it. Contrary to the current state of the art, this permits separating the concerns: developers of symbolic systems could focus on the logical core of their system and developers of generic MKM services gain access to relevant-size libraries.

Finally, our archive's uniform representation language for libraries enables – for the first time – systematic large scale investigations into the integration of large libraries written in different formalisms. In the long run, this enables the seamless combining and merging of libraries into a universal large-scale knowledge space.

0.7 Zusammenfassung

Mathematisches Wissen liegt im Kernbereich von Wissenschaft und Technologie und ist ein wesentlicher Innovationsfaktor. Allerdings wächst das verfügbare Wissen schneller als unsere Fähigkeit es zu organisieren und zu nutzen. Maschinelle Unterstützung durch symbolische (Software)-Systeme könnte theoretisch Abhilfe schaffen, aber benötigt in der Regel Bibliotheken formalisierten mathematischen Hintergrundwissens. Somit wird die maschinelle Unterstützung durch die hohen Formalisierungskosten ausgebremst. Schlimmer noch: Die Systeme und ihre Bibliotheken sind nicht interoperabel, da sie auf unterschiedlichen logischen Grundlagen aufbauen. So gehen Ressourcen durch Parallelentwicklung grundständiger Bibliotheken verloren, die dann wiederum für die Abdeckung neuer Wissensgebiete fehlen. Zudem zwingt die Pluralität der Bibliotheksformate zur Parallelentwicklung von Bibliotheksmanagementfunktionalitäten statt die originären Systemfunktionalitäten zu verbessern.

Das beantragte OAF-Projekt trägt zur Lösung der Interoperabilitäts- und Pluralitätsprobleme durch die Entwicklung eines offenen Archivs für Formalisierungen bei, also einer gemeinschaftlichen Infrastruktur zur Nutzung und zum Management formalen mathematischen Wissens wie Theorien, Definitionen, oder Beweisen. Diese Infrastruktur ist auf Skalierbarkeit sowohl über große Wissensmengen als auch über unterschiedliche logische Grundlagen ausgelegt. Insbesondere basiert es auf einem einheitlichen metalogischen Repräsentationsformat, in dem die logischen Grundlagen zusammen mit den Bibliotheken formalisiert werden, so dass es als Fundament für die semantische Vernetzung von Bibliotheken dienen kann.

Dadurch wird das OAF Projekt zwei wichtige Engpässe beseitigen. Zum einen stellt es eine allgemeine Archivlösung bereit, die sich nicht alle einzelnen Arbeits- oder Nutzergruppen alleine leisten können. Zum anderen liefert es ein standardisiertes, systemübergreifendes Bibliotheksformat, das die Vergleichbarkeit und dadurch die Evolution von Systemen katalysiert.

Entwickler symbolischer Systeme können das Bibliotheksmanagement an das OAF-System delegieren, und die Mathematical Knowledge Management (MKM) Community kann auf der Basis von OAF Mehrwertdienste entwickeln. Im Gegensatz zum momentanen Stand der Technik können sich die Entwickler symbolischer Systeme auf die logischen Grundlagen ihrer Systeme konzentrieren und die Entwickler generischer MKM-Dienste erhalten Zugang zu Bibliotheken relevanter Größe.

Unsere systemübergreifende Repräsentationssprache ermöglicht – zum ersten Mal – die systematische Erforschung der Integration von großen Bibliotheken verschiedener Formalismen. Längerfristig ermöglicht dies die nahtlose Verknüpfung und Verschmelzung von Bibliotheken zu einem großen formalen Wissensraum.

1 State of the Art and Preliminary Work

Deduction Systems

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

All deduction systems in this area are based on a fixed foundation, i.e., a fixed logic in which all formalizations in that system are stated. Setting aside philosophical differences (e.g., constructive logic vs. classical set theory; different choice axioms), these foundations differ mainly in details that are irrelevant for high-level formalization goals.

Most current systems derive either from constructive type theories or higher-order logic. The former are 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 mathematical proofs as data. Systems include Nuprl [Con+86], Agda [Nor05], Coq [Tea], and Matita [Asp+06]. The second group is based on Church's higher-order logic [Chu40] using the LCF architecture [Mil72]. These systems include HOL4 [HOL], ProofPower [Art], Isabelle/HOL [NPW02], and HOL Light [Har96]. The foundation 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. Notably, only Mizar [TB85] and Isabelle/ZF [PC93] are based on variants of axiomatic set theory and thus most similar to the set theoretical language common to mathematics.

Regarding mathematical knowledge in particular, Wiedijk identifies HOL Light, Coq, ProofPower, Mizar, and Isabelle/HOL as the most advanced systems using a sample of 100 representative mathematical theorems [Wie07]. If we consider formalizations of computer systems rather than mathematical knowledge, the field is similar, but we should also rank PVS among the most advanced systems.

Almost all formalizations in these systems are based on the *homogeneous* method, which fixes one foundation with all primitive notions (e.g., types, axioms, and rules) and uses only conservative extensions (e.g., definitions, theorems) to model domain knowledge. 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.

On the other hand, the *heterogeneous* method, going back to the works by Bourbaki [Bou64], focuses on defining theories that may introduce new primitive notions, and considers truth relative to a theory. The heterogeneous method optimizes reusability by stating every result in the weakest possible theory and using *theory morphisms* to move results between theories in a truth-preserving way. This is often called the *little theories* approach [FGT92].

Many systems support heterogeneous reasoning (e.g., locales in Isabelle, parametric theories in PVS, records in Coq). Typically, it is employed ad hoc as a high-level construct that is defined in terms of the low-level constructs of the foundation. Only IMPS uses the heterogeneous method systematically as a primitive. This dominance of the homogeneous method stands in contrast to the success of the heterogeneous method in model theory and algebraic specification (e.g., [GB92; CoF04]) where module systems are used to build large theories out of little ones [SW83]. Similarly, scientific practice prefers the heterogeneous method. For example, while all mathematics can be reduced to first principles (e.g., using the homogeneous method based on axiomatic set theory), it is usually carried out in highly abstracted settings that hide the foundation. For example, the category of categories is used routinely without focusing on its foundational subtleties.

The combination of fixed foundation and homogeneous method means that 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. However, the resulting formalizations are actually less valuable: It becomes virtually impossible to move them between foundations. Therefore, almost all current systems are mutually incompatible, with only a few ad hoc translations between them (e.g., [KW10; KS10]).

Formal Libraries

Overview All deduction systems provide 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 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 [Hal14] and Georges Gonthier’s work in Coq for the recently proved Feit-Thompson theorem [1]. John Harrison’s formalizations in his HOL Light system [2] and the NASA PVS library [3] 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 ad hoc heterogeneity. Gonthier’s Coq work in the Mathematical Components project [4] does so most systematically.

In principle, highly integrated libraries can be developed best with the heterogeneous method. This was pursued in the IMPS library [5] and (by the proposers) in the LATIN atlas [Cod+11; KMR09].

Another heterogeneous library is the TPTP library [SSY94] It is special in that it focuses on formalizations without proofs to be used as challenge problems for automated theorem provers. As such, it is mainly restricted to variants of first-order logic, but several extensions have been proposed recently.

The OpenTheory project [Hur09] has somewhat similar objectives to OAF but is limited to 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 two 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 *connect or merge two libraries* from different systems. Usually, this requires translating the libraries into a common language and then identifying and eliminating overlap between the two libraries.

No strong tool support is available for either of the two 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 [Hal14], 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 foundation!

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 languages optimized for fast and convenient writing by human users. Consequently, highly non-trivial algorithms for parsing, type reconstruction, and theorem proving have been developed to build the corresponding abstract data structures. This has the effect that for each library, there is essentially only a single system able to read it.

Moreover, these 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. Thus, they often do not provide good support for exporting libraries is easier-to-read formats. And even where they do – for example, Mizar, HOL Light, and Coq provide at least idiosyncratic, system-near exports – the exports have two problems.

Firstly, the use of the homogeneous method means that the export contains the elaborated low-level data structures and not the high-level structure that would be more valuable for reuse. For example, in the case of Mizar, it proved notoriously difficult [DW97; BK07; Urb03; Urb06] until the proposers obtained an export [**IanKohRabUrb:tmmliotaa11:base**] that they could actually make use of in their applications.

Secondly, the export code quickly becomes 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 even though the XML data structures were tightly integrated with (and thus essential for) the main system.

