ΩMEGA: Computer Supported Mathematics

Jörg Siekmann and Christoph Benzmüller

Saarland University, Saarbrücken, Germany,
siekmann@dfki.de, chris@ags.uni-sb.de
http://www-ags.dfki.uni-sb.de/
http://ags.uni-sb.de/~chris

Dedicated to Martin Davis

Abstract. The year 2004 marks 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.

Classical 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, search based as well as proof planning 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, in particular it supports proof development at a human oriented level of abstraction. It is a modular system with a central proof data structure and several supplementary subsystems including automated deduction and computer algebra systems. ΩMEGA has many characteristics in common with systems like NuPrL [ACE+00], CoQ [Coq03], Hol [GM93], Pvs [ORR+96], and Isabelle [Pau94,NPW02]. However, it differs from these systems with respect to its focus on *proof planning* and in that respect it is more similar to the proof planning systems Clam and λClam at Edinburgh [RSG98,BvHHS90].

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

S. Biundo, T. Frühwirth, and G. Palm (Eds.): KI 2004, LNAI 3238, pp. 3–28, 2004.

1 Introduction

The vision of computer-supported mathematics and a system which provides integrated support for all work phases of a mathematician (see Fig. 1) has fascinated researchers in artificial intelligence, particularly in the deduction systems area, and in mathematics for a long time. 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.

Fig. 1. Calculemus illustration of different challenges for a mathematical assistance system.

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 [Rob65], 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[2]. It is only very recently that this trend is reversed, with the CALCULEMUS[3] and MKM[4] 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 in the new millennium by a complementary top-down approach starting from existing, mainly pen and paper based mathematical practice down to system support.

We shall provide an overview and the main developments of the ΩMEGA project in the following and then point to current research and some future goals.

2 ΩMEGA

The ΩMEGA project 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 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 [Chu40]. 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* (see Section 2.3). Proof construction is also supported by already proven assertions and theorems and by calls to external systems to simplify or solve subproblems.

2.1 System Overview

At the core of ΩMEGA is the *proof plan data structure* $\mathcal{PDS}$ [CS00], in which proofs and *proof plans* are represented at various levels of granularity and abstraction (see 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

[2] The history of the field is presented in a classical paper by Martin Davis [Dav83] and also in [Dav01] and more generally in his history of the making of the first computers [Dav65]. Another source is Jörg Siekmann [Sie92] and more recently [Sie04].

[3] www.calculemus.org

[4] monet.nag.co.uk/mkm/index.html

and classified with respect to a taxonomy of mathematical theories in the mathematical knowledge base MBase [FK00a,KF01]. The user of ΩMEGA, or the proof planner Multi [MM00], or else the suggestion mechanism ΩAnts [BS00] 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.

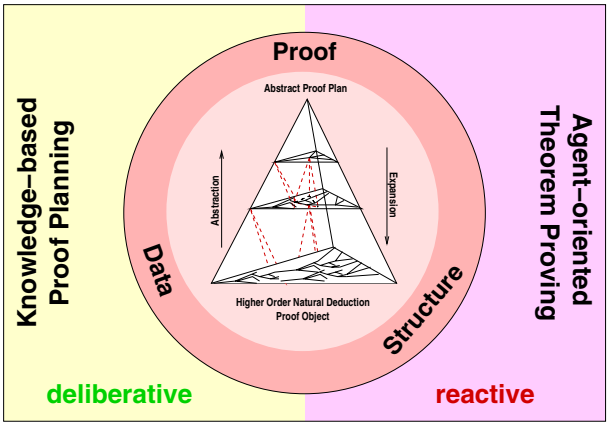

Fig. 2. The proof plan datastructure PDS is at the core of the ΩMEGA system. Proof construction is facilitated by knowledge-based proof planning (deliberative), agent-oriented theorem proving (reactive), or by user interaction.

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 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 $\mathcal{LOUI}$ [SHB+99] (see Fig. 4) provides both a graphical and a tabular view of the proof under consideration, and the interactive proof explanation system *P.rex* [Fie01b,Fie01a,Fie01c] 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 [ZK02]. 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 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 [Dor01]), proof planning and verification tasks.

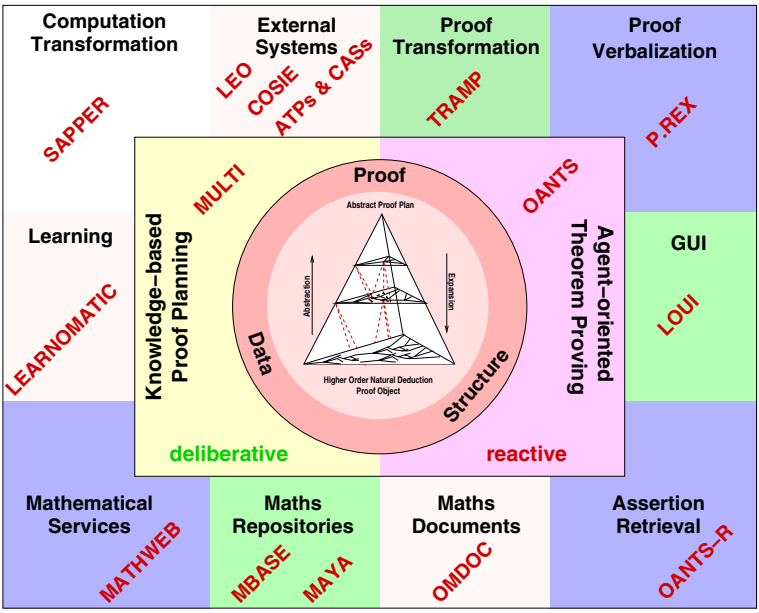

Fig. 3. The vision of an all encompassing mathematical assistance environment: we have now modularized and out-sourced many of the support tools such that they can also be used by other systems via the MathWeb-SB software bus.

2.2 External Systems

Proof problems require many different skills for their solution. Therefore, it is desirable to have access to several systems with complementary capabilities, to orchestrate their use, and to integrate their results. ΩMEGA interfaces heterogeneous external systems such as *computer algebra systems (CASs)*, higher and first order *automated theorem proving systems (ATPs)*, *constraint solvers (CSs)*, and *model generation systems (MGs)*.

Their use is twofold: they may provide a solution to a subproblem, or they may give hints for the control of the search for a proof. In the former case, the output of an incorporated reasoning system is translated and inserted as a subproof into the $\mathcal{PDS}$. This is beneficial for interfacing systems that operate at different levels of abstraction, and also for a human-oriented display and inspection of a partial proof. Importantly, it also enables us to check the soundness of each contribution by expanding the inserted subproof to a logic-level proof and then verify it by ΩMEGA's proof checker.

