Computer Supported Formal Work:
Towards a Digital Mathematical Assistant

Jörg Siekmann and Serge Autexier

DFKI GmbH & Saarland University
Saarbrücken, Germany
{siekmann|autexier}@ags.uni-sb.de
www.ags.uni-sb.de/~serge

Abstract. The year 2004 marked the fiftieth birthday of the first computer generated proof of a mathematical theorem: "the sum of two even numbers is again an even number" (with Martin Davis' implementation of Presburger Arithmetic in 1954). While Martin Davis and later the research community of automated deduction used machine oriented calculi to find the proof for a theorem by automatic means, the AUTOMATH project of N.G. de Bruijn[1] – more modest in its aims with respect to automation – showed in the late 1960s and early 70s that a complete mathematical textbook could be coded and proof-checked by a computer. Roughly at the same time in 1973, the MIZAR project[2] started as an attempt to reconstruct mathematics based on computers. Since 1989, the most important activity in the MIZAR project has been the development of a database for mathematics. International cooperation resulted in creating a database which includes more than 7000 definitions of mathematical concepts and more than 42000 theorems.

The work by Milner and others on LCF [28] spawned a research community on tactical theorem proving, which again modest with respect to automation, showed that nevertheless important sequences of deduction could be encapsulated into a tactic and then invoked by the mathematically trained user. Classical automated theorem proving procedures of today are based on ingenious search techniques to find a proof for a given theorem in very large search spaces – often in the range of several billion clauses. But in spite of many successful attempts to prove even open mathematical problems automatically, their use in everyday mathematical practice is still limited. The shift from search based methods to more abstract planning techniques however opened up a new paradigm for mathematical reasoning on a computer and several systems of the new kind now employ a mix of interactive tactic based, search based as well as proof planning based techniques. The ΩMEGA system is at the core of several related and well-integrated research projects of the ΩMEGA research group, whose aim is to develop system support for the working mathematician and formal software engineer, in particular it supports proof development at a human oriented level of abstraction. Sum-

---

[1] www.win.tue.nl/automath

[2] www.mizar.org

231

marizing what we have today in terms of technological support, we shall then address the most pressing needs of the future in this paper.

1   Introduction

Computer-based information processing plays an increasingly important role in modern societies and meanwhile touches the very basic organization of our daily life. Our dependency on the well-functioning of this information processing motivates the development of *provably* reliable software and hardware. This both includes human comprehensible specifications and their automatic or interactive verification.

Logical systems were developed originally to provide a foundation for mathematics and their application to the specification and verification of software resulted – since the mid 1950s – in *formal* software development methods. Since then, there is a myriad of formalisms and systems, both for mathematics and formal methods, which are unfortunately not always interoperable. Today even non-trivial mathematical problems can be proved by machines and formal methods have been employed for complex software and hardware developments. In order to do so, increasing parts of mathematics and software have been formalized in system-specific libraries that are mostly distributed over different physical locations.

The vision of computer-supported mathematics and a system which provides integrated support for all work phases of a mathematician (or a software engineer using formal methods) has fascinated researchers in artificial intelligence, particularly in the deduction systems area, and in mathematics for a long time (see Fig. 1). The dream of mechanizing (mathematical) reasoning dates back to Gottfried Wilhelm Leibniz in the 17th century with the touching vision that two philosophers engaged in a dispute would one day simply code their arguments into an appropriate formalism and then *calculate* (Calculemus!) who is right. At the end of the 19th century modern mathematical logic was born with Frege's *Begriffsschrift* and an important milestone in the formalization of mathematics was Hilbert's program and the 20th century Bourbakism.

With the logical formalism for the representation and calculation of mathematical arguments emerging in the first part of the twentieth century it was but a small step to implement these techniques now on a computer as soon as it was widely available. In 1954 Martin Davis' Presburger Arithmetic Program was reported to the US Army Ordnance and the Dartmouth Conference in 1956 is not only known for giving birth to artificial intelligence in general but also more specifically for the demonstration of the first automated reasoning programs for mathematics by Herb Simon and Alan Newell.

However, after the early enthusiasm of the 1960s, in particular the publication of the resolution principle in 1965 [40], and the developments in the 70s a more sober realization of the actual difficulties involved in automating everyday mathematics set in and the field increasingly fragmented into many subareas which all developed their specific techniques and systems[3].

---

[3] The history of the field is presented in a classical paper by Martin Davis [18] and also in [19] and more generally in his history of the making of the first computers [17]. Another source is Jörg Siekmann [43] and more recently [44].
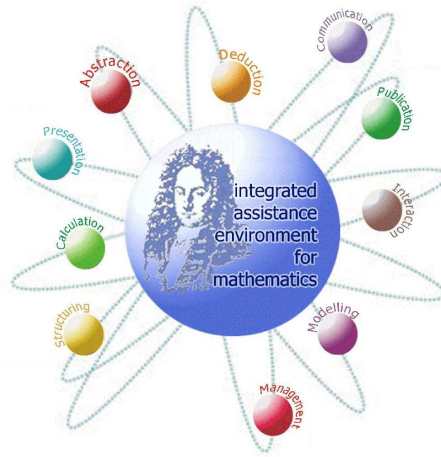
Fig. 1. CALCULEMUS illustration of different challenges for a mathematical assistant system.

Apart from its fragmentation, four major deficiencies of the proof assistant systems are in the way for a more widespread use in mathematics and software engineering. First, the user interfaces of current systems are not convenient enough to attract non-expert users. In particular, the internal representation languages of these systems are still too far away from reaching the flexibility, expressivity and elegance of common mathematical vernacular. Secondly, the formal proof assistant systems have their relative strengths and weaknesses, however they are usually incompatible, both at the conceptual and linguistic levels. Thirdly, according to the Common Criteria [39] the deliverables of a piece of software must also contain documentation or even *security targets* and *protection profiles*. However, current systems do not provide sufficient support for linking such informal or semi-formal documents to formal proof developments. The same applies for a mathematical or engineering publication: How do we link the informal text paragraphs with the formal content? Finally and fourth, the maintenance techniques of existing systems are too immature to cope with the increasing sizes and complexities of their libraries.