[1] http://www.msr-inria.inria.fr/events-news/feit-thompson-proved-in-coq
[2] http://www.cl.cam.ac.uk/~jrh13/hol-light/
[3] http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/
[4] http://www.msr-inria.inria.fr/Projects/math-components
[5] http://imps.mcmaster.ca/theories/theory-library.html

Library Imports If an export is available, importing a library requires one out 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 successful sequences of export followed by import between two large deduction systems.

A small number of imports have been realized using ad-hoc logic translations, typically in special situations. [KW10] translates from HOL Light [Har96] to Coq [Coq14] and [OS06] from 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 aided by the use of the same logical framework. The Coq library has been imported into Matita, aided by the fact that both use very similar foundations. The OpenTheory format [Hur09] was developed to facilitate 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 [**IanKohRabUrb:tmmliotaa11:base**] for Mizar using the LATIN framework based on LF [HHP93], and making the library available to knowledge management services. Another example is the recent Dedukti system [BCH12], which imports Coq and HOL Light into a similar 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 would be interesting to reuse in a different library are usually defined as conservative extensions. Their properties are proved theorems in the single fixed foundation rather than axioms in a dedicated theory. Consequently, exports do not preserve the high-level structure implicit in the formalization. This is disastrous from an integration perspective because different systems inevitably use different definitions.

For example, basic definitions like lists or the real numbers usually exist in all systems but using different definitions. Being defined concepts, their translation is induced by the logic translation. Thus, a logic translations 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.

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. 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 [KRSC11].

Knowledge Management

Overview Mathematical Knowledge Management (MKM) is a relatively young field (~ 10 years) combining formal mathematics and software engineering to handle the large amount of mathematical knowledge. A major line of research is the development of representation languages that can act in particular as standardized library formats. OpenMath [Bus+04] and MathML [**W3C:MathML3:biblatex**] 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 [Koh06b] extends these with abstract and concrete syntax for mathematical documents.

These languages have been used as interchange formats for assistant 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 [**ActivemathAima03**] 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. 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 large scale MKM services that can be developed and run independent of the deduction system. If such representations exist, such as Coq's compiled 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].

These services operate either on the plain text input files or on files generated by the proof assistant, and both are highly specific to the respective system. An integration of generic MKM systems with deduction systems is often difficult because the necessary library exports into standardized formats are missing.

Preliminary Work

The KWARC group has contributed considerably to the state of the art of representation languages for deduction systems, formal libraries, and knowledge management; this has been covered above. We will briefly summarize and highlight our contributions, which concentrate on three areas.

Logical Frameworks Theoretically, the main prerequisite has been established in the LATIN project [KMR09]. The LATIN logical framework [Rab13; Cod+12] integrates institutional representations of model theory and type theoretical representations of proof theory and thus permits combining the benefits of both worlds.

A key element is the explicit representation of the foundation underlying a deduction system as a theory of the framework [IR11], which we call *foundations-as-theories*. Inspired by work on categorical logic, this permits representing *models-as-morphisms* from the logic to the foundation. A paradigmatic example was published as [HR11].

The LATIN atlas [Cod+11] employs this framework to build a heterogeneous, highly integrated library of formalized logics. It includes formalizations of the syntax, model theory, and proof theory of logics (e.g., propositional, first-order, higher-order and their variants), type theories (e.g., λ-cube, Martin-Löf type theory, Curry and Church encodings and their variants), and foundations (various set theories and type theories based on the former). It also includes several translations between these languages formalized as morphisms in the LATIN category.

Representation Languages We have developed the OMDoc representation format [Koh06b]. It extends formula representation standards like OpenMath [Bus+04] and MathML [**W3C:MathML3:biblatex**] (to which we have heavily contributed [KR12; HKR11; DK09b; DK09a]) with a representational infrastructure for statements, heterogeneous theory development, and documents.

In the last five years we re-developed the formal core in the MMT language [RK13b; HKR12; KRSC11], fixing semantical aspects that were left implicit in OMDoc, greatly extending expressivity, and clarifying the representational primitives. MMT introduces a foundation-independent approach, i.e., it is designed in a way that maximizes genericity (and thus reuse) avoiding any commitment to a particular foundation.

MMT is the intended basis for representing knowledge in the OAF project.

Library Management Systems We have invested heavily into building a tool stack for supporting the creation and life-cycle management of OMDoc/MMT-encoded corpora of mathematical knowledge documents in libraries. The OMDoc and MMT languages come with implementations that provide MKM support for their libraries.

This has led to a series of experiments demonstrating the feasibility of foundation-independent MKM services including interactive browsing [GLR09], databases [ZK09; KRZ10], project management [Hor+11], change management [IR12b], and search/querying [**IanKohRabUrb:tmmliotaa11:base**; Rab12; KI12; KMP12; KŞ06].

Applications The technology stack summarized above has been used in various projects and case studies, and here we will summarize the ones that are directly relevant to the OAF project.

The Planetary framework provides a Drupal-based user- and content management system for formal and informal mathematical libraries. It has been instantiated to the PlanetMath.org [PM] encyclopedia, the PantaRhei [PR] course support system, and – experimentally – a LATIN Atlas Portal.

The jEdit-MMT system provides a plugin for the text editor jEdit that adds MMT support [IR12a]. It utilizes MMT's design to provide a foundation-independent editor for formalizations. It offers semantic navigation, context-sensitive auto-completion, and interactive display of inferred information and errors.

The Semantic Alliance framework [Dav+12] allows to annotate semantic objects in open-API applications with concepts of an OMDoc-encoded domain description and uses the theory-graph information and content for a semantic help and navigation system; semantic change management and verification services. This framework has been instantiated in *i*) the SiSsI system [KK09b; KK09a; KK13] spreadsheets (the semantic objects are spreadsheet cells and functional blocks) in cooperation with DFKI Bremen. *ii*) the FormalCAD system [Koh13; Koh+09] (the semantic objects are CAD assemblies and components) in cooperation with FAU Erlangen.

1.1 List of Project-Related Publications

1.1.1 Peer-Reviewed Articles

[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.

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

[IR12b] M. Iancu and F. Rabe. "Management of Change in Declarative Languages." In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 325–340. ISBN: 978-3-642-31373-8.

[KRZ10] M. Kohlhase, F. Rabe, and V. Zholudev. "Towards MKM in the Large: Modular Representation and Scalable Software Architecture." In: *Intelligent Computer Mathematics*. Ed. by S. Autexier, J. Calmet, D. Delahaye, P. D. F. Ion, L. Rideau, R. Rioboo, and A. P. Sexton. LNAI 6167. Springer Verlag, 2010, pp. 370–384. ISBN: 3642141277. arXiv: `1005.5232v2 [cs.OH]`.

[KŞ06] M. Kohlhase and I. Şucan. "A Search Engine for Mathematical Formulae." In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*. Ed. by T. Ida, J. Calmet, and D. Wang. LNAI 4120. Springer Verlag, 2006, pp. 241–253. URL: `http://kwarc.info/kohlhase/papers/aisc06.pdf`.

[Rab12] F. Rabe. "A Query Language for Formal Mathematical Libraries." In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 142–157. ISBN: 978-3-642-31373-8. arXiv: `1204.4685 [cs.LO]`.

[Rab13] F. Rabe. "A Logical Framework Combining Model and Proof Theory." In: *Mathematical Structures in Computer Science* 23.5 (2013), pp. 945–1001.

[RK13a] F. Rabe and M. Kohlhase. "A Scalable Module System." In: *Information and Computation* 230.1 (2013), pp. 1–54.

1.1.2 Other Publications

None.

1.1.3 Patents

None.

2 Objectives and Work Schedule

2.1 Anticipated total duration of the project

The total project duration is anticipated to be six years; *the current proposal requests DFG funds for the initial project phase of three years*, which will develop a first functional and useful prototype of the OAF system and seed it with material converted from five paradigmatic systems.

In the subsequent, second project phase we will expand on the results – supporting more proof assistants and more services – and deepen the integration of libraries with each other and of proof assistants with the OAF.

2.2 Objectives

O1: Library Archive We want to realize a universal archiving solution for formal mathematical libraries. This archive must satisfy two conflicting goals: On the one hand, it must be so generic that it is open to all logics and implementations; on the other hand, it must be aware of the semantics of the formalized content so that it can offer meaningful services. These services must be independent of both the formal system and the implementation used to produce the library and offer a uniform high-level interface for both users and machines to access the combined library.