Currently, the following external systems are integrated in ΩMEGA:

CASs provide symbolic computation, which can be used in two ways: first, to compute hints to guide the proof search (e.g., witnesses for existential variables), and, second, to perform some complex algebraic computation such as to normalize or simplify terms. In the latter case the symbolic computation is directly translated into proof steps in ΩMEGA. CASs are integrated via the transformation and translation module SAPPER [Sor00]. Currently, ΩMEGA uses the systems MAPLE [CGG+92] and GAP [S+95].

ATPs are employed to solve subgoals. Currently ΩMEGA uses the first order provers BLIKSEM [dN99], EQP [McC97], OTTER [McC94], PROTEIN [BF94], SPASS [WAB+99], WALDMEISTER [HJL99], the higher order systems TPS [ABI+96], and $\mathcal{LEO}$ [BK98,Ben99], and we plan to incorporate VAMPIRE [RV01]. The first order ATPs are connected via TRAMP [Mei00], which is a proof transformation system that transforms resolution-style proofs into assertion-level ND proofs to be integrated into ΩMEGA's $\mathcal{PDS}$. TPS already provides ND proofs, which can be further processed and checked with little transformational effort [BBS99].

MGs provide either witnesses for free (existential) variables, or counter-models, which show that some subgoal is not a theorem. Hence, they help to guide the proof search. Currently, ΩMEGA uses the model generators SATCHMO [MB88] and SEM [ZZ95].

CSs construct mathematical objects with theory-specific properties as witnesses for free (existential) variables. Moreover, a constraint solver can help to reduce the proof search by checking for inconsistencies of constraints. Currently, ΩMEGA employs $\mathcal{CoSIE}$ [MZM00], a constraint solver for inequalities and equations over the field of real numbers.

2.3 Proof Planning

ΩMEGA's main focus is on knowledge-based proof planning [Bun88,Bun91], [MS99], where proofs are not conceived in terms of low-level calculus rules, but at a much higher level of abstraction that highlights the main ideas and de-emphasizes minor logical or mathematical manipulations on formulae.

Knowledge-based proof planning is a new paradigm in automated theorem proving, which swings the motivational pendulum back to its AI origins in that it employs and further develops many AI principles and techniques such as hierarchical planning, knowledge representation in frames and control rules, constraint solving, tactical theorem proving, and meta-level reasoning. It differs from traditional search-based techniques in automated theorem proving not least in its level of abstraction: the proof of a theorem is planned at an abstract level where an outline of the proof is found. This outline, that is, the abstract proof plan, can be recursively expanded to construct a proof within a logical calculus provided the proof plan does not fail. The plan operators represent mathematical techniques familiar to a working mathematician. While the knowledge of such a mathematical domain as represented within methods and control rules is specific to the mathematical field, the representational techniques and reasoning procedures

are general-purpose. For example, one of our first case studies [MS99] used the limit theorems proposed by Woody Bledsoe [Ble90] as a challenge to automated reasoning systems. The general-purpose planner makes use of this mathematical domain knowledge and of the guidance provided by declaratively represented control rules, which correspond to mathematical intuition about how to prove a theorem in a particular situation. These rules provide a basis for meta-level reasoning and goal-directed behavior.

Domain knowledge is encoded into methods, control rules, and strategies. Moreover, methods and control rules can employ external systems (e.g., a computer algebra system) and make use of the knowledge in these systems. ΩMEGA's multi-strategy proof planner MULTI [MM00] searches then for a plan using the acquired methods and strategies guided by the control knowledge in the control rules.

2.3.1 AI Principles in Proof Planning. A *planning problem* is a formal description of an *initial state*, a *goal*, and some *operators* that can be used to transform the initial state via some intermediate states to a state that satisfies the goal. Applied to a planning problem, a *planner* returns a sequence of *actions*, that is, instantiated operators, which reach a goal state from the initial state when executed. Such a sequence of actions is also called a *solution plan*.

Proof planning considers mathematical theorems as planning problems [Bun88]. The initial state of a proof planning problem consists of the proof *assumptions* of the theorem, whereas the goal is the *theorem* itself. The operators in proof planning are the methods.

In ΩMEGA, proof planning is the process that computes actions, that is, instantiations of methods, and assembles them in order to derive a theorem from a set of assumptions. The effects and the preconditions of an action in proof planning are proof nodes with formulae in the higher order language $\mathcal{POST}$, where the effects are considered as logically inferable from the preconditions. A proof plan under construction is represented in the proof plan data structure $\mathcal{PDS}$ (see Section 2.5). Initially, the $\mathcal{PDS}$ consists of an open node containing the statement to be proved, and closed, that is, justified, nodes for the proof assumptions. The introduction of an action changes the $\mathcal{PDS}$ by adding new proof nodes and justifying the effects of the action by applications of the method of the action to its premises. The aim of the proof planning process is to reach a *closed* $\mathcal{PDS}$, that is, a $\mathcal{PDS}$ without open nodes. The *solution proof plan* produced is then a record of the sequence of actions that lead to a closed $\mathcal{PDS}$.

By allowing for forward and backward actions ΩMEGA's proof planning combines forward and backward state-space planning. Thus, a *planning state* is a pair of the current world state and the current goal state. The initial world state consists of the given proof assumptions and is transfered by forward actions into a new world state. The goal state consists of the initial open node and is transfered by backward actions into a new goal state containing new open nodes. From this point of view the aim of proof planning is to compute a sequence of actions that derives a current world state in which all the goals in the current goal state are satisfied.

As opposed to precondition achievement planning (e.g., see [Wel94]), effects of methods in proof planning do not cancel each other. For instance, an action with effect $\neg F$ introduced for the open node L_1 does not threaten the effect F introduced by another action for the open node L_2. Dependencies among open nodes result from shared variables for witness terms and their constraints. Constraints can be, for instance, instantiations for the variables but they can also be mathematical constraints such as $x < c$, which states that, whatever the instantiation for x is, it has to be smaller than c. The constraints created during the proof planning process are collected in a constraint store. An action introducing new constraints is applicable only if its constraints are consistent with the constraints collected so far. Dependencies among goals with shared variables are difficult to analyze and can cause various kinds of failures in a proof planning attempt. First results about how to analyze and deal with such failures are discussed in [Mei03].