It is only very recently that this trend is reversed, with the CALCULEMUS[4] and MKM[5] communities as driving forces of this movement. In CALCULEMUS the viewpoint is bottom-up, starting from existing techniques and tools developed in the community. MKM approaches the goal of computer-based mathematics and formal methods in the new millennium by a complementary top-down approach starting from existing, mainly pen and paper based mathematical practice down to system support.

---

[4] www.calculemus.org

[5] monet.nag.co.uk/mkm

In both communities the MIZAR project [41, 42] served as an important existence proof, that indeed formalized mathematics is possible: with close to 8000 definitions and more than 42000 theorems represented and proved within the system is now generally viewed as a milestone.

We shall now address two questions: What do we have achieved today in the ΩMEGA project and what do we need in the future. In Sections 2 and 3 we provide an overview and the main developments of the ΩMEGA project and then point to current research and some future goals.

2 What we have: Ωmega

The ΩMEGA project[6] represents one of the major attempts to build an all encompassing assistant tool for the working mathematician. It is a representative of systems in the new paradigm of *proof planning* and combines interactive tactical and automated proof construction for domains with rich and well-structured mathematical knowledge. The inference mechanism at the lowest level of abstraction is an interactive theorem prover based on a higher order natural deduction (ND) variant of a soft-sorted version of Church's simply typed λ-calculus [16]. The logical language, which also supports partial functions, is called $\mathcal{POST}$, for *p*artial functions and *o*rder *s*orted *t*ype theory. While this represents the "machine code" of the system the user will seldom want to see, the search for a proof is usually conducted at a higher level of abstraction defined by *tactics* and *methods*. Automated proof search at this abstract level is called *proof planning*. Proof construction is also supported by already proved assertions and theorems and by calls to external systems to simplify or solve subproblems.

At the core of ΩMEGA is the *proof plan data structure $\mathcal{PDS}$* [15], in which proofs and *proof plans* are represented at various levels of granularity and abstraction (see the central part in Fig. 2). The $\mathcal{PDS}$ is a directed acyclic graph, where *open nodes* represent unjustified propositions that still need to be proved and *closed nodes* represent propositions that are already proved. The proof plans are developed and classified with respect to a taxonomy of mathematical theories in the mathematical knowledge base MBASE [26, 31]. The user of ΩMEGA, or the proof planner MULTI [36], or else the suggestion mechanism ΩANTS [11] modify the $\mathcal{PDS}$ during proof development until a complete proof plan has been found. They can also invoke external reasoning systems, whose results are included in the $\mathcal{PDS}$ after appropriate transformation. Once a complete proof plan at an appropriate level of abstraction has been found, this plan must be expanded by sub-methods and sub-tactics into lower levels of abstraction until finally a proof at the level of the logical calculus is established. After expansion of these high-level proofs to the underlying ND calculus, the $\mathcal{PDS}$ can be checked by ΩMEGA's proof checker.

Hence, there are two main tasks supported by this system, namely (i) to find a proof plan, and (ii) to expand this proof plan into a calculus-level proof; and both jobs can be equally difficult and time consuming. Task (ii) employs an LCF-style tactic expansion mechanism, proof search or a combination of both in order
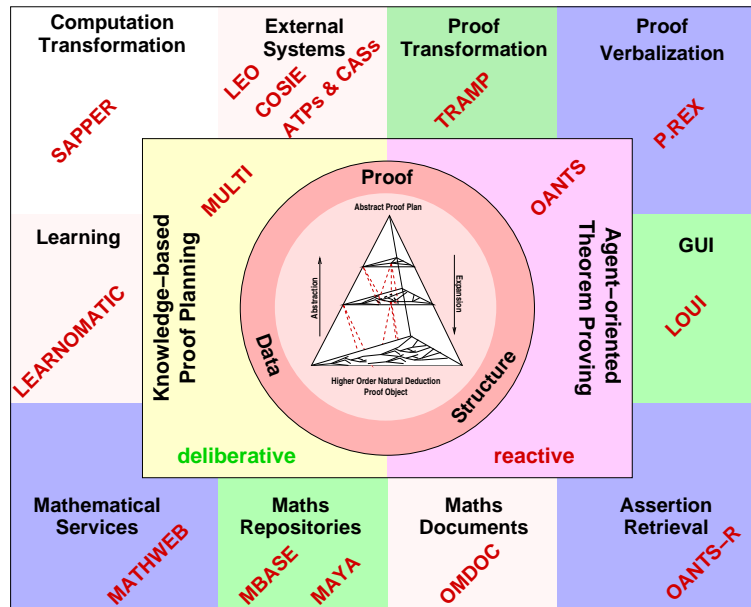
---

[6] www.ags.uni-sb.de/~omega

Fig. 2. The vision of an all encompassing mathematical assistant environment: we have now modularized and out-sourced many of the support tools such that they can also be used by other systems via the MATHWEB-SB software bus.

to generate a lower-level proof object. It is a design objective of the $\mathcal{PDS}$ that various *proof levels* coexist with their respective relationships being dynamically maintained.

The graphical user interface $\mathcal{LOUI}$ [46] provides both a graphical and a tabular view of the proof under consideration, and the interactive proof explanation system *P.rex* [24, 23, 25] generates a natural-language presentation of the proof.

The previously monolithic system has been split up and separated into several independent modules, which are connected via the mathematical software bus MATHWEB-SB [52]. An important benefit is that MATHWEB-SB modules can be distributed over the Internet and are then remotely accessible by other research groups as well. There is now a very active MATHWEB-SB user community with sometimes several thousand theorems and lemmata being proven per day. Most theorems are generated automatically as (currently non-reusable and non-indexed) subproblems in natural language processing (see the Doris system [21]), proof planning and verification tasks.