This will resolve two major bottlenecks in the current state of the art. It will provide a permanent archiving solution that not all systems and user communities can afford to maintain separately. And it will establish a standardized and open library format that serves as a catalyst for comparison and thus evolution of systems.

Concretely, we see three ways the formal methods and mathematical knowledge management communities can benefit from the OAF: *i)* users can view formerly disparate developments in a common, neutral framework and compare them, *ii)* system developers can import libraries from other logical systems to extend the reach of formalizations and avoid duplicate development *iii)* the existence of a library management system (and importable content) can lower the entry hurdle for developing new logic-based systems.

O2: Library Management Based on our archive, we can provide uniform library management support. Here our past experiments have shown that the foundation-independence of our representation language and its implementation can provide a clean and scalable interface between the formal libraries and the MKM services.

Thus, system developers will be able to focus on exporting their libraries in our format, and MKM system developers will be able to develop high-level services on top of it. Contrary to the current state of the art, this permits separating the concerns: Currently, developers of deduction systems would love to focus on the logical core of their system but find themselves forced to invest into ad hoc MKM support to build large libraries scalably; and vice versa, developers of MKM systems would love to focus on the abstract MKM level, but find themselves forced to invest into brittle connections to deduction systems to gain access to relevant-size libraries.

O3: Library Integration Our archive's uniform representation language for libraries enables – for the first time – systematic large scale investigations into the integration of libraries written in different formalisms. In the long run, this will permit the seamless combining and merging of libraries.

However, the full library integration problem as described in Section 1 remains out of reach for current research. Instead, we will focus on establishing fundamental prerequisites that prepare for future approaches to the problem. These include representational primitives that support integration, tool support for refactoring homogeneous libraries into heterogeneous ones that enable reuse across libraries, and heuristics for finding possible overlap between libraries.

2.3 Work Programme Including Proposed Research Methods

The project is organized around 18 work packages, which we summarize in and explain in the following.

WA/P	Title	MK RM	MK RAM	FR RM	FR RAM	total RM	total RAM
WA1	**Open Archive Infrastructure**	**8**	**16**	**4**	**6**	**12**	**22**
WP1.1	Back End	3	6	3	3	**6**	**9**
WP1.2	Front End	4	6			**4**	**6**
WP1.3	Import/Export Workflows	1	4	1	3	**2**	**7**
WA2	**Seeding the Archive with Libraries**	**8**	**0**	**13**	**9**	**21**	**9**
WP2.1	Mizar			2		**2**	**0**
WP2.2	HOL Light			2	3	**2**	**3**
WP2.3	IMPS	2				**2**	**0**
WP2.4	Matita			6		**6**	**0**
WP2.5	PVS	6				**6**	**0**
WP2.6	Standard Interface Library			3	6	**3**	**6**
WA3	**Library Integration**	**10**	**8**	**8**	**12**	**18**	**20**
WP3.1	Interface Logics			4		**4**	**0**
WP3.2	Pragmatic Interfaces			2	6	**2**	**6**
WP3.3	Interface Theory Generation			2	6	**2**	**6**
WP3.4	Theory Refactoring	5	6			**5**	**6**
WP3.5	View Finder	5	2			**5**	**2**
WA4	**Global Services**	**10**	**12**	**11**	**9**	**21**	**21**
WP4.1	Search & Querying	6	6			**6**	**6**
WP4.2	Versioning and Management of Change			5	6	**5**	**6**
WP4.3	Generic Type/Proof Inference			6	3	**6**	**3**
WP4.4	Active Libraries	4	6			**4**	**6**
	totals	**36**	**36**	**36**	**36**	**72**	**72**

R(A)M $\widehat{=}$ Researcher (Assistant) Months; WP lead efforts light gray italicised

Table 1: Work Areas and Work Packages

Work Area 1: Open Archive Infrastructure

In this work area, we design and develop the open archive for formalized mathematical knowledge envisioned in Objective O1. The two major starting points are the LATIN logic graph, which formalizes the logics, and the MMT language, which permits the uniform representation language for libraries. Each logic of the logic graph serves as the root of an MMT theory graph representing that logic's library.

The main deliverable of this work area is a first functional and useful prototype of the OAF system. This will be based on the technologies that we have developed within and in the vicinity of the LATIN project. In particular, this includes versioned databases [ZK09], web front ends [Koh+11; IR12a] for user interaction, and our math archives [Hor+11] as the library organization format. The main challenge is a coherent integration of these technologies into a production-ready system that can be deployed to the community.

The work in this work area will be carried out early on in the project to provide supporting infrastructure for the other work areas. Due to the existing version of a Mizar import, it can be evaluated directly without depending on Work area WA2.

Work Package 1.1	Site	MK	FR	**all**
Back End	Effort (RM+RAM)	3+6	3+3	**6+9**

In this work package, we will design and implement the basic infrastructure of the archive system back end. Even though our technology is well-prepared for large scale application [KRZ10], we foresee the need for further investments into the scalability and automation of the associated work flows due to the size of the involved libraries.

- **T1 MMT kernel** We will use and adapt the MMT API to act as the interface layer through which the libraries are accessed.
- **T2 Database back end** Libraries will be stored in a versioned database that permits SVN-style checkout, e.g., our TNTBase system [ZK09]. We expect the main task here to be developing OAF-specific indexes (complementing and extending the XML Database indices TNTBase already supplies) and making them available to the MMT kernel efficiently.
- **T3 Memory management** We expect that with the much larger libraries encountered in the OAF project, we will no longer be able to keep the whole library in memory. Therefore we will need to develop memory management techniques that allow to dynamically load (and unload!) parts of the repository as they are needed for computation.

Work Package 1.2	Site	MK	FR	**all**
Front End	Effort (RM+RAM)	4+6	+	**4+6**

A web front end will offer users a uniform browsing interface to the OAF. We will employ the Drupal content management system for the front end, building on our experiences from the Planetary framework. Besides logic-oriented services described in Work package WP4.4, this front end will provide several community-oriented features for discussions between users.

- **T1 OAF content integration** We have to integrate the MMT kernel with Drupal's flexible content models that cache, style, and adapt presentations generated by MMT. This amounts to designing and implementing a controller component that mediates between the MMT model and the Drupal view components.
- **T2 Commenting/rating/refereeing** We will use Drupal's customizable content aggregators that allow building discussion forums, rating systems, etc. We have already experimented with localized commenting systems, and we will extend them to a full-blown refereeing system, where referee comments to library submissions can be localized to particular fragments of formalizations.
- **T3 Dissemination channels** Building on the data from **T1.2.2**, we will experiment with establishing dissemination channels for formal content. These can act both as a means to get an overview over available content as well as a means to attribute academic credit to formalization work.[6] Concretely, we will aggregate refereed "journals" of featured formalizations (inspired by the Journal of Formalized Mathematics of the Mizar community), and a slashdot-like self-regulating, karma-based news-feed about OAF content.

Work Package 1.3	Site	MK	FR	**all**
Import/Export Workflows	Effort (RM+RAM)	1+4	1+3	**2+7**

While the logic graph is relatively stable, libraries will be frequently imported (whenever the maintainer releases a new version) and exported (whenever a user checks out the library). Therefore, we will develop scalable push-button workflows for importing and exporting individual libraries.

- **T1 Export** The export work flow is relatively simple because it is essentially a checkout from the OAF repository developed in WP1.1. Users must be able to chose between exporting the original source (which can be used in the respective deduction system) and the OAF representation format (which can be used to run MKM services locally). Therefore, OAF must maintain both versions with fine-granular (i.e., referencing line-column positions in files) cross-references between them. Moreover, OAF must provide a dependency management component that permits exporting just the dependency closure of a fragment of a library.
- **T2 Import** The import workflow consists of 2 steps. Firstly, a deduction system must export its library in our uniform representation format or any format that exposes the same information. This must include the cross-references and the dependency relation needed for the export. This is the most difficult step – it is system-specific and handled in Work area WA2.
  Secondly, we must import this representation into the OAF repository and generate the HTML-oriented representation to be cached by Drupal. This step will be based on MMT's build processes [Hor+11] coupled with

[6] It has been often lamented that there is no way to reward the non-trivial and important work of formalization in our current academic system.

an appropriate Drupal module. Based on our experience with Mizar, we know how to approach the scalability issues here. In particular, the import will be *local*: The import of a file F must only depend on F and may not retrieve any of the (possibly many) other files that F logically depends on. Local imports makes it easy to split library imports into multiple batches, resume failed imports, and to restrict future imports to the changed files.