Methods, Control Rules, and Strategies. *Methods* are traditionally perceived as tactics in tactical theorem proving augmented with preconditions and effects, called *premises* and *conclusions*, respectively. A method represents the inference of the conclusion from the premises. For instance, NotI-M is a method whose purpose is to prove a goal $\Gamma \vdash \neg P$ by contradiction. If NotI-M is applied to a goal $\Gamma \vdash \neg P$ then it closes this goal and introduces the new goal to prove falsity, $\bot$, under the assumption P, that is, $\Gamma, P \vdash \bot$. Thereby, $\Gamma \vdash \neg P$ is the conclusion of the method, whereas $\Gamma, P \vdash \bot$ is the premise of the method. NotI-M is a *backward* method, which reduces a goal (the conclusion) to new goals (the premises). *Forward* methods, in contrast, derive new conclusions from given premises. For instance, =Subst-m performs equality substitutions by deriving from two premises $\Gamma \vdash P[a]$ and $\Gamma \vdash a = b$ the conclusion $\Gamma \vdash P[b]$ where an occurrence of a is replaced by an occurrence of b. Note that NotI-M and =Subst-m are simple examples of domain-independent, logic-related methods, which are needed in addition to domain-specific, mathematically motivated methods. Knowledge base proof planning expands on these ideas and allows for more general mathematical methods to be encapsulated into *methods*.

Control rules represent mathematical knowledge about how to proceed in the proof planning process. They can influence the planner's behavior at choice points (e.g., which goal to tackle next or which method to apply next) by preferring members of the corresponding list of alternatives (e.g., the list of possible goals or the list of possible methods). This way promising search paths are preferred and the search space can be pruned.

Strategies employ different sets of methods and control rules and, thus, tackle the same problem in different ways. The reasoning as to which strategy to employ on a problem is an explicit choice point in MULTI. In particular, MULTI can backtrack from chosen strategies and search at the level of strategies.

Detailed discussions of ΩMEGA's method and control rule language can be found in [Mei03,MMP02]. A detailed introduction to proof planning with multiple strategies is given in [MM00].

2.4 Interface and System Support

ΩMEGA's graphical user interface $\mathcal{L}\Omega\mathcal{UI}$ [SHB+99] displays the current proof state in multiple modalities: a graphical map of the proof tree, a linearized presentation of the proof nodes with their formulae and justifications, a term browser, and a natural language presentation of the proof via *P.rex* (see Fig. 4 and 5).

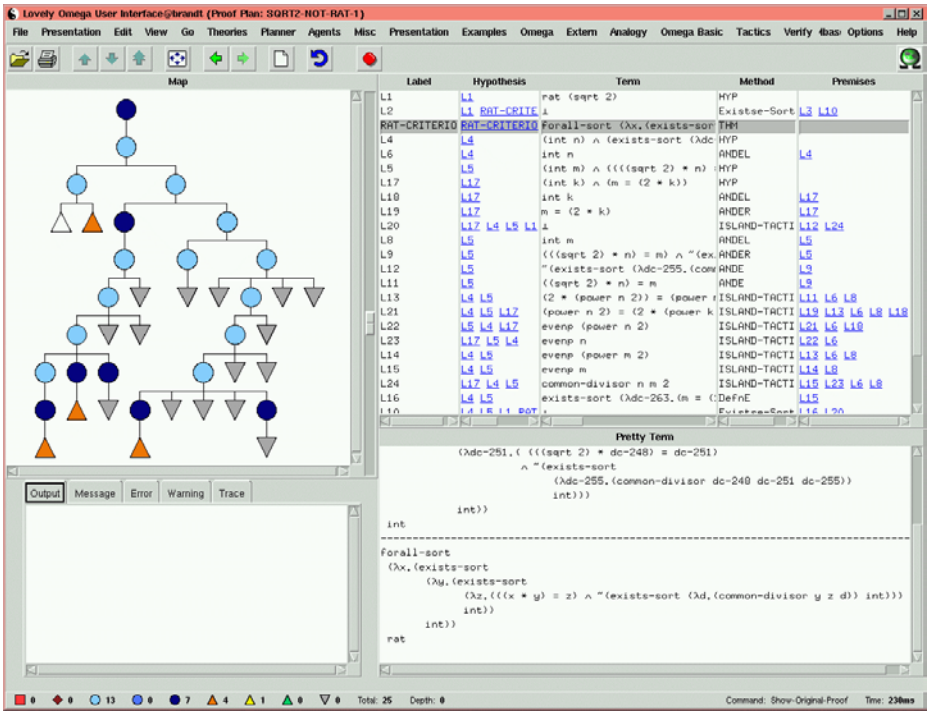

Fig. 4. Multi-modal proof presentation in the graphical user interface $\mathcal{L}\Omega\mathcal{UI}$.

When inspecting a part a proof, the user can switch between alternative levels of abstraction, for example, by expanding a node in the graphical map of the proof tree, which causes appropriate changes in the other presentation modes. Moreover, an interactive natural language *explanation* of the proof is provided by the system *P.rex* [Fie01b,Fie01a,Fie01c], which is adaptive in the following sense: it explains a proof step at the most abstract level (which the user is assumed to know) and then reacts flexibly to questions and requests, possibly at a lower level of abstraction, for example, by detailing some ill-understood subproof.

Another system support is the guidance mechanism provided by the suggestion module ΩANTS [BS98,BS99,BS00,Sor01], which searches pro-actively for possible actions that may be helpful in finding a proof and orders them in a preference list. Examples for such actions are an application of a particular cal-

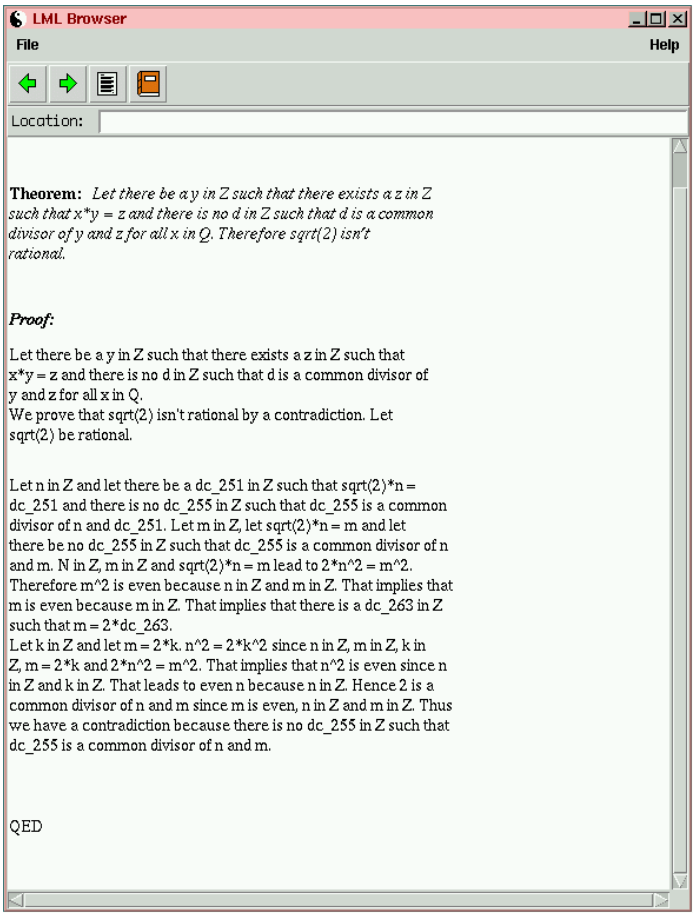

Fig. 5. Natural language proof presentation by *P.rex* in $\mathcal{LOUI}$.