For more details about the ΩMEGA system and research activities around the system we refer the interested reader to [45, 3] and to the web-page of the project (www.ags.uni-sb.de/~omega).

3   What do we need in future?

The vision of a powerful mathematical assistance environment which provides computer-based support for most tasks of a mathematician has stimulated new projects and international research networks across the disciplinary and systems boundaries. Examples are the European Calculemus[7] (Integration of Symbolic Reasoning and Symbolic Computation) and Mkm[8] (Mathematical Knowledge Management, [13]) initiatives, the EU projects Monet[9], Openmath and Mowgli[10], and the American Qpq[11] repository of deductive software tools. Of course, a milestone in this area has been the Mizar project[12], and there are now numerous national projects in the US and Europe, which cover partial aspects of this vision, such as knowledge representation, deductive system support, user interfaces, mathematical publishing tools, etc.

The longterm goal of the Ωmega project is the all-embracing integration of symbolic reasoning, i.e. computer algebra and deduction systems, into mathematical research, mathematics education, and formal methods in computer science. We anticipate that in the long run these systems will change mathematical practice and they will have a strong societal impact, not least in the sense that a powerful infrastructure for mathematical research and education will become commercially available. Computer supported mathematical reasoning tools and integrated assistance systems will be further specialized to have a strong impact also in many other theoretical fields such as safety and security verification of computer software and hardware, theoretical physics and chemistry and other related subjects.

Our current approach is strictly bottom-up: Starting with existing techniques and tools of our partners for symbolic reasoning (deduction) and symbolic computation (computer algebra), we will step by step improve their interoperability up to the realization of an integrated systems via the mathematical software bus MathWeb-SB. The envisaged system will support the full life-cycle of the evolutionary nature of mathematical research (see Fig. 3) helping an engineer or mathematician who works on a mathematical problem in the improvement, the exploration, the distributed maintenance, the retrieval and the proving and calculation tasks and finally the publication of mathematical theories.

So what does this vision entail in the immediate future? We want to develop methods and tools for a better integration of formal proof systems into everyday mathematical activity and into the development of provably reliable software. Four major aspects are:

– The development of non-proprietary, *semi-formal representation formats* which support a continuous range of proof details in informal mathematical documents.

---

[7] www.calculemus.org

[8] monet.nag.co.uk/mkm

[9] monet.nag.co.uk/cocoon/monet

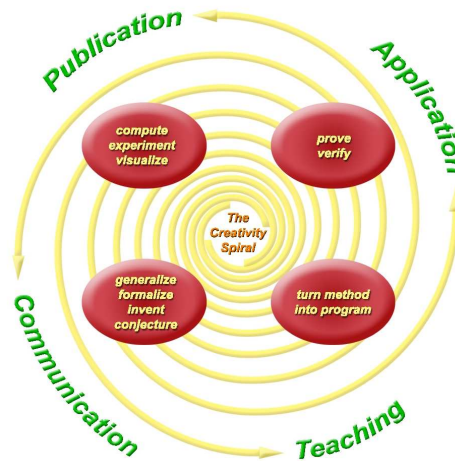[10] www.mowgli.cs.unibo.it

[11] www.qpq.org

[12] www.mizar.org

Fig. 3. Mathematical Creativity Spiral; [by Buchberger in 1995]

– Support for the interactive development of semi-formal documents inside an existing open-source scientific WYSIWYG text editor.
– The integration of existing mathematical assistant and verification systems into such an editor.
– The development of sophisticated, semantics-based truth-maintenance techniques for the distributed libraries *inclusive of* their mathematical documents.

Bridging the gap from the informal common mathematical language to the formal proof assistant oriented languages will provide software developers (respectively working mathematicians[13]) access to the reasoning capabilities and the pieces of formalized mathematics for their support.

More specifically, we are working currently towards the following goals:

3.1 Formalization and Proving at a Higher Level of Abstraction

Mathematical reasoning with the ΩMEGA system is at the comparatively high level of abstraction of the proof planning methods. However, as these methods have to be expanded eventually to the concrete syntax of our higher order ND-calculus, the system still suffers from the effect and influence this logical representation has. In contrast, the proofs developed by a mathematician, say for a mathematical publication, and the proofs developed by a student in a mathematical tutoring system are

---

[13] We do not expect the high class mathematicians at the forefront of mathematical research to profit from these developments in the foreseeable future. However many mathematical application areas (such as statistics for constructing bridges or buildings, some areas of theoretical physics or mechanical engineering, etc.) are close to formalization anyway, and may well be amendable using computer algebra and automated reasoning tools.

typically developed at an argumentative level. This level has been formally catego-
rized as *proofs at the assertion level* [29] with different types of *under-specification*
[2].[14] The CORE calculus [1, 7] for contextual reasoning has been designed to sup-
port reasoning directly on the assertion level and our current goal, which is by large
completed, is to completely exchange the natural deduction calculus by the CORE
calculus. The CORE calculus has been developed for a large class of logics, com-
prises a uniform notion of a logical context of subformulas as well as replacement
rules available in a logical context. Replacement rules operationalize assertion level
proof steps and technically are generalized resolution and paramodulation rules,
which in turn should suit the implementation of automatic reasoning procedures.