Work Area 2: Seeding the Archive with Libraries

In this work area, we import a set of selected formal libraries of deduction systems into the OAF. We will focus on large-scale logics, i.e., logics that are backed by industry-strength implementations with *i)* advanced (semi-)automated proof support, *ii)* a stable developer and user community, and *iii)* a large library of formalizations. For each system, we need to convert between the respective native and our uniform representation. As described in Section 1 and Task T2 of Work package WP1.3, a major difficulty here is that each import depends on an export realized as a part of the respective deduction system.

We have already developed such an import for Mizar, where the key to success was an intense collaboration with Josef Urban, who has developed the corresponding export as a part of Mizar. Similarly, all exports must be realized in collaboration with experts from the respective developer community. Therefore, for each system, we will collaborate extensively with at least one expert as listed below. All these collaborations were established in the past and can be directly utilized for OAF (see also Section **??**).

In addition to Mizar, we will prioritize imports from the HOL/Light, Matita, PVS, and IMPS libraries, which, together, cover the major representational paradigms in foundations of deduction systems. Once the OAF project has started, we plan to apply for an EU project to build OAF import facilities for the Isabelle (Munich/Paris), ProofPower (London), Coq (Paris), and NuPRL (Potsdam) systems.

The work packages in this work area are independent of each other and are spread out over the project duration. The relatively easier ones are placed at the beginning to collect experience for Work package WP1.3, which then in turn provides support for the remaining work packages.

Work Package 2.1	Site	MK	FR	**all**
Mizar	Effort (RM+RAM)	+	2+	**2+0**

We choose Mizar as the biggest library of formalized mathematics. Moreover, Mizar is special in that it is the only large library written in a language close to typical mathematics.

A partial version of this import already exists based on Josef Urban's XML export. The main missing part is the import of Mizar proofs, where we do not expect substantial problems. However, recent work on Mizar has caused the XML export to be out of sync so that some upgrading will be necessary.

Being available early on, we will be able to use Mizar as the main case study to evaluate the OAF infrastructure developed in Work area WA1 and the services from Work area WA4. This includes, in particular, the work flow that automatically synchronizes the official Mizar library with OAF upon every library release. Pending good progress in OAF, we want to offer hosting the official Mizar library either as part of the OAF or using a special installation of OAF on the Mizar servers.
Collaborating expert: Adam Naumowicz, chair of the Mizar library committee
Collaborating expert: Josef Urban, the main developer of the XML export

T1 **Mizar Proofs** We update the Mizar logic in LATIN to include Mizar's proof theory and extend the import to proofs.

T2 **Synchronization Workflow** We develop the automatic synchronization work flow that incrementally imports all changes from the Mizar library into the OAF and thus synchronizes it.

Work Package 2.2	Site	MK	FR	**all**
HOL Light	Effort (RM+RAM)	+	2+3	**2+3**

We choose HOL/Light as a prominent representative of the HOL family, where Isabelle/HOL, HOL4, and ProofPower are the main alternatives. HOL Light has the advantage of relatively coherent large libraries developed by John Harrison and within the Flyspeck project.

In addition, it provides the best export capabilities of all HOL systems – a well-maintained XML export – which makes the HOL Light import relatively easy compared to that of, e.g., Isabelle/HOL.
Collaborating expert: Cezary Kaliszyk, the main developer of the XML export

T1 **Incremental Transformation** The HOL/Light native XML export has to be read and converted into OAF. This is relatively simple. Special attention must be paid to scalability: It is necessary that the exported XML files can be imported individually. Indeed, dependencies between the files have caused major scalability problems when developing the export. (For example, the developers used single-letter XML tags to speed up writing the triple-digit-gigabyte library to hard drives.)

T2 **Structuring/Sharing** The export is currently unstructured, producing a single large (~ 100000 statements) in a homogeneous theory. We will investigate whether it can be improved to preserve more high-level structure that is implicit in the sources. This would be crucial for library integration in Work area WA3.

Work Package 2.3	Site	MK	FR	**all**
IMPS	Effort (RM+RAM)	2+	+	**2+0**

IMPS and its library were developed in the 1990s and were very influential in developing the heterogeneous method and the little-theories approach [FGT92]. The library is in danger of being lost because development of the IMPS system has been discontinued for a decade.

The use of IMPS and the development of the library have declined accordingly, and the library is less interesting from the perspective of formalizing large parts of mathematics. However, IMPS is the only major system developed using the heterogeneous method, which is crucial for library integration. In fact, the smaller IMPS library may ultimately prove more useful than larger libraries in other systems because of its better reusability. Moreover, the theories of the IMPS library provide a good starting point for the interface theories to be developed in Work package WP2.6.

We have an excellent collaboration with William Farmer and have already started the necessary exporter as a part of IMPS.

Collaborating expert: William Farmer, one of the main developers

T1 **Lutins** We represent the Lutins [J. 91] logic underlying IMPS in LATIN.

T2 **Exporter** We develop the IMPS exporter building on prior work by the first PI from summer 2009. This export will preserve the modular structure of IMPS theories completely by exporting it to the corresponding MMT primitives.

Work Package 2.4	Site	MK	FR	**all**
Matita	Effort (RM+RAM)	+	6+	**6+0**

Matita and Coq are the main representatives of the family of deduction systems based on constructive type theory, specifically the calculus of constructions. The choice between the two is difficult. On the one hand, Coq is more more widely used and provides larger libraries, in particular the one developed by Georges Gonthier. On the other hand, Matita is younger, and its attention to modern MKM support creates better synergies with the OAF project. In particular, this synergy ensures that an export from Matita is more robust, i.e., will be maintained more reliably in the long run, than an export from Coq. Moreover, the Matita library includes a translation of the Coq library.

Therefore, we prioritize Matita at this point, delaying the integration Coq to future work.

Collaborating expert: Claudio Sacerdoti Coen, one the main developers

T1 **CiC in LATIN** For the export we have to implement the underlying logic (the Calculus of Inductive Constructions) in the LATIN logic atlas. As dependent types are already supported by MMT, this will mainly involve supporting universes as well as inductive and record types.

T2 **Representation of Proofs** Matita is a good candidate for the uniform representation of high-level proofs due to its modern tactic language and experiments with the $\bar{\lambda}\mu\tilde{\mu}$ calculus [SC06; ASC06]. Based on this, we will develop a universal format for the structured proofs of deduction systems that will also subsume the proofs of Mizar.

T3 **Exporter** We can realize a direct export from Matita into MMT. As for HOL Light, special attention must be paid to exposing as much of the implicit heterogeneous structure as possible.

Work Package 2.5	Site	MK	FR	**all**
PVS	Effort (RM+RAM)	6+	+	**6+0**

We chose the PVS system as a major system used specifically in software engineering. PVS maintains only a relatively small centralized library as a part of the PVS distribution. Larger libraries, which are not systematically distributed, have been developed decentrally. The most important one is the NASA Langley library[7].

Here the integration into OAF can be a major advantage to the PVS community because it provides a central library hosting infrastructure. This benefit to the PVS community helps maintaining the export in the long run.

Collaborating expert: Natarajan Shankar, one the main developers

T1 **PVS Logic** We represent the PVS logic in LATIN by composing the available logic modules.

T2 **Exporter** Based on the existing – outdated – export developed by the first proposer, and a native XML exporter from the LogoSphere project we develop a new export and integrate it into the PVS system. Heterogeneous PVS features, in particular theories, are exported in terms of their MMT analogues.

Work Package 2.6	Site	MK	FR	**all**
Standard Interface Library	Effort (RM+RAM)	+	3+6	**3+6**

In this work package, according to the framework developed in Work area WA3, we formalize concrete interface logics and interface theories that cover the most important parts of the imported libraries. Regarding the interface logics, many can already be obtained by combining the logic features already defined in LATIN. Regarding interface theories, we will pick the most important domains, e.g., real numbers or lists, as a starting point.

[7]`http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html`

We envision this interface library to become for interactive theorem proving what the TPTP library is for automated theorem proving. It will constitute a suite of reference formalizations of frequently used domains that is independent of particular deduction systems. These formalizations will focus on collecting and stating theorems rather than proving them, and thus form challenge problems for formalization.

T1 Flexible Type Theory We develop *flexible type theory* as a family of interface logics similar to how LATIN has developed flexible formalizations of logics. These are modular formalizations of type theories together with their set theoretical semantics. Modules will include both typical primitive type constructors (functions, products, etc.) as well as less common derived type constructors (e.g., power types and quotient types). Explicating the latter permits writing concise interface theories without committing to system-specific definitions.