culus rule, the call of a tactic or a proof method as well as a call of an external reasoning system, or the search for and insertion of facts from the knowledge base MBase. The general idea is the following: every inference rule, tactic, method or external system is "agentified" in the sense that every possible *action* searches concurrently for the fulfillment of its application conditions and once these are satisfied it suggests its execution. User-definable heuristics select and display the suggestions to the user. ΩAnts is based on a hierarchical blackboard, which collects the data about the current proof state.

2.5 Proof Objects

The central data structure for the overall search is the proof plan data structure $\mathcal{PDS}$ in Fig. 2. This is a hierarchical data structure that represents a (partial) proof at different levels of abstraction (called partial proof plans). Technically,

it is an acyclic graph, where the nodes are justified by tactic applications. Conceptually, each such justification represents a proof plan (the expansion of the justification) at a lower level of abstraction, which is computed when the tactic is executed. In ΩMEGA, we explicitly keep the original proof plan as well as intermediate expansion layers in an expansion hierarchy. The coexistence of several abstraction levels and the dynamical maintenance of their relationship is a central design objective of ΩMEGA's $\mathcal{PDS}$. Thus the $\mathcal{PDS}$ makes the hierarchical structure of proof plans explicit and retains it for further applications such as proof explanation with *P.rex* or an analogical transfer of plans. The lowest level of abstraction of a $\mathcal{PDS}$ is the ND calculus.

The proof object generated by ΩMEGA for example for the "irrationality of $\sqrt{2}$" theorem is recorded in a technical report [BFMP02], where the unexpanded and the expanded proof objects are presented in great detail, that is in a little less than a thousand proof steps. A general presentation of this interesting case study is [SBF+03].

2.6 Case Studies

Early developments of proof planning in Alan Bundy's group at Edinburgh used proofs by induction as their favorite case studies [Bun88]. The ΩMEGA system has been used in several other case studies, which illustrate in particular the interplay of the various components, such as proof planning supported by heterogeneous external reasoning systems.

A typical example for a class of problems that cannot be solved by traditional automated theorem provers is the class of ϵ–δ–proofs [MS99,Mel98a]. This class was originally proposed by Woody Bledsoe [Ble90] and it comprises theorems such as LIM+ and LIM*, where LIM+ states that the limit of the sum of two functions equals the sum of their limits and LIM* makes the corresponding statement for multiplication. The difficulty of this domain arises from the need for arithmetic computation in order to find a suitable instantiation of free (existential) variables (such as a δ depending on an ϵ). Crucial for the success of ΩMEGA's proof planning is the integration of suitable experts for these tasks: the arithmetic computation is done by the computer algebra system MAPLE, and an appropriate instantiation for δ is computed by the constraint solver $\mathcal{CoSIE}$. We have been able to solve all challenge problems suggested by Bledsoe and many more theorems in this class taken from a standard textbook on real analysis [BS82].

Another class of problems we tackled with proof planning is concerned with residue classes [MPS02,MPS01]. In this domain we show theorems such as: "the residue class structure $(\mathbb{Z}_5, \bar{+})$ is associative", "it has a unit element", and similar properties, where $\mathbb{Z}_5$ is the set of all congruence classes modulo 5 $\{\bar{0}_5, \bar{1}_5, \bar{2}_5, \bar{3}_5, \bar{4}_5\}$ and $\bar{+}$ is the addition on residue classes. We have also investigated whether two given structures are isomorphic or not and altogether we have proved more than 10,000 theorems of this kind (see [Sor01]). Although the problems in this domain are still within the range of difficulty a traditional automated theorem prover can handle, it was nevertheless an interesting case study

for proof planning, since multi-strategy proof planning generated substantially different proofs based on entirely different proof ideas.

Another important proof technique is Cantor's diagonalization technique and we also developed methods and strategies for this class [CS98]. Important theorems we have been able to prove are the undecidability of the halting problem and Cantor's theorem (cardinality of the set of subsets), the non-countability of the reals in the interval $[0, 1]$ and of the set of total functions, and similar theorems.

Finally, a good candidate for a standard proof technique are completeness proofs for refinements of resolution, where the theorem is usually first shown at the ground level using the excess-literal-number technique and then ground completeness is lifted to the predicate calculus. We have done this for many refinements of resolution with ΩMEGA [Geb99].

However, ΩMEGA's main aim is to become a proof assistant tool for the working mathematician. Hence, it should support interactive proof development at a user-friendly level of abstraction. The mathematical theorem that $\sqrt{2}$ is not rational, and its well-known proof dating back to the School of Pythagoras, provides an excellent challenge to evaluate whether this ambitious goal has been reached. In [Wie02] fifteen systems that have solved the $\sqrt{2}$-problem show their results. The protocols of their respective sessions have been compared on a multi-dimensional scale in order to assess the "naturalness" by which real mathematical problems of this kind can be proved within the respective system.

This represents an important shift of emphasis in the field of automated deduction away from the somehow artificial problems of the past – as represented, for example, in the test set of the TPTP library [SSY94] – back to real mathematical challenges.

We participated in this case study essentially with three different contributions. Our initial contribution was an interactive proof in ΩMEGA without adding special domain knowledge to the system. For further details on this case study, which particularly demonstrates the use of ΩMEGA as a tactical theorem prover, we refer to [BFMP02]. The most important albeit not entirely new lesson to be learned from this experiment is that the level of abstraction common in most automated and tactical theorem proving environments is far too low. While our proof representation is already an abstraction (called the *assertion level* in [Hua94]) from the calculus level typical for most ATPs, it is nevertheless clear that as long as a system does not hide all these excruciating details, no working mathematician will feel inclined to use such a system. In fact, this is in our opinion one of the critical impediments for using first order ATPs and one, albeit not the only one, of the reasons why they are not used as widely as, say, computer algebra systems.