The exchange of the logic layer in ΩMEGA requires the adaptation of all rea-
soning procedures that are currently tailored to it, including proof planning and
the integration of external systems. The first step in that direction was to stan-
dardize the basic proof construction mechanism used by both the human user and
the automatic reasoning procedures that previously used proprietary mechanisms.
This is realized in the so-called *tasklayer* [20] which is an instance of the new proof
datastructure (PDS) [4]. The nodes of the PDS are annotated with *tasks*, which
are Gentzen-style multi-conclusion sequents augmented by means to define multi-
ple foci of attention on subformulas that are maintained during the proof. Each
task is reduced to a possibly empty set of subtasks by one of the following proof
construction steps: (1) the introduction of a proof sketch [51][15], (2) deep struc-
tural rules for weakening and decomposition of subformulas, (3) the application of
a lemma that can be postulated on the fly, (4) the substitution of meta-variables,
and (5) the application of a *inferences* inside formulas. *Inferences* are the major
proof construction means and unite the concepts of tactics and methods that were
separate concepts in the previous versions of ΩMEGA. An inference is a proof step
with multiple premises and conclusions augmented by (1) a possibly empty set of
hypotheses for each premise, (2) a set of *application conditions* that must be ful-
filled upon inference application, (3) a set of *outline functions* that can compute
the values of premises and conclusions from values of other premises and conclu-
sions, and (4) an *expansion function* that refines the abstract inference step. Each
premise and conclusion consists of a unique name and a formula scheme. Note that
we employ the term *inference* in its general meaning; Taken in that sense, an in-
ference can be either valid or invalid in contrast to the formal logic notion of an
inference rule.

*Ω*ANTS. The *Ω*ANTS-system was originally developed to support a user in an in-
teractive theorem proving environment by distributively searching via agents for
possible next proof steps [12, 47] and was later extended to a fully automated sys-
tem. Conceptually the system consists of two kinds of agents: *command agents* and
argument agents. Each command agent is associated to a tactic, method or calculus
rule, uniformly called rule, and orders suggestions for the associated rule according

---

[14] "Under-specification" is a technical term borrowed from research on the semantics of
natural language. Illustrating examples and a discussion of our notion can be found in
[2, 9].

[15] In the old ΩMEGA system this was realized by using so-called *Island*-methods.

to some heuristics. These suggestions are generated by a society of argument agents which are assigned to a command agent. These command and argument agents had to be specified manually in the previous version of the ΩMEGA system. In [8] we automated this process by defining a mechanism to automatically create for a given inference a respective command agent together with an optimal set of argument agents.

3.2 Mathematical Knowledge Representation

A mathematical proof assistant relies upon different kinds of knowledge: first, of course, the formalized mathematical domain as organized in structured theories of definitions, lemmata, and theorems. Secondly, there is mathematical knowledge on how to prove a theorem, which is encoded in tactics and methods, in ΩANTS agents, in control knowledge and in strategies. This type of knowledge can be general, theory specific or even problem specific.

The integration of a mathematical proof assistant into the typical and everyday activities of a mathematician requires however other types of knowledge as well. For example, a mathematical tutoring system for students relies upon a database with different samples of proofs and proof plans linked by meta-data in order to advise the student. Another example is the support for mathematical publications: the documents containing both formalized and non-formalized parts need to be related to specific theories, lemmas, theorems, and proofs. This raises the research challenge on how the usual structuring mechanisms for mathematical theories (such as theory hierarchies or the import of theories via renaming or general morphisms) can be extended to tactics and methods as well as to proofs, proof plans and mathematical documents. Furthermore, changing any of these elements requires maintenance support as any change in one part may have consequences in other parts. For example, the validity of a proof needs to be checked again after changing parts of a theory, which in turn may affect the validity of the mathematical documents. This management of change [30, 5, 38] has been implemented in the MAYA system [6], which is now integrated into the ΩMEGA system as well (see Fig. 4). It supports an evolutionary formal development by allowing users to specify and verify developments in a structured manner, incorporates a uniform mechanism for verification in-the-large to exploit the structure of the specification, and maintains the verification work already done when changing the specification. MAYA relies on development graphs [30] as a uniform representation of structured specifications, which enables the use of various (structured) specification languages to formalize the software development. Moreover, MAYA allows the integration of different theorem provers to deal with the actual proof obligations arising from the specification, i.e. to perform verification in-the-small.

Hierarchically structured mathematical knowledge, i.e. an ontology of mathematical theories and assertions has initially been stored in ΩMEGAs hardwired mathematical knowledge base. This mathematical knowledge base was later (end of the 90s) out-sourced and linked to the development of MBASE [27]. We now assume that a mathematical knowledge base also maintains domain specific control rules, strategies, and linguistic knowledge. While this is not directly a subject of
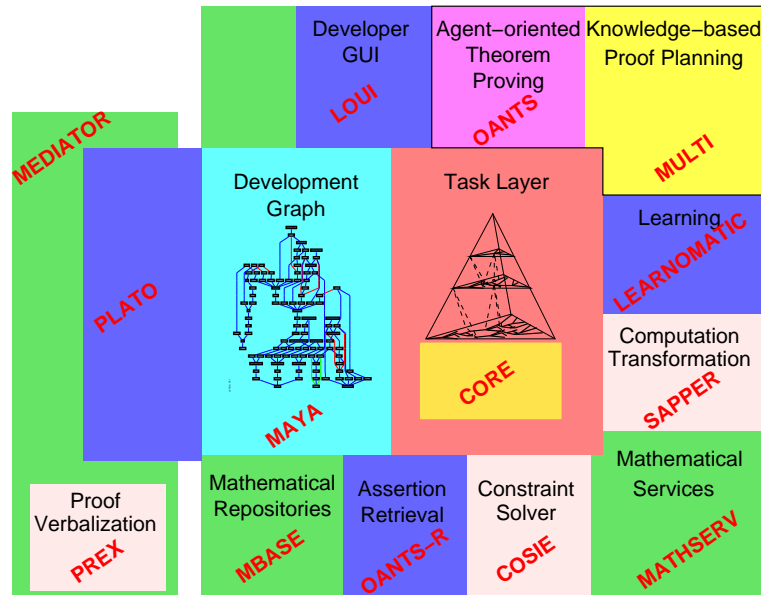
Fig. 4. Architecture of the new version of the mathematical assistance system ΩMEGA.

research in the ΩMEGA project, relying here on other groups of the MKM community and hence on the general development of a worldwide mathematical knowledge base ("the Semantic Web for Mathematicians"), we shall nevertheless concentrate on one aspect, namely how to find the appropriate information.