T2 Interface Theory Graph We identify valuable interface theories based on an analysis of the imported libraries. Then we formalize these theories in the appropriate logic. Theory and logic morphisms are used to interpret the interfaces in the individual libraries.

Work Area 3: Library Integration

In this work area, we conduct the research into library integration described in Objective O3. We develop several novel techniques to find, enable, and exploit integration potential. Each work package corresponds to one such technique, and all work packages are largely independent from each other.

The library integration problem exists mainly because of the use of the homogeneous method in current deduction systems. Therefore, many of these techniques are aimed at in some way making use of the heterogeneous structure that is typically implicitly available in the formalizations.

The main deliverables of this work area are *i*) the development of the notion and system support for interface theories, which act as theory-level specifications for formalized libraries, and *ii*) two semantic services for enhancing the structure of and connectivity between theories based on an inspection of their logical structure. Both are crucial first steps to support library integration.

This work area depends on some of the imports from Work area WA2 being available for testing and evaluation. Therefore, it is scheduled roughly in the second half of the project.

Work Package 3.1	Site	MK	FR	**all**
Interface Logics	Effort (RM+RAM)	+	4+	**4+0**

In this work package, we develop the method of interface logics, which abstract from the inner structure of (a set of) formalized theories.

Very often, a foundation is substantially more expressive than the minimal logic needed to formalize the theory under consideration. For example, Mizar [TB85] uses axiomatic set theory and Coq [Coq14] the calculus of constructions as foundations. Then both employ sophisticated definition principles to define the real numbers and prove their properties. But the minimal logic for the real numbers is much weaker than either foundation, namely first-order logic extended with a quantifier over sets of real numbers. While giving a logic translation between Coq and Mizar can be prohibitively difficult, importing this minimal logic logic into most deduction systems is straightforward. We dub these minimal logics *interface logics* and their theories (e.g., the real numbers from above) *interface theories*.

Typically, almost all of the complexity of a foundation stems from the type theory, the definition principles, and the proof language. But these are less interesting or even harmful for library integration. Instead, library integration benefits from high-level interfaces that abstract from the details of the type theory and thus maximize reuse. In particular, often theorems of interest can be stated in relatively weak interface logics, whereas the proofs depend heavily on the deduction system's foundation. Therefore, interface theories can often omit the definientia and proof objects (while keeping the types of defined objects and the theorems).

According to the heterogeneous method, we want to formulate every theory in the weakest possible interface logic. Here we leverage LATIN's logic development, which already provides a large fine-grained variety of interface logics. Given a library based on these interface logics, a deduction system only has to import the appropriate interface logic to obtain imports of all developments based on it. Conversely, the interface logics act as targets for the export of theorems.

In some sense, the use of interface logics inverts the typical design of deduction systems: Successful deduction systems are usually simple formal languages that are logically very expressive (because such languages can be fixed and implemented once and for all). Interface logics, on the other hand, should be as inexpressive as possible, even if that requires a relatively complex formal language. This inversion is necessary because the properties that make deduction system successful in the homogeneous method, also make library integration difficult.

$$\begin{tikzcd} & \boxed{\begin{matrix} T \\ c, \ldots \end{matrix}} \arrow[dl, "v_1"'] \arrow[dr, "v_2"] & \\ \boxed{S_1} \arrow[rr, leftrightarrow, dotted, "v_1(c) \approx v_2(c)"'] & & \boxed{S_2} \end{tikzcd}$$

T1 Theoretical Basis We develop the theoretical basis of interface logics and theories in terms of theory morphisms in the LATIN framework. This includes a formal definition of whether and how a concrete foundation realizes an interface logic. The main intuition is that an MMT morphism $v : T \to S$ witnesses how a system S realizes an interface T.

T2 Alignment We develop the concept of alignments between library fragments. Intuitively, consider an interface theory T that is realized by two systems S_1 and S_2 witnessed by two theory morphisms $v_i : T \to S_i$. For a T-symbol c, we say that $v_1(c)$ and $v_2(c)$ are *aligned* under the span (v_1, v_2).
Alignments provide a semantically backed concept for cross-references between libraries. They also lead to the notion of alignment-respecting library translations, which provide a starting point to overcome the library integration problem.

T3 Theorem Transport As a first step towards alignment-respecting translations, we develop conditions under which theorems can be moved along alignments. As a main criterion, we develop a notion of *conservativity* of realizations: Intuitively, $v_1 : T \to S_1$ is conservative if S_1 can only prove statements that are also true in T. In that case, theorems in S_1 give rise to theorems in S_2. We presented a first idea on how to obtain conservativity in [KRSC11] by using a partial inverse of v_1.

Work Package 3.2	Site	MK	FR	**all**
Pragmatic Interfaces	Effort (RM+RAM)	+	2+6	**2+6**

Formalizations often have to code high-level principles in terms of more primitive ones. For example, a case-based definition of a first-order function symbol can be coded as as an undefined function symbol together with one axiom for each case. Most deduction systems support such high-level statements and elaborate them internally. Typical examples are subtype definitions (e.g., in HOL), implicit and case-based functions (e.g., in Mizar), and recursive functions (e.g., in Coq).

The high-level formulations are much more suitable for library integration than the elaborated version because they hide system-specific formalization details. In particular, deduction systems often offer the same or very similar high-level statements (because these are driven by domain-specific considerations rather than system-specific ones), but may code them in very different system-specific ways. However, current deduction systems often are not able to preserve the high-level formulation. Instead, they compute and work with the elaborated version, from which the high-level formulation cannot be recovered easily.

In the LATIN project, we developed declaration patterns – schemata for groups of declarations that occur together. We showed in [HKR12] that they can be used to represent such high-level statements, which we called *pragmatic statements*. For example, the case-based function symbol definition from above would be a single pragmatic statement, whose elaboration produces a group of low-level statements (the undefined symbol and the axioms).

In this work package, we build on these results to use pragmatic statements as interfaces between libraries.

T1 Extension of logic representations For all logics imported into OAF, we add their pragmatic statement schemata to the representations of their logics. Furthermore, we enrich the interface logics from Work package WP3.1 with corresponding pragmatic statement schemata.

T2 Pragmatics-preserving exports We investigate how the library exports from Work area WA2 can be modified to preserve pragmatic statements wherever possible. This depends strongly on implementation details of the respective deduction system because systems do not necessarily keep track of the pragmatic version.

T3 Pragmatics-based integration The above tasks result, respectively, in interface logics that include pragmatic statement schemata and in libraries, whose statements make use these schemata. Based on that, we can build library translations at the pragmatic level, i.e., translations that ignore the elaboration and map pragmatic statements to their counterparts.

Work Package 3.3	Site	MK	FR	**all**
Interface Theory Generation	Effort (RM+RAM)	+	2+6	**2+6**

While the interface logics from Work package WP3.1 have already been built in LATIN, only few interface theories exist, i.e., theory developments that systematically use the weak interface logics. And in Work package WP2.6, we can only build so many of them manually.

Obtaining interface theories from existing libraries of deduction systems is difficult – we have to *i*) choose an appropriate subset of a homogeneous library (e.g., the real numbers), *ii*) choose a set of definientia and proofs to abstract from, *iii*) determine the weakest logic in which the resulting theory – which we call an interface theory – can be expressed. This process has to be repeated often to obtain many interface theories. Moreover, the necessary choices often requires human intelligence.

Therefore, we investigate the most promising methods to partially automate this process. In particular, step *iii*) is amenable to automation because we can check which logic features are used in the theory. Moreover, heuristic methods can be used to identify candidates for step *i*) and *ii*). For example, it often makes sense to put theorems from the same source file into one interface theory.

T1 Interface Extraction We specify and automate the extraction of an interface theory given that the information from *i*), *i*), and *iii*) is provided.

T2 Automation We automate step *iii*) from above.

T3 Case Studies We conduct case studies to determine what methods for automating steps *i*) and *ii*) are successful in practice and develop these further.

Work Package 3.4	Site	MK	FR	**all**
Theory Refactoring	Effort (RM+RAM)	5+6	+	**5+6**

In this work package, we investigate methods to refactor homogeneous libraries in ways that make them more heterogeneous, i.e., to introduce a modular structure of theories and imports between them. This is currently only possible for humans, and essentially no tool support is available. Two refactoring operations are especially important.