This is the crucial issue of the ΩMEGA project and our main motivation for departing from the classical paradigm of automated theorem proving about fifteen years ago.

Our second contribution to the case study of the $\sqrt{2}$-problem is based on interactive *island planning* [Mel96], a technique that expects an outline of the

proof, i.e. the user provides main subgoals, called *islands*, together with their assumptions. The details of the proof, eventually down to the logic level, are postponed. Hence, the user can write down *his* proof idea in a natural way with as many gaps as there are open at this first stage of the proof. Closing the gaps is ideally fully automatic, in particular, by exploiting external systems. However, for difficult theorems it is necessary more often than not that the user provides additional information and applies the island approach recursively.

In comparison to our first tactic-based solution the island style supports a much more abstract and user-friendly interaction level. The proofs are now at a level of abstraction similar to proofs in mathematical textbooks.

Our third contribution to the case study of the $\sqrt{2}$-problem is a fully automatically planned and expanded proof of the theorem. The details of this very important case study, that shows best what (and what cannot) be achieved with current technology are presented in [SBF+03], [SBF+02], and [BFMP02].

The most important question to ask is: Can we find the essential and creative steps automatically? The answer is yes, as we have shown in [SBF+03]. However, while we can answer the question in the affirmative, not every reader may be convinced, as our solution touches upon a subtle point, which opens the Pandora Box of critical issues in the paradigm of proof planning [Bun02]: It is always easy to write some specific methods, which perform just the steps in the interactively found proof and then call the proof planner MULTI to fit the methods together into a proof plan for the given problem. This, of course, shows nothing of substance: Just as we could write down all the definitions and theorems required for the problem in first order predicate logic and hand them to a first order prover[5], we would just hand-code the final solution into appropriate methods.

Instead, the goal of the game is to find *general* methods for a whole class of theorems within some theory that can solve not only this particular problem, but also all the other theorems in that class. While our approach essentially follows the proof idea of the interactively constructed proof for the $\sqrt{2}$-problem, it relies essentially on more general concepts such that we can solve, for example, $\sqrt[j]{l}$-problems for arbitrary natural numbers j and l.

However, this is certainly not the end of the story; in order to evaluate the appropriateness of a proof planning approach we suggest the following three criteria:

(1) How general and how rich in mathematical content are the methods and control rules?
(2) How much search is involved in the proof planning process?
(3) What kind of proof plans, that is, what kind of proofs, can we find?

These criteria should allow us to judge how general and how robust our solution is. The art of proof planning is to acquire domain knowledge that, on the one hand, comprises meaningful mathematical techniques and powerful heuristic guidance, and, on the other hand, is general enough to tackle a broad

[5] This was done when OTTER tackled the $\sqrt{2}$-problem; see [Wie02] for the original OTTER case study and [BFMP02] for its replay with ΩMEGA.

class of problems. For instance, as one extreme, we could have methods that encode ΩMEGA's ND calculus and we could run MULTI without any control. This approach would certainly be very general, but MULTI would fail to prove any interesting problems. As the other extreme case, we could cut a known proof into pieces, and code the pieces as methods. Guided by control rules that always pick the next right piece of the proof, MULTI would assemble the methods again to the original proof without performing any search.

The amount of search and the variety of potential proof plans for a given problem are measures for the generality of the methods and also for the appropriateness for tackling the class of problems by planning. If tight control rules or highly specific methods restrict the search to just one branch in the search tree, then the resulting proof plans will merely instantiate a pattern. In this case, a single tactic or method that realizes the proof steps of the underlying pattern is more suitable than planning. The possibility of creating a variety of proof plans with the given methods and control rules is thus an important feature.

What general lessons can we learn from small, albeit typical mathematical challenges of this kind?

1. The devil is in the detail, that is, it is always possible to hide the crucial creative step (represented as method or represented in the object language by an appropriate lemma) and to pretend a level of generality that has not actually been achieved. To evaluate a solution *all* tactics, methods, theorems, lemmata and definitions have to be made explicit.
2. The enormous distance between the well-known (top-level) proof of the Pythagorean School, which consists of about a dozen single proof steps in comparison to the final (non-optimized) proof at the ND level with 753 inference steps is striking. This is, of course, not a new insight. While mathematics can *in principle* be reduced to purely formal logic-level reasoning as demonstrated by Russell and Whitehead as well as the Hilbert School, nobody would actually want to do so *in practice* as the influential Bourbaki group showed: only the first quarter of the first volume in the several dozen volume set on the foundation of mathematics starts with elementary, logic-level reasoning and then proceeds with the crucial sentence [Bou68]: "No great experience is necessary to perceive that such a project [of complete formalization] is absolutely unrealizable: the tiniest proof at the beginning of the theory of sets would already require several hundreds of signs for its complete formalization."
3. Finally and more to the general point of interest in mathematical support systems: Now that we can prove theorems in the $\sqrt[3]{l}$-problem class, the skeptical reader may still ask: *So what?* Will this ever lead to a *general* system for mathematical assistance?
   We have shown that the class of ϵ-δ-proofs for limit theorems can indeed be solved with a few dozen mathematically meaningful methods and control rules (see [MS99,Mel98b,Mei03]). Similarly, the domain of group theory with its class of residue theorems can be formalized with even fewer methods

(see [MS00,MPS01,MPS02])[6]. An interesting observation is also that these methods by and large correspond to the kind of mathematical knowledge a freshman would have to learn to master this level of professionalism.

Do the above observations now hold for our $\sqrt[j]{l}$-problems? The unfortunate answer is probably *No!* Imagine the subcommittee of the United Nations in charge of the maintenance of the global mathematical knowledge base in a hundred years from now. Would they accept the entry of our methods, tactics and control rules for the $\sqrt[j]{l}$-problems? Probably not!

Factual mathematical knowledge is preserved in books and monographs, *but the art of doing mathematics* [Pol73,Had44] is passed on by word of mouth from generation to generation. The methods and control rules of the proof planner correspond to important mathematical techniques and "ways to solve it", and they make this implicit and informal mathematical knowledge explicit and formal.

The theorems about $\sqrt[j]{l}$-problems are shown by contradiction, that is, the planner derives a contradiction from the equation $l \cdot n^j = m^j$, where n and m are integers with no common divisor. However, these problems belong to the more general class to determine whether two complex mathematical objects $\mathcal{X}$ and $\mathcal{Y}$ are equal. A general mathematical principle for comparison of two complex objects is to look at their characteristic properties, for example, their normal forms or some other uniform notation in the respective theory.