Semantic Mediators for Mathematical Knowledge Bases. Knowledge acquisition and retrieval in the currently emerging large repositories of formalized mathematical knowledge should not be based purely on syntactic matching, but it needs to be supported by semantic mediators, which suggest applicable theorems and lemmata in a given proof context.

We are working on appropriately limited higher order logic (HOL) reasoning agents for domain- and context-specific retrieval of mathematical knowledge from mathematical knowledge bases. For this we shall adapt a two stage approach as in [10], which combines syntactically oriented pre-filtering with semantic analysis. The pre-filter employ efficiently processable criteria based on meta-data and ontologies that identify sets of candidate theorems from a mathematical knowledge base that are potentially applicable to a focused proof context. The HOL agents act as post-filters to exactly determine the applicable theorems of this set. Exact semantic retrieval includes the following aspects: (i) logical transformations to see the connection between a theorem in a mathematical knowledge base and a focused subgoal. Consider, e.g., a theorem of the form $A \Leftrightarrow B$ in the mathematical knowledge base and a subgoal of the form $(A \Rightarrow B) \wedge (\neg A \Rightarrow \neg B)$; they are not equal in any syntactical sense, but they denote the same assertion. (ii) The variables of a

theorem in a mathematical knowledge base may have to be instantiated with terms occurring in a focused subgoal; consider, e.g., a theorem $\forall X \, . \, is{-}square(X \times X)$ and the subgoal $is{-}square(2 \times 2)$. (iii) Free variables (meta-variables) may occur in a focused subgoal and they may have to be instantiated with terms occurring in a theorem of the mathematical knowledge base; consider, e.g., a subgoal $irrational(X)$ with metavariable X and a theorem $irrational(\sqrt{2})$.

We are investigating whether this approach can be successfully coupled with state-of-the-art search engines such as Google.

Synthesizing Inferences. In the old ΩMEGA-system the knowledge contained in the mathematical theories was not automatically available to the proof planner or the ΩANTS-system. In order to make them available for these systems, they had to be specified manually as methods or tactics and agents. In [8] we developed a mechanism that allows the computation of a set of inferences for arbitrary formulas. As example consider the lemma

$$\forall U, V, W : set.U \subseteq V \wedge V \subseteq W \Rightarrow U \subseteq W \tag{1}$$

The inference obtained from that lemma is

$$\frac{P_1 : U \subseteq V \qquad P_2 : V \subseteq W}{C : U \subseteq W} \tag{2}$$

where U, V, W are meta-variables. The benefit of that approach is that proof procedural information used by the automatic reasoning procedures is directly synthesized from declarative representations of mathematical knowledge. Together with the calculus where the application of inferences is the basic proof construction step, this allows interactive and automatic proof development directly at the assertion level and paves the way to offer proof support inside text editors and in a form that is intuitive for a human user.

3.3   MathServe: A Global Web for Mathematical Services

The Internet provides a vast collection of data and computational resources. For example, a travel booking system combines different information sources, such as the search engines, price computation schemes, and the travel information in distributed very large databases, in order to answer complex booking requests. The access to such specialized travel information sources has to be planned, the obtained results combined and, in addition the consistency of time constraints has to be guaranteed.

In [53, 54] this methodology was transferred and applied to mathematical problem solving and developed a system that plans the combination of several mathematical information sources (such as mathematical databases), computer algebra systems, and reasoning processes (such as theorem provers or constraint solvers). Based on the well-developed MATHWEB-SB network of mathematical services [52], the existing client-server architecture has been extended by advanced problem solving capabilities and semantic brokering of mathematical services.

The reasoning systems integrated in MATHWEB-SB had to be accessed directly via their API, thus the interface to MATHWEB-SB is *system-oriented*. However, these reasoning systems are used also in applications that are not necessarily theorem provers, e.g. for the semantic analysis of natural language, small verification tasks, etc. The main goals of the MATHSERVE framework were therefore:

Problem-Oriented Interface: to develop a more abstract communication level for MATHWEB-SB, such that general mathematical problem descriptions can be sent to MATHSERVE which in turn returns a solution to that problem. Essentially, this goal is to move from a *service* oriented interface of MATHWEB-SB to a *problem* oriented interface for the MATHSERVE.

Advanced Problem Solving Capabilities: Typically, a given problem cannot be solved by a single service but only by a combination of several services. In order to support the automatic selection and combination of existing services, the key idea is as follows: an ontology is used for the qualitative description of MATHWEB-SB services and *these descriptions are then used as AI planning operators*, in analogy to todays proof planning approach. MATHSERVE uses planning techniques [14, 22] to automatically generate a plan that describes how existing services must be combined to solve a given mathematical problem.

3.4 Publishing Tools for Mathematics and Formal Methods

Proof construction is an important but only a small part of a much wider range of mathematical activities an ideal mathematical assistant system should support (see Fig. 1). Therefore the ΩMEGA system is currently extended to support the writing of mathematical publications and documentations of formal methods in software engineering or advising students during proof construction.

The research questions we plan to investigate arise from the following scenario of preparing a mathematical research article with formalized content in a textbook style *and* in professional type-setting quality.

Mathematical Research Article Preparation Scenario: The author starts writing a new mathematical document in a format suitable for publication by using mathematical concepts from different mathematical domains. New mathematical concepts or lemmas introduced in the paper in turn should result in corresponding new formal objects. Furthermore, when writing the document service tools can be used to perform, for instance, intermediate computations in illustrating examples, querying mathematical databases for mathematical publications introducing similar concepts, etc. Proofs of lemmas and theorems contained in the document should be amenable to formal proof checking techniques. Submission of such a document to some journal allows then for instance to check the proofs contained therein to some degree and a long-term goal may be fully automated verification. Publication of such verified mathematical documents then makes the paper and especially its formal contributions accessible to other authors.