Firstly, we can increase the number of theories by identifying subsets of the library that should be grouped into separate theories (*theory building*). These subsets must be self-contained except for possibly importing other theories. Moreover, they should consist of a few primitive and a large number of derived statements that are often used together.

Secondly, if some theory structure already exists, we can identify overlap between theories, i.e., a group of statements that is duplicated (*theory intersection*). These can be factored out into a separate theory and replaced with imports. Such overlap is particularly likely to exist in those libraries that are written by multiple people with little or no coordination (e.g., the Mizar or Coq library) or when multiple libraries in one system are written independently (e.g., in PVS).

T1 Library transformations We develop basic library transformation operations for theory building and theory intersection. These compute the necessary changes to the library and apply them.

T2 Heuristic Building We develop heuristic methods for identifying subsets of a library that are suitable for theory building. This will be based on an analysis of the dependency relation between statements – information that often has to be provided by the respective deduction system and has become available only recently (e.g., [AMU12]).

T3 Heuristic Intersection We develop a heuristic method for identifying candidates for theory intersection. This will generalize the existing method employed in [Nor08] to the logic-independent level.

Work Package 3.5	Site	MK	FR	**all**
View Finder	Effort (RM+RAM)	5+2	+	**5+2**

The heterogeneous method makes use of two kinds of theory morphisms that are at the center of the MMT language. Firstly, imports reuse existing theories when building new theories. We investigate the discovery of such structure in Work package WP3.4. In this work package, we investigate the discovery of the second kind of relation: *views*, which are morphisms between two existing theories.

For example, the theory of groups is typically built using imports from the theory of monoids. Views are used to show that further theories are subsumed by groups, e.g., the theory of loops. Because views into a theory can be given a-posteriori, they increase the reusability of theorems within a library without a need for changing the theories in it.

We investigate heuristic methods for automatically finding views, potential views, or approximate views. Potential views occur when structural similarities between theories are found but automatic verification of views is undecidable. Approximate views arise when verification of views fails "closely" – for example, if all but one axiom of theory S can be represented in theory T, appropriate refactoring of S yields an exact view from a subtheory of S.

Some special cases are particularly interesting. Views from an interface theory into a library demonstrate that a library realizes the interface, which permits importing its theorems. And views within a library indicate overlap between libraries that can be exploited for reuse or refactoring. More generally and ambitiously, we will also look for views across libraries, but it is difficult to predict how successful automated methods can be in that case.

T1 Method We generalize the theory normalization method developed in [Nor08] for first-order logic to arbitrary logics. This method detects structural similarities between theories to generate and possibly check candidates for views.

T2 Evaluation We apply the method to the overall OAF archive and evaluate the results.

Work Area 4: Global Services

In this work area we develop and implement a set of services leveraging the unified library of formalizations we import into OAF. OAF is uniquely prepared for two kinds of services. Firstly, *global* services are services that are most valuable if applied to multiple libraries at once. Secondly, *generic* services are services that can be applied to multiple libraries in the same way and thus only have to be developed once.

The main deliverables are the development of two global services in WP4.1 and and WP4.2 and two generic services from WP4.4 and WP4.3. The individual services are developed independently throughout the project duration.

Work Package 4.1	Site	MK	FR	**all**
Search & Querying	Effort (RM+RAM)	6+6	+	**6+6**

Search and querying are among the most interesting global services because users can receive answers from multiple libraries for the same search query. This is impossible without an infrastructure like ours. Our infrastructure also permits generic search and querying services that go beyond the – often text-based or ad hoc – solutions in existing deduction systems.

Building on our existing unification-based formula-search engine [KI12; KMP12] and our MMT-based structured query language [Rab12], we design a query language that comprises all aspects of the OAF libraries. Searchable information will include formulas, statements, modular structure, dependencies between statements, and meta-data. Queries will combine unification queries, compositional query evaluation, and semantic web-style RDF-based queries.

T1 Query language and engine We design the query language and implement the indexing and retrieval components and practical user interfaces. Particular emphasis will be set on extending the unification indexes to typed term structures.

T2 Indexing induced statements We develop advanced indexing techniques that permit indexing induced statements, i.e., statements that are not physically present in a library because they are induced by reuse along theory morphisms (imports or views).

T3 Search modulo alignment We realize search up to alignment. This permits the registration of alignments to find results in multiple libraries.

T4 Applicable theorem search We customize indexing and querying to permit searching for theorems that are applicable in a given context. We already provided such a service for Mizar [**IanKohRabUrb:tmmliotaa11:base**].

Work Package 4.2	Site	MK	FR	**all**
Versioning and Management of Change	Effort (RM+RAM)	+	5+6	**5+6**

Management of Change (MoC) is an essential ingredient for any archive of interdependent content; and this is especially true for formal/mathematical archives, which are constantly being extended, generalized, and refactored: when an axiom, definition, or proof tactic changes, all theorems that use them need to be re-proved.

MoC is one of the most suitable examples of a generic service: It is becoming more and more important in practice as libraries and teams are growing, but it is conceptually and technically so expensive that no single deduction system community has had the resources to develop a strong solution. Our experience has shown [Aut+11; IR12b] that our uniform representation language is well-suited for generic change management.

T1 Dependency relations We amend library exports from deduction systems to include the dependency relation between statements. This is a crucial prerequisites to propagate the impacts of a change. Such dependency information must be provided by the deduction system as recently demonstrated in [AMU12].

T2 MoC service Based on the dependency relation, we implement a generic change management service at the OAF level. This will proceed along the lines of [IR12b]. The most important applications are to compute all differences between two versions of the same library and to compute the impact (i.e., the theorems that have to be reproved) of a possible change.

T3 User interface We develop a user interface for requesting and visualizing the change management results of the MoC service.

T4 Workflows We develop work flows for feeding back the results of our change management analysis into specific deduction systems. We plan to use the Mizar library as a case study here: We want to provide our service to the Mizar library committee as a tool to manage successive releases of the Mizar library.

Work Package 4.3	Site	MK	FR	**all**
Generic Type/Proof Inference	Effort (RM+RAM)	+	6+3	**6+3**

In user interfaces, such as web front-ends or IDEs, it is important to display the types of subterms to users. Due to the Curry-Howard representations of proofs-as-terms, this includes the inference of assumptions and conclusions of individual proof steps. This is straightforward only for very tight integrations between user interface and a fixed deduction system – which usually do not exist.

But in MKM applications that operate without the support of the deduction system, it is extremely difficult. It is usually not feasible either, to call the deduction system for interactive type inference queries because each query requires the deduction system to load the complete context (which may be any subset of the whole library). And even then, the connection between application and deduction system would have to be implemented for every system.

Instead, we can use the representation of the foundations in the LATIN framework, where LF-based algorithms can induce type inference generically for all foundations.

T1 Church-style encodings We develop a type inference service for LF within MMT. This immediately provides type inference for foundations whose typing relation is represented in terms of LF typing (Church style encodings).

T2 Curry style encodings Curry-style encodings represent typing as an LF predicate with explicit typing rules. This is more flexible and permits, e.g., elegant representations of subtyping and undecidable type systems. We build an engine that transforms a Curry-style LF encoding into an executable type inference algorithm.

Work Package 4.4	Site	MK	FR	**all**
Active Libraries	Effort (RM+RAM)	4+6	+	**4+6**

In this work package, we develop special logic-oriented features for the web front end from Work package WP1.2. Current library browsers for individual systems usually offer very little interaction because all web pages are precomputed and served by a standard HTML server. In contrast, we can employ our MKM technology, e.g., [GLR09], to obtain a truly interactive global library browsing service. This includes, for example, dynamic folding of subterms, navigation based on theory graphs, dynamic display of bracketing and inferred types and arguments.

This also includes cross-referencing between aligned formalizations in different libraries. By navigating along alignments, users can compare different formalizations of the same domain or mentally abstract from the technical gaps between them.

T1 Basic interface We develop a basic interface for displaying individual documents, theories, and statements in a library. This includes cross-references, syntax coloring, and interactive highlighting and folding. The presentation is built dynamically using Ajax requests to the MMT HTTP server, thus permitting very flexible display components.

T2 Graph-based interface We develop a theory graph-based interface that follows the logical structure of the library displaying connected by imports and views. Both the hierarchic and the graph-based interface can span libraries by connecting theories along alignments.

T3 Integration of the other services We integrate the search interface from Work package WP4.1 to choose dynamically which items to display. We integrate the change management service from Work package WP4.2 to visualize differences between versions and impacts of changes. And we integrate the type inference service from Work package WP4.3 to permit the dynamic type inference of any subterm the user selects.