And this is the crux of the matter: to find general mathematical principles and encode them into appropriate methods, control rules and strategies such that an appropriately large *class of problems* can be solved with these methods.

We are now working on formalizing these methods in more general terms and then instantiate them with appropriate parameters to the domain in question (number theory, set theory, or polynomial rings) – and the crucial creative step of the system MULTI is then to find the instantiation by some general heuristics.

3 The Future: What Next?

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, [BGH03]) initiatives, the EU projects MONET[9], OPENMATH and MOWGLI[10], and the American QPQ[11] repository of deductive software tools.

[6] The generally important observation is not, of course, whether we need a dozen or a hundred methods, but that we don't need a few thousand or a million. A few dozen methods seem to be generally enough for a restricted mathematical domain.

[7] www.calculemus.org

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

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

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

[11] www.qpq.org

Furthermore 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.

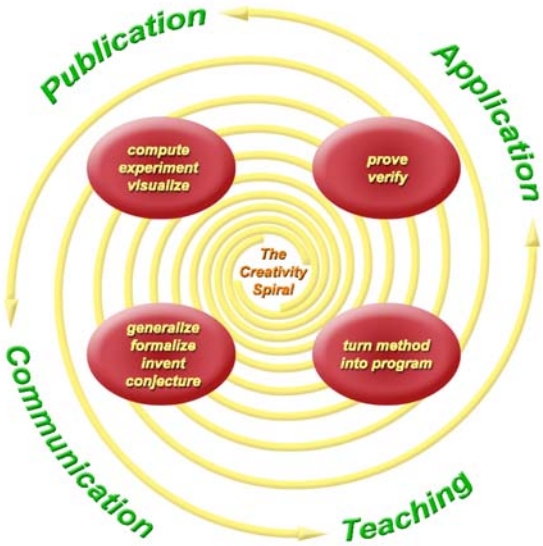

Fig. 6. Mathematical Creativity Spiral; [Buchberger, 1995].

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. 6) 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?

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 typically developed at an argumentative level. This level has been formally categorized as *proofs at the assertion level* [Hua94] with different types of *under-specification* [ABF+03][12]. The CoRe system [Aut03] has been designed to achieve this and the goal is now to completely exchange the current natural deduction calculus by the CoRe calculus.

The proposed exchange of the logic layer in ΩMEGA requires the adaptation of all reasoning procedures that are currently tailored to it, including proof planning and the integration of external systems.

3.2 ΩANTS: Agent-Oriented Theorem Proving

Our agent-based suggestion and reasoning mechanism is called ΩANTS [BS00], whose initial motivation is to turn the hitherto passive ΩMEGA system into a pro-active counter-player of the user which autonomously exploits available resources. It provides societies of pro-active agents organized via an hierarchical blackboard architecture that dynamically and concurrently generate suggestions on applicable proof operators. These ΩANTS agents may also call external systems or perform search for data in mathematical knowledge bases (see [BMS04]).

We will now provide improved higher order theorem proving agents based on the provers $\mathcal{LEO}$ [BK98] and TPS [ABB00], which analyze the proof context and determine promising "control settings". These higher order proof agents will work in competition with traditional first order proof agents and other "agentified" reasoning systems.

3.3 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

[12] "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 [ABF+03,BFG+03].

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 [AHMS02,AM02,AH02,Hut00,MAH01], originally developed for evolutionary formal software engineering at the DFKI, will now be integrated into the ΩMEGA system as well.

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 [FK00b]. 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 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 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 [BMS04], 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 of a mathematical knowledge bases 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.

3.4 VerMath: 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.

We want to transfer and apply this methodology to mathematical problem solving and develop 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, the existing client-server architecture will be extended by advanced problem solving capabilities and semantic brokering of mathematical services.

The reasoning systems currently integrated in MathWeb-SB have 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 semantical analysis of natural language, small verification tasks, etc. The main goal of this project[13] is therefore twofold:

Problem-Oriented Interface: to develop a more abstract communication level for MathWeb-SB, such that general mathematical problem descriptions can be sent to the MathWeb-SB which in turn returns a solution to that problem. Essentially, this goal is to move from a *service* oriented interface to a *problem* oriented interface for the MathWeb-SB. This is a very old idea in the development of AI programming languages (early work included Planner and other languages driven by matching of general descriptions).

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 will be used for the qualitative de-

[13] This is a joint project between the University of Saarbrücken (Jörg Siekmann) and the International University Bremen (Michael Kohlhase).

scription of MathWeb-SB services and *these descriptions will then be used as AI planning operators*, in analogy to todays proof planning approach. We can then use planning techniques [CBE+92,EHN94] to automatically generate a plan that describes how existing services must be combined to solve a given mathematical problem.

3.5 Publishing Tools for Mathematics

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 advising students during proof construction.

With respect to the former we envision that a mathematician writes a new paper in some specific mathematical domain using a LaTeX-like environment. The definitions, lemmas, theorems and especially their proofs give rise to extensions of the original theory and the writing of some proof goes along with an interactive proof construction in ΩMEGA. 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 TeXmacs [vdH01] with the ΩMEGA proof assistant and other mathematical support services (see Fig. 7)

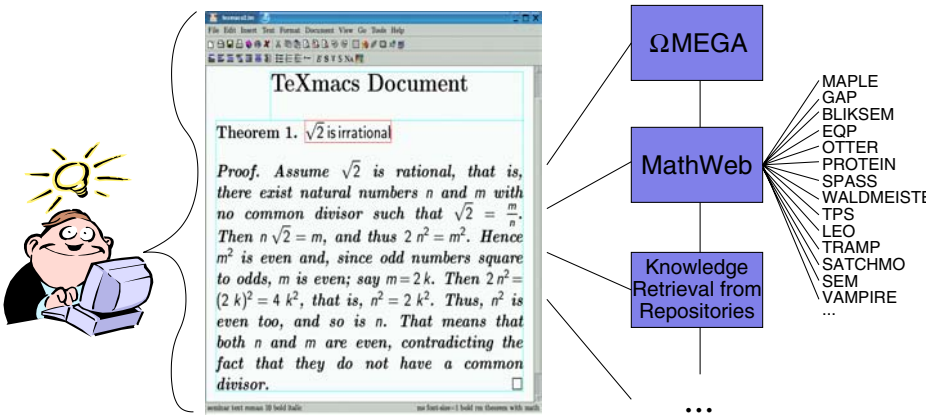

Fig. 7. Semantical documents in TeXmacs: The user will be supported in by different dynamic mathematical reasoning services that "understand" the document content.