The close relationship between the preparation of a mathematical article and the preparation of documents about software is illustrated by the following scenario which consists of the preparation of protection profiles using formal software development techniques.

242

Preparation of Protection Profiles: The vision underlying this scenario is to support the development cycle of protection profiles and security targets for formal software development as required by the Common Criteria [39]. It starts with the preparation of a protection profile document in publishable format and the three separated parts *Threats*, *Objectives*, and *Requirements*. The first part contains the informal description of the security threats and relates to a formal specification of the threats. The second part formulates the security objectives, which specify the desired system behavior together with a *correspondence demonstration* that this excludes the behaviors leading to threatening situations as specified in the first part. Both the threats and the security objectives are related to formal specifications and the correspondence section to a set of formal proofs. Finally, the security requirement specification further refines the security objectives and also corresponds to a formal specification and a proof that the security requirements indeed refine the security objectives.

The obtained final protection profile is a formal document which can be independently checked by certification authorities (such as in Germany the BSI[16] and the TÜV[17]) in order to obtain a formally certified protection profile. A developer of a specific piece of software (and hardware) can query for existing (certified) protection profiles based on the semantic descriptions of the threats and type of software contained in the documents. The protection profile can be refined to obtain a *security target* document for the developed software, and again can be independently checked by, for instance, the customers.

The vision is to enable a document-centric approach to formalizing and verifying mathematics and software, which is a continuation of the approach taken in MIZAR[18] [41, 42]. However, unlike the approach taken in MIZAR, an author shall be able to prepare a document with a standard text preparation system and without having to follow linguistic or structural restrictions. All concepts, conjectures and (partial) proofs formalized in a document should be processable by the proof assistant of her choice. The author shall be able to access the formal logical parts contained in documents from third parties by a simple citation mechanism. In turn the author can grant other persons access to her document either in the proprietary format of her text preparation system or in a semantically annotated portable print format, e.g. PDF. The proof assistant shall provide authoring support inside the text preparation tool: the type of support ranges from simple type checking, via type reconstruction for underspecified parts, to interactive and automatic proof checking and proof construction support. Any of these supports, including proof construction, shall be in a style that is intuitive even if the author is not an expert in formal logic.

As a result this allows the development of mathematical documents in a publishable style which in addition are formally validated by ΩMEGA, hence obtaining *certified mathematical documents*. A first step in that direction is currently under development by linking the WYSIWYG mathematical editor TeX$_{MACS}$ [49] with the ΩMEGA proof assistant and other proof assistant systems. To this end we developed
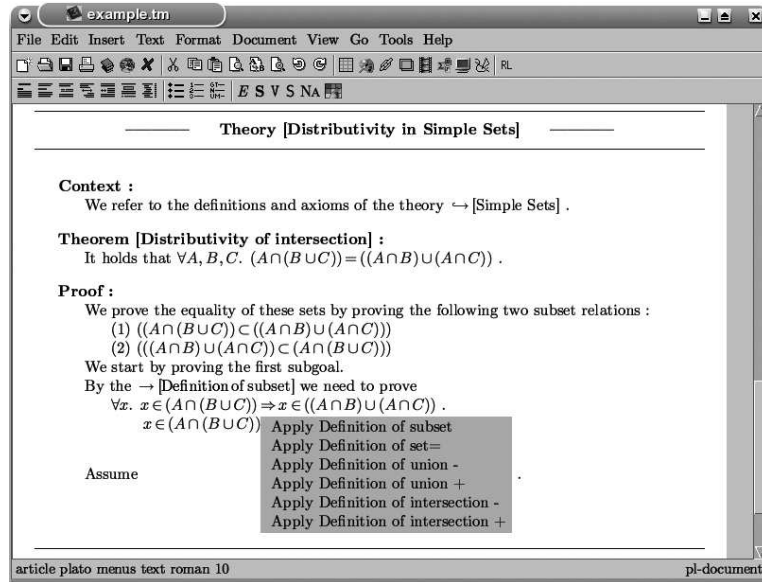
---

[16] www.bsi.de

[17] www.tuev.de

[18] www.mizar.org

Fig. 5. Document in TEX$_{\text{MACS}}$ with an activated menu provided by Ωmega.

the PlatΩ system, that mediates between text-editors and the proof assistants [32, 50] and that maintains the consistency of the representations on either side. Furthermore, it context-sensitively relays service requests from inside the text-editor to the proof assistant and transforms any modifications done by the proof assistant into patches to the document. Fig. 5 presents a document authored in TEX$_{\text{MACS}}$ together with a context-sensitive menu that is provided by Ωmega and displayed inside the text-editor.

Currently, we rely on a semantic markup attached to the different parts of the document, which must be provided manually and contains all information that is relevant to the proof assistant system, e.g. the begin of a theory, the definition of a typed constant, the structure of formulas and the different proof construction steps. The TEX$_{\text{MACS}}$-system provides LATEX-like editing and macro-definition features, and we provide the author a set of predefined macros to annotate her document. Only the semantic markup is handled by PlatΩ which ignores the text itself. This provides a clean interface to the proof assistant systems, that can remain fixed while one can gradually include text analysis tools to automatically create the semantic markup. This allows us to translate new textual definitions and lemmas into the formal representation, as well as to translate (partial) textbook proofs into (partial) proof plans. On the other hand, natural language generation tools can be included to present parts of proofs found by the proof assistant system in natural language inside the document. We have started to use parsers for formulas written in standard LATEX-style syntax in TEX$_{\text{MACS}}$ and especially to allow the user to define her own notation for the introduced syntax. The user-defined syntax is used for parsing as well as for generation of formulas obtained from the proof assistant.