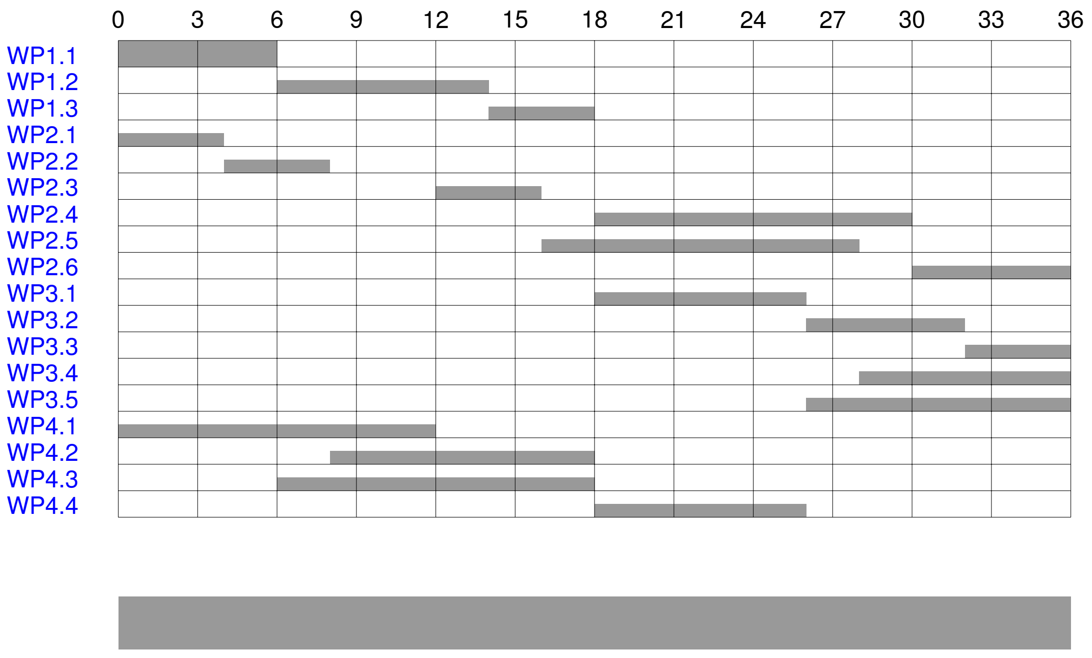

Figure 1: Gantt Chart: Overview Work Package Activities– lower bar shows the overall effort (RM only) per month

2.4 Data Handling

The OAF project will not systematically produce research data.

2.5 Other Information

None

2.6 Explanations on the Proposed Investigations

Not applicable.

[7]Bars shown at reduced height (e.g. 50%) indicate reduced intensity during that work phase (e.g. to 50%).

2.7 Information on Scientific and Financial Involvement of International Cooperation Partners

Not applicable.

3 Bibliography concerning the state of the art, the research objectives, and the work programme

[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.

[AMU12] J. Alama, L. Mamane, and J. Urban. “Dependencies in Formal Mathematics: Applications and Extraction for Coq and Mizar.” In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. Campbell, J. Carette, G. D. Reis, P. Sojka, M. Wenzel, and V. Sorge. Springer, 2012, pp. 1–16.

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

[ASC06] S. Autexier and C. Sacerdoti Coen. “A Formal Correspondence Between OMDoc with Alternative Proofs and the $\overline{\lambda}\mu\tilde{\mu}$-calculus.” In: *Mathematical Knowledge Management (MKM)*. Ed. by J. Borwein and W. M. Farmer. LNAI 4108. Springer Verlag, 2006, pp. 67–81.

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

[Aut+11] S. Autexier, C. David, D. Dietrich, M. Kohlhase, and V. Zholudev. “Workflows for the Management of Change in Science, Technologies, Engineering and Mathematics.” In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, F. Rabe, and J. Urban. LNAI 6824. Springer Verlag, 2011, pp. 164–179. ISBN: 978-3-642-22672-4. URL: http://kwarc.info/kohlhase/papers/planetary-moc.pdf.

[BCH12] M. Boespflug, Q. Carbonneaux, and O. Hermant. “The $\lambda\Pi$-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] G. Bancerek and M. 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] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaëtano, and M. Kohlhase. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: http://www.openmath.org/standard/om20.

[Car+09] J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt, eds. *MKM/Calculemus Proceedings*. LNAI 5625. Springer Verlag, July 2009. ISBN: 978-3-642-02613-3.

[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, M. Kohlhase, T. Mossakowski, and F. Rabe. “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.

[Cod+12] M. Codescu, F. Horozal, M. Kohlhase, T. Mossakowski, F. Rabe, and K. Sojakova. “Towards Logical Frameworks in the Heterogeneous Tool Set Hets.” In: *Recent Trends in Algebraic Development Techniques 2010*. Ed. by T. Mossakowski and H. Kreowski. Springer, 2012, pp. 139–159.

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

[Dav+11] J. Davenport, W. Farmer, F. Rabe, and J. Urban, eds. *Intelligent Computer Mathematics*. LNAI 6824. Springer Verlag, 2011. ISBN: 978-3-642-22672-4.

[Dav+12] C. David, C. Jucovschi, A. Kohlhase, and M. Kohlhase. “Semantic Alliance: A Framework for Semantic Allies.” In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 49–64. ISBN: 978-3-642-31373-8. URL: http://kwarc.info/kohlhase/papers/mkm12-SAlly.pdf.

[DK09a] J. H. Davenport and M. Kohlhase. “Quantifiers and Big Operators in OpenMath.” In: *22nd OpenMath Workshop*. Ed. by J. H. Davenport. July 2009. URL: http://kwarc.info/kohlhase/papers/om09-quantifiers.pdf.

[DK09b] J. H. Davenport and M. Kohlhase. “Unifying Math Ontologies: A tale of two standards.” In: *MKM/Calculemus Proceedings*. Ed. by J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt. LNAI 5625. Springer Verlag, July 2009, pp. 263–278. ISBN: 978-3-642-02613-3. URL: http://opus.bath.ac.uk/13079/1/MKM2009v2.pdf.

[DW97] I. Dahn and C. 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 U. Furbach and M. P. 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.

[GB92] J. Goguen and R. Burstall. “Institutions: Abstract Model Theory for Specification and Programming.” In: *Journal of the Association for Computing Machinery* 39(1) (1992), pp. 95–146.

[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.

[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.

[HKR11] F. Horozal, M. Kohlhase, and F. Rabe. “Extending OpenMath with Sequences.” In: *Intelligent Computer Mathematics, Work-in-Progress Proceedings*. Ed. by A. Asperti, J. Davenport, W. Farmer, F. Rabe, and J. Urban. University of Bologna, 2011, pp. 58–72.

[HKR12] F. Horozal, M. Kohlhase, and F. Rabe. “Extending MKM Formats at the Statement Level.” In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 65–80. ISBN: 978-3-642-31373-8. URL: http://kwarc.info/kohlhase/papers/mkm12-p2s.pdf.

[Hor+11] F. Horozal, A. Iacob, C. Jucovschi, M. Kohlhase, and F. Rabe. “Combining Source, Content, Presentation, Narration, and Relational Representation.” In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, F. Rabe, and J. Urban. LNAI 6824. Springer Verlag, 2011, pp. 212–227. ISBN: 978-3-642-22672-4. URL: http://kwarc.info/frabe/Research/HIJKR_dimensions_11.pdf.

[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] P. Horn and D. Roozemond. “OpenMath in SCIEnce: SCSCP and POPCORN.” In: *MKM/Calculemus Proceedings*. Ed. by J. Carette, L. Dixon, C. Sacerdoti Coen, and S. 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. D. Reis and L. Théry. ACM, 2009, pp. 31–37.

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

[IR12a] M. Iancu and F. Rabe. “(Work-in-Progress) An MMT-Based User-Interface.” In: *Workshop on User Interfaces for Theorem Provers*. Ed. by C. Kaliszyk and C. Lüth. 2012.

[IR12b] M. Iancu and F. Rabe. “Management of Change in Declarative Languages.” In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 325–340. ISBN: 978-3-642-31373-8.

[Jeu+12] J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge, eds. *Intelligent Computer Mathematics*. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012. ISBN: 978-3-642-31373-8.