The TeXmacs-system provides LaTeX-like editing and macro-definition features, and we are defining macros for theory-specific knowledge such as types, constants, axioms, and lemmata. 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.

As a second activity we are 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 [BFG+03] and, aside from all linguistic analysis problems, gives rise to the problem of *under-specification* in proofs.

References

[ABB00] P.B. Andrews, M. Bishop, and C.E. Brown. System description: TPS: A theorem proving system for type theory. In *Conference on Automated Deduction*, pages 164–169, 2000.

[ABF+03] 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.

[ABI+96] P.B. Andrews, M. Bishop, S. Issar, D. Nesmith, F. Pfenning, and H. Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[ACE+00] S. Allen, R. Constable, R. Eaton, C. Kreitz, and L. Lorigo. The Nuprl open logical environment. In McAllester [McA00].

[AH02] 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.

[AHMS02] 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.

[AM02] S. Autexier and T. Mossakowski. Integrating HOL-CASL into the development graph manager MAYA. In A. Armando, editor, *Proceedings of FROCOS'02*, volume 2309 of *LNAI*, pages 2–17. Springer, April 2002.

[Aut03] S. Autexier. *Hierarchical Contextual Reasoning*. PhD thesis, Computer Science Department, Saarland University, Saarbrücken, Germany, 2003. forthcoming.

[BBS99] C. Benzmüller, M. Bishop, and V. Sorge. Integrating TPS and ΩMEGA. *Journal of Universal Computer Science*, 5:188–207, 1999.

[Ben99] C. Benzmüller. *Equality and Extensionality in Higher-Order Theorem Proving*. PhD thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 1999.

[BF94] P. Baumgartner and U. Furbach. PROTEIN, a PROver with a Theory INterface. In Bundy [Bun94], pages 769–773.

[BFG+03] 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.

[BFMP02] C. Benzmüller, A. Fiedler, A. Meier, and M. Pollet. Irrationality of $\sqrt{2}$ – a case study in ΩMEGA. Seki-Report SR-02-03, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2002.

[BGH03] 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.

[BK98] C. Benzmüller and M. Kohlhase. LEO – a higher-order theorem prover. In Kirchner and Kirchner [KK98].

[Ble90] W. Bledsoe. Challenge problems in elementary calculus. *Journal of Automated Reasoning*, 6:341–359, 1990.

[BMS04] C. Benzmüller, A. Meier, and V. Sorge. Bridging theorem proving and mathematical knowledge retrieval. In Hutter and Stephan [HS04]. To appear.

[Bou68] N. Bourbaki. Theory of sets. In *Elements of Mathematics*, volume 1. Addison-Wesley, 1968.

[BS82] R. Bartle and D. Sherbert. *Introduction to Real Analysis*. Wiley, 2nd edition, 1982.

[BS98] C. Benzmüller and V. Sorge. A blackboard architecture for guiding interactive proofs. In Giunchiglia [Giu98].

[BS99] Christoph Benzmüller and Volker Sorge. Critical agents supporting interactive theorem proving. In Pedro Borahona and Jose J. Alferes, editors, *Proceedings of the 9th Portuguese Conference on Artificial Intelligence (EPIA'99)*, number 1695 in LNAI, pages 208–221, Evora, Portugal, 1999. Springer.

[BS00] C. Benzmüller and V. Sorge. Ωants – An open approach at combining Interactive and Automated Theorem Proving. In Kerber and Kohlhase [KK00].

[Bun88] A. Bundy. The use of explicit plans to guide inductive proofs. In Lusk and Overbeek [LO88], pages 111–120.

[Bun91] A. Bundy. A science of reasoning. In G. Plotkin J.-L. Lasser, editor, *Computational Logic: Essays in Honor of Alan Robinson*, pages 178–199. MIT Press, 1991.

[Bun94] A. Bundy, editor. *Proceedings of the 12th Conference on Automated Deduction*, number 814 in LNAI. Springer, 1994.

[Bun02] A. Bundy. A critique of proof planning. In *Computational Logic: Logic Programming and Beyond*, number 2408 in LNCS, pages 160–177. Springer, 2002.

[BvHHS90] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam System. In M. Stickel, editor, *Proceedings of the 10th Conference on Automated Deduction*, number 449 in LNCS, pages 647–648, Kaiserslautern, Germany, 1990. Springer.

[CBE+92] 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.

[CGG+92] B. Char, K. Geddes, G. Gonnet, B. Leong, M. Monagan, and S. Watt. *First leaves: a tutorial introduction to Maple V*. Springer, 1992.

[Chu40] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Coq03] Coq Development Team. *The Coq Proof Assistant Reference Manual*. INRIA, 1999-2003. See http://coq.inria.fr/doc/main.html.

[CS98] L. Cheikhrouhou and J. Siekmann. Planning diagonalization proofs. In Giunchiglia [Giu98], pages 167–180.

[CS00] 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.

[Dav65] M. Davis, editor. *The Undecidable: Basic Papers on undecidable Propositions, unsolvable Problems and Computable Functions.* Raven Press Hewlett, New York, 1965.

[Dav83] 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.

[Dav01] 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.

[dN99] H. de Nivelle. Bliksem 1.10 user manual. Technical report, Max-Planck-Institut für Informatik, 1999.

[Dor01] The Doris system is available at http://www.cogsci.ed.ac.uk/ jbos/doris/, 2001.

[EHN94] 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.

[Fie01a] 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.

[Fie01b] A. Fiedler. P.rex: An interactive proof explainer. In Goré et al. [GLN01].

[Fie01c] A. Fiedler. *User-adaptive proof explanation.* PhD thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2001.

[FK00a] A. Franke and M. Kohlhase. System description: MBase, an open mathematical knowledge base. In McAllester [McA00].

[FK00b] Andreas Franke and Michael Kohlhase. System description: Mbase, an open mathematical knowledge base. In David McAllester, editor, *Automated Deduction, CADE-17 (CADE-00) : 17th International conference on Automated Deduction ; Pittsburgh, PA, USA, June 17-20, 2000*, volume 1831 of *Lecture notes in computer science*. Springer, 2000.

[Gan99] H. Ganzinger, editor. *Proceedings of the 16th Conference on Automated Deduction*, number 1632 in LNAI. Springer, 1999.

[Geb99] H. Gebhard. Beweisplanung für die Beweise der Vollständigkeit verschiedener Resolutionskalküle in ΩMEGA. Master's thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 1999.

[Giu98] F. Giunchiglia, editor. *Proceedings of 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'98)*, number 1480 in LNAI. Springer, 1998.