244

3.5 E-Learning Mathematics

We are also involved in the DFKI project ACTIVEMATH, which develops an e-learning tool for tutoring students, in particular in advising a student to develop a proof. Thereby the interaction with the student should be conducted via a textual dialog. This scenario is currently under investigation in the DIALOG project [9] and, aside from all linguistic analysis problems, gives rise to the problem of *under-specification* in proofs. An overview of the ACTIVEMATH project gives [37] and more technical reports are [34, 35, 33, 48].

References

1. S. Autexier. *Hierarchical Contextual Reasoning*. PhD thesis, Computer Science Department, Saarland University, Saarbrücken, Germany, 2003.
2. S. Autexier, C. Benzmüller, A. Fiedler, H. Horacek, and Q. Bao Vo. Assertion-level proof representation with under-specification. *Electronic in Theoretical Computer Science*, 93:5–23, 2003.
3. S. Autexier, C. Benzmüller, and J. Siekmann. Computer supported mathematics with ΩMEGA. 2005.
4. S. Autexier, Chr. Benzmüller, D. Dietrich, A. Meier, and C.-P. Wirth. A generic modular data structure for proof attempts alternating on ideas and granularity. In M. Kohlhase, editor, *Proceedings of MKM'05*, LNAI 3863, IUB Bremen, Germany, January 2006. Springer.
5. S. Autexier and D. Hutter. Maintenance of formal software development by stratified verification. In M. Baaz and A. Voronkov, editors, *Proceedings of LPAR'02*, LNCS, Tbilissi, Georgia, September 2002. Springer.
6. S. Autexier, D. Hutter, T. Mossakowski, and A. Schairer. The development graph manager MAYA. In H. Kirchner and C. Ringeissen, editors, *Proceedings 9th International Conference on Algebraic Methodology And Software Technology (AMAST'02)*, volume 2422 of *LNCS*. Springer, September 2002.
7. Serge Autexier. The CORE calculus. In Robert Nieuwenhius, editor, *Proceedings of the 20th Conference on Automated Deduction (CADE)*, LNAI, Tallinn, Estonia, July 22–27, 2005. Springer.
8. Serge Autexier and Dominik Dietrich. Synthesizing proof planning methods and oants agents from mathematical knowledge. In Jon Borwein and Bill Farmer, editors, *Proceedings of MKM'06*, volume 4108 of *LNAI*, pages 94–109. Springer, August 2006.
9. C. Benzmüller, A. Fiedler, M. Gabsdil, H. Horacek, I. Kruijff-Korbayova, M. Pinkal, J. Siekmann, D. Tsovaltzi, B. Quoc Vo, and M. Wolska. Tutorial dialogs on mathematical proofs. In *Proceedings of IJCAI-03 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, pages 12–22, Acapulco, Mexico, 2003.
10. C. Benzmüller, A. Meier, and V. Sorge. Bridging theorem proving and mathematical knowledge retrieval. In D. Hutter and W. Stephan, editors, *Festschrift in Honour of Jörg Siekmann's 60s Birthday*, LNAI. Springer, 2004.
11. C. Benzmüller and V. Sorge. Ωants – An open approach at combining Interactive and Automated Theorem Proving. In M. Kerber and M. Kohlhase, editors, *8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus-2000)*. AK Peters, 2000.

12. Chr. Benzmüller and V. Sorge. ΩAnts – an open approach at combining interactive and automated theorem proving. In M. Kerber and M. Kohlhase, editors, *Proceedings of Calculemus-2000*, St. Andrews, UK, 6–7 August 2001. AK Peters.

13. B. Buchberger, G. Gonnet, and M. Hazewinkel. Special issue on mathematical knowledge management. *Annals of Mathematics and Artificial Intelligence*, 38(1-3): 3–232, May 2003.

14. J. Carbonell, J. Blythe, O. Etzioni, Y. Gil, R. Joseph, D. Kahn, Craig. Knoblock, S. Minton, M. A. Pérez, S. Reilly, M. Veloso, and X. Wang. PRODIGY 4.0: The Manual and Tutorial. CMU Technical Report CMU-CS-92-150, Carnegie Mellon University, June 1992.

15. L. Cheikhrouhou and V. Sorge. PDS — A Three-Dimensional Data Structure for Proof Plans. In *Proceedings of the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications (ACIDCA'2000)*, Monastir, Tunisia, 22–24 March 2000.

16. A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

17. M. Davis, editor. *The Undecidable: Basic Papers on undecidable Propositions, unsolvable Problems and Computable Functions*. Raven Press Hewlett, New York, 1965.

18. M. Davis. The prehistory and early history of automated deduction. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning*, volume 2 Classical Papers on Computational Logic 1967–1970 of *Symbolic Computation*. Springer, 1983.

19. M. Davis. The early history of automated deduction. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 1, pages 3–15. Elsevier Science, 2001.

20. D. Dietrich. The task-layer of the Ωmega system. Diploma thesis, FR 6.2 Informatik, Universität des Saarlandes, Saarbrücken, Germany, 2006.

21. The Doris system is available at www.cogsci.ed.ac.uk/˜jbos/doris, 2001.

22. K. Erol, J. Hendler, and D. Nau. Semantics for hierarchical task network planning. Technical Report CS-TR-3239, UMIACS-TR-94-31, Computer Science Department, University of Maryland, March 1994.

23. A. Fiedler. Dialog-driven adaptation of explanations of proofs. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1295–1300, Seattle, WA, 2001. Morgan Kaufmann.

24. A. Fiedler. P.rex: An interactive proof explainer. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning — 1st International Joint Conference, IJCAR 2001*, number 2083 in LNAI. Springer, 2001.

25. A. Fiedler. *User-adaptive proof explanation*. PhD thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2001.

26. A. Franke and M. Kohlhase. System description: MBase, an open mathematical knowledge base. In D. McAllester, editor, *Proceedings of the 17th Conference on Automated Deduction*, number 1831 in LNAI. Springer, 2000.