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

[KK09a] A. Kohlhase and M. Kohlhase. “Compensating the Computational Bias of Spreadsheets with MKM Techniques.” In: *MKM/Calculemus Proceedings*. Ed. by J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt. LNAI 5625. Springer Verlag, July 2009, pp. 357–372. ISBN: 978-3-642-02613-3. URL: http://kwarc.info/kohlhase/papers/mkm09-sachs.pdf.

[KK09b] A. Kohlhase and M. Kohlhase. “Spreadsheet Interaction with Frames: Exploring a Mathematical Practice.” In: *MKM/Calculemus Proceedings*. Ed. by J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt. LNAI 5625. Springer Verlag, July 2009, pp. 341–356. ISBN: 978-3-642-02613-3. URL: http://kwarc.info/kohlhase/papers/mkm09-framing.pdf.

[KK13] A. Kohlhase and M. Kohlhase. “Spreadsheets with a Semantic Layer.” In: *Electronic Communications of the EASST: Specification, Transformation, Navigation – Special Issue dedicated to Bernd Krieg-Brückner on the Occasion of his 60th Birthday* 62 (2013). Ed. by T. Mossakowski, M. Roggenbach, and L. Schröder, pp. 1–20. URL: http://journal.ub.tu-berlin.de/eceasst/article/view/870.

[Kle+10] G. Klein 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] M. Kohlhase, B. A. Matican, and C. C. Prodescu. “MathWebSearch 0.5 – Scaling an Open Formula Search Engine.” In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. 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.

[KMR09] M. Kohlhase, T. Mossakowski, and F. Rabe. *The LATIN Project*. see https://trac.omdoc.org/LATIN/. 2009.

[Koh+09] M. Kohlhase, J. Lemburg, L. Schröder, and E. Schulz. “Formal Management of CAD/CAM Processes.” In: *16th International Symposium on Formal Methods (FM 2009)*. Ed. by A. Cavalcanti and D. Dams. LNCS 5850. Springer Verlag, 2009, pp. 223–238. URL: http://kwarc.info/kohlhase/papers/fm09.pdf.

[Koh+11] M. Kohlhase et al. “The Planetary System: Web 3.0 & Active Documents for STEM.” In: *Procedia Computer Science* 4 (2011): *Special issue: Proceedings of the International Conference on Computational Science (ICCS)*. Ed. by M. Sato, S. Matsuoka, P. M. Sloot, G. D. van Albada, and J. Dongarra. Finalist at the Executable Paper Grand Challenge, pp. 598–607. DOI: 10.1016/j.procs.2011.04.063.

[Koh06a] M. Kohlhase, ed. *Mathematical Knowledge Management, MKM’05*. LNAI 3863. Springer Verlag, 2006.

[Koh06b] M. 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.

[Koh13] M. Kohlhase. “Knowledge Management for Systematic Engineering Design in CAD Systems.” In: *Professionelles Wissenmanagement Management, Konferenzbeiträge der 7. Konferenz*. Ed. by F. Lehner, N. Amende, and N. Fteimi. GITO Verlag, 2013, pp. 202–217. ISBN: 978-3-95545-016-8. URL: http://kwarc.info/kohlhase/papers/wm13-formalcad.pdf.

[KR12] M. Kohlhase and F. Rabe. “Semantics of OpenMath and MathML3.” In: *Mathematics in Computer Science* 6.3 (2012), pp. 235–260. URL: http://kwarc.info/kohlhase/papers/mcs12.pdf.

[KRSC11] M. Kohlhase, F. Rabe, 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. LNAI 6824. Springer Verlag, 2011, pp. 107–122. ISBN: 978-3-642-22672-4. URL: http://kwarc.info/kohlhase/papers/cicm11-integration.pdf.

[KRZ10] M. Kohlhase, F. Rabe, and V. Zholudev. “Towards MKM in the Large: Modular Representation and Scalable Software Architecture.” In: *Intelligent Computer Mathematics*. Ed. by S. Autexier, J. Calmet, D. Delahaye, P. D. F. Ion, L. Rideau, R. Rioboo, and A. P. Sexton. LNAI 6167. Springer Verlag, 2010, pp. 370–384. ISBN: 3642141277. arXiv: 1005.5232v2 [cs.OH].

[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] M. Kohlhase and I. Şucan. “A Search Engine for Mathematical Formulae.” In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC’2006*. Ed. by T. Ida, J. Calmet, and D. Wang. LNAI 4120. Springer Verlag, 2006, pp. 241–253. URL: http://kwarc.info/kohlhase/papers/aisc06.pdf.

[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.

[MML07] T. Mossakowski, C. Maeder, and K. Lüttich. “The Heterogeneous Tool Set.” In: *Tools and Algorithms for the Construction and Analysis of Systems 2007*. Ed. by O. Grumberg and M. Huth. Vol. 4424. Lecture Notes in Computer Science. 2007, pp. 519–522.

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

[Nor08] I. Normann. “Automated Theory Interpretation.” PhD thesis. Bremen, Germany: Jacobs University, 2008. URL: https://svn.eecs.jacobs-university.de/svn/eecs/archive/phd-2008/normann.pdf.

[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: *Automated Reasoning*. Ed. by N. Shankar and U. Furbach. Vol. 4130. Springer, 2006.

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

[Pfe+03] F. Pfenning, C. Schürmann, M. Kohlhase, N. Shankar, and S. Owre. *The Logosphere Project*. http://www.logosphere.org/. 2003.

[PM] *PlanetMath.org – Math for the people, by the people*. URL: http://planetmath.org (visited on 11/11/2012).

[PR] *Panta Rhei, the Active Course Site at Jacobs University*. URL: http://panta.kwarc.info (visited on 02/22/2013).

[Rab12] F. Rabe. “A Query Language for Formal Mathematical Libraries.” In: *Intelligent Computer Mathematics*. Ed. by J. Jeuring, J. A. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 142–157. ISBN: 978-3-642-31373-8. arXiv: 1204.4685 [cs.LO].

[Rab13] F. Rabe. “A Logical Framework Combining Model and Proof Theory.” In: *Mathematical Structures in Computer Science* 23.5 (2013), pp. 945–1001.

[RK13a] F. Rabe and M. Kohlhase. “A Scalable Module System.” In: *Information and Computation* 230.1 (2013), pp. 1–54.

[RK13b] F. Rabe and M. Kohlhase. “A Scalable Module System.” In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: http://kwarc.info/frabe/Research/mmt.pdf.

[SC06] C. Sacerdoti Coen. “Explanation in Natural Language of $\overline{\lambda}\mu\tilde{\mu}$-terms.” In: *Mathematical Knowledge Management, MKM'05*. Ed. by M. Kohlhase. LNAI 3863. Springer Verlag, 2006.

[SSY94] G. Sutcliffe, C. Suttner, and T. Yemenis. “The TPTP Problem Library.” In: *Proceedings of the 12th Conference on Automated Deduction*. Ed. by A. Bundy. LNAI 814. Nancy, France: Springer Verlag, 1994.

[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] C. D. Team. *The Coq Proof Assistant: Reference Manual*. INRIA. URL: https://coq.inria.fr/refman/.

[Urb03] J. Urban. “Translating Mizar for First-Order Theorem Provers.” In: *Mathematical Knowledge Management, MKM'03*. Ed. by A. Asperti, B. Buchberger, and J. H. Davenport. LNCS 2594. Springer Verlag, 2003, pp. 203–215.

[Urb06] J. Urban. “XML-izing Mizar: making semantic processing and presentation of MML easy.” In: *Mathematical Knowledge Management, MKM'05*. Ed. by M. 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 J. Jeuring, J. A. Campbell, J. Carette, G. D. Reis, P. Sojka, M. Wenzel, and V. Sorge. 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.

[ZK09] V. Zholudev and M. Kohlhase. “TNTBase: a Versioned Storage for XML.” In: *Proceedings of Balisage: The Markup Conference*. Vol. 3. Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2009. DOI: 10.4242/BalisageVol3.Zholudev01.

[CoF04] CoFI (The Common Framework Initiative). *CASL Reference Manual*. Vol. 2960. LNCS. Springer, 2004.

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

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

[Hal14] T. Hales et al. *A formal proof of the Kepler conjecture*. http://arxiv.org/abs/1501.02155. 2014.

[J. 91] J. Guttman. “An Interface Logic for Verification Environments.” In: *International Workshop on Higher Order Logic Theorem Proving and its Applications*. Ed. by M. Archer, J. Joyce, K. Levitt, and P. Windley. IEEE Computer Society Press, 1991.