27. Andreas Franke and Michael Kohlhase. System description: Mbase, an open mathematical knowledge base. In David McAllester, editor, *17th International Conference on Automated Deduction CADE-17, Pittsburgh, USA, June 17-20, 2000*, volume 1831 of *Lecture notes in computer science*. Springer, 2000.

28. M. J. Gordon, A. J. Milner, and C. P. Wadsworth. *Edinburgh LCF – A mechanised logic of computation*. Springer Verlag, 1979. LNCS 78.

29. X. Huang. Reconstructing Proofs at the Assertion Level. In A. Bundy, editor, *Proceedings of the 12th Conference on Automated Deduction*, number 814 in LNAI, pages 738–752. Springer, 1994.

30. D. Hutter. Management of change in structured verification. In *Proceedings of Automated Software Engineering, ASE-2000*. IEEE, 2000.

31. M. Kohlhase and A. Franke. MBASE: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation; Special Issue on the Integration of Computer Algebra and Deduction Systems*, 32(4):365–402, September 2001.

32. Christoph Benzmüller, Marc Wagner, Serge Autexier. Plato: A mediator between text-editors and proof assistance systems. In Christoph Benzmüller, Serge Autexier, editor, *7th Workshop on User Interfaces for Theorem Provers (UITP'06)*, ENTCS. Elsevier, August 2006.

33. E. Melis and E. Andrés. Global feedback in ActiveMath. 2003.

34. E. Melis, E. Andrés, J. Bdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education*, 12:385–407, 2001.

35. E. Melis, J. Büdenbender E. Andrés, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. Knowledge representation and management in ActiveMath. *Annals of Mathematics and Artificial Intelligence, Proceedings of the first Conference on Mathematical Knowledge Management MKM'01*, 38:47–64, 2003.

36. E. Melis and A. Meier. Proof planning with multiple strategies. In J. Loyd, V. Dahl, U. Furbach, M. Kerber, K. Lau, C. Palamidessi, L.M. Pereira, Y. Sagivand, and P. Stuckey, editors, *First International Conference on Computational Logic (CL-2000)*, number 1861 in LNAI, pages 644–659, London, UK, 2000. Springer.

37. E. Melis and J. Siekmann. E-learning logic and mathematics: What we have and what we need. In *Festschrift to Dov Gabbay*. Kings College Publication, 2005.

38. Till Mossakowski, Serge Autexier, and Dieter Hutter. Extending development graphs with hiding. *Journal of Logic and Algebraic Programming, special issue on Algebraic Specification and Development Techniques*, 2005.

39. Common Criteria Project Sponsoring Organisations. Common criteria for information technology security evaluation (CC) version 2.1, 1999. Also ISO/IEC 15408: IT – Security techniques – Evaluation criteria for IT security.

40. J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

41. P. Rudnicki. An overview of the MIZAR-project. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*. Chalmers University of Technology, 1992.

42. P. Rudnicki and A. Trybulec. On equivalents of well-foundedness. An experiment in MIZAR. *Journal of Automated Reasoning*, 23:197–234, 1999.

43. J. Siekmann. Geschichte des automatischen beweisens (history of automated deduction). In *Deduktionssysteme, Automatisierung des Logischen Denkens*. R. Oldenbourg Verlag, 2nd edition, 1992. Also in English with Elsewood.

44. J. Siekmann. History of computational logic. In D. Gabbay and J. Woods, editors, *The Handbook of the History of Logic*, volume I-IX. Elsevier, 2005.

45. J. Siekmann and C. Benzmüller. Omega: Computer supported mathematics. In G. Palm S. Biundo, T. Frhwirth, editor, *KI 2004: Advances in Artificial Intelligence: 27th Annual German Conference on AI*, number 3228 in LNAI, pages 3–28, Ulm, Germany, 2004.

46. J. Siekmann, S. Hess, C. Benzmüller, L. Cheikhrouhou, A. Fiedler, H. Horacek, M. Kohlhase, K. Konrad, A. Meier, E. Melis, M. Pollet, and V. Sorge. $\mathcal{LOUI}$: $\mathcal{L}$ovely ΩMEGA $\mathcal{U}$ser $\mathcal{I}$nterface. *Formal Aspects of Computing*, 11:326–342, 1999.

47. V. Sorge. *A Blackboard Architecture for the Integration of Reasoning Techniques into Proof Planning*. PhD thesis, FR 6.2 Informatik, Universität des Saarlandes, Saarbrücken, Germany, November 2001.

48. C. Ullrich, P. Libbrecht, S. Winterstein, and M. Mhlenbrock. A flexible and efficient presentation-architecture for adaptive hypermedia: Description and technical evaluation. pages 21–25. IEEE Computer Society, 2004.
49. J. van der Hoeven. GNU TeXmacs: A free, structured, wysiwyg and technical text editor. In *Actes du congrès Gutenberg*, number 39-40, in Actes du congrès Gutenberg, pages 39–50, Metz, May 2001.
50. Marc Wagner. Mediation between text-editors and proof assistance systems. Diploma thesis, Saarland University, Saarbrücken, Germany, 2006.
51. F. Wiedijk. Formal proof sketches. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani, editors, *Types for Proofs and Programs: Third International Workshop, TYPES 2003*, LNCS 3085, pages 378–393, Torino, Italy, 2004. Springer.
52. J. Zimmer and M. Kohlhase. System description: The Mathweb Software Bus for distributed mathematical reasoning. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in LNAI, pages 138–142. Springer, 2002.
53. Jürgen Zimmer. MATHSERVE – *A Framework for Semantic Reasoning Services*. PhD thesis, FR 6.2 Informatik, Universität des Saarlandes, 2007.
54. Jürgen Zimmer and Serge Autexier. The MATHSERVE framework for semantic reasoning web services. In U. Furbach and N. Shankar, editors, *Proceedings of IJCAR'06*, LNAI, pages 140–144, Seattle, USA, August 2006. Springer.