[GLN01] R. Goré, A. Leitsch, and T. Nipkow, editors. *Automated Reasoning – 1st International Joint Conference, IJCAR 2001*, number 2083 in LNAI. Springer, 2001.

[GM93] M. Gordon and T. Melham. *Introduction to HOL – A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[Had44] J. Hadamard. *The Psychology of Invention in the Mathematical Field.* Dover Publications, New York, USA; edition 1949, 1944.

[HJL99] Th. Hillenbrand, A. Jaeger, and B. Löchner. System description: Waldmeister – improvements in performance and ease of use. In Ganzinger [Gan99], pages 232–236.

[HS04] D. Hutter and W. Stephan, editors. *Festschrift in Honour of Jörg Siekmann's 60s Birthday*, LNAI. Springer, 2004. To appear.

[Hua94] X. Huang. Reconstructing Proofs at the Assertion Level. In Bundy [Bun94], pages 738–752.

[Hut00] D. Hutter. Management of change in structured verification. In *Proceedings of Automated Software Engineering, ASE-2000*. IEEE, 2000.

[KF01] 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.

[KK98] C. Kirchner and H. Kirchner, editors. *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in LNAI. Springer, 1998.

[KK00] M. Kerber and M. Kohlhase, editors. *8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus-2000)*. AK Peters, 2000.

[KR00] H. Kirchner and C. Ringeissen, editors. *Frontiers of combining systems: Third International Workshop, FroCoS 2000*, volume 1794 of *LNAI*. Springer, 2000.

[LO88] E. Lusk and R. Overbeek, editors. *Proceedings of the 9th Conference on Automated Deduction*, number 310 in LNCS, Argonne, Illinois, USA, 1988. Springer.

[MAH01] T. Mossakowski, S. Autexier, and D. Hutter. Extending development graphs with hiding. In Heinrich Hussmann, editor, *Proceedings of Fundamental Approaches to Software Engineering (FASE 2001)*, number 2029 in LNCS, pages 269–283, Genova, April 2001. Springer.

[MB88] R. Manthey and F. Bry. SATCHMO: A theorem prover implemented in Prolog. In Lusk and Overbeek [LO88], pages 415–434.

[McA00] D. McAllester, editor. *Proceedings of the 17th Conference on Automated Deduction*, number 1831 in LNAI. Springer, 2000.

[McC94] W. W. McCune. Otter 3.0 reference manual and guide. Technical Report ANL-94-6, Argonne National Laboratory, Argonne, Illinois 60439, USA, 1994.

[McC97] W. McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[Mei00] A. Meier. TRAMP: Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level. In McAllester [McA00].

[Mei03] A. Meier. *Proof Planning with Multiple Strategies*. PhD thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2003.

[Mel96] E. Melis. Island planning and refinement. Seki-Report SR-96-10, Department of Computer Science, Saarland University, Saarbrücken, Germany, 1996.

[Mel98a] E. Melis. AI-techniques in proof planning. In H. Prade, editor, *Proceedings of the 13th European Conference on Artifical Intelligence*, pages 494–498, Brighton, UK, 1998. John Wiley & Sons, Chichester, UK.

[Mel98b] E. Melis. AI-techniques in proof planning. In *European Conference on Artificial Intelligence*, pages 494–498, Brighton, 1998. Kluwer.

[MM00] 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.

[MMP02] A. Meier, E. Melis, and M. Pollet. Towards extending domain representations. Seki Report SR-02-01, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2002.

[MPS01] A. Meier, M. Pollet, and V. Sorge. Classifying Isomorphic Residue Classes. In R. Moreno-Diaz, B. Buchberger, and J.-L. Freire, editors, *A Selection of Papers from the 8th International Workshop on Computer Aided Systems Theory (EuroCAST 2001)*, number 2178 in LNCS, pages 494–508. Springer, 2001.

[MPS02] A. Meier, M. Pollet, and V. Sorge. Comparing Approaches to the Exploration of the Domain of Residue Classes. *Journal of Symbolic Computation, Special Issue on the Integration of Automated Reasoning and Computer Algebra Systems*, 34(4):287–306, October 2002. Steve Linton and Roberto Sebastiani, eds.

[MS99] E. Melis and J. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65–105, 1999.

[MS00] A. Meier and V. Sorge. Exploring properties of residue classes. In Kerber and Kohlhase [KK00].

[MZM00] E. Melis, J. Zimmer, and T. Müller. Integrating constraint solving into proof planning. In Kirchner and Ringeissen [KR00].

[NPW02] T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.

[ORR+96] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. Henzinger, editors, *Computer-Aided Verification, CAV '96*, number 1102 in LNCS, pages 411–414, New Brunswick, NJ, 1996. Springer.

[Pau94] L. Paulson. *Isabelle: A Generic Theorem Prover*. Number 828 in LNCS. Springer, 1994.

[Pol73] G. Polya. *How to Solve it*. Princeton University Press, 1973.

[Rob65] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. J. ACM 12(1): 23-41 (1965).

[RSG98] J. Richardson, A. Smaill, and I. Green. System description: Proof planning in higher-order logic with λClam. In Kirchner and Kirchner [KK98].

[RV01] A. Riazanov and A. Voronkov. Vampire 1.1 (system description). In Goré et al. [GLN01].

[S+95] M. Schönert et al. *GAP – Groups, Algorithms, and Programming*. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1995.

[SBF+02] J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, and M. Pollet. Proof development with OMEGA: Sqrt(2) is irrational. In M. Baaz and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 9th International Conference, LPAR 2002*, number 2514 in LNAI, pages 367–387. Springer, 2002.

[SBF+03] J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, I. Normann, and M. Pollet. Proof development in OMEGA: The irrationality of square root of 2. In F. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, Kluwer Applied Logic series (28), pages 271–314. Kluwer Academic Publishers, 2003. ISBN 1-4020-1656-5.

[SHB+99] 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.

[Sie92] 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.

[Sie04] J. Siekmann. History of computational logic. In D. Gabbay and J. Woods, editors, *The Handbook of the History of Logic*, volume I-IX. Elsevier, 2004. To appear.

[Sor00] V. Sorge. Non-Trivial Computations in Proof Planning. In Kirchner and Ringeissen [KR00].

[Sor01] V. Sorge. *ΩANTS – A Blackboard Architecture for the Integration of Reasoning Techniques into Proof Planning*. PhD thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2001.

[SSY94] G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In Bundy [Bun94].

[vdH01] 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.

[Vor02] A. Voronkov, editor. *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in LNAI. Springer, 2002.

[WAB+99] C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, Th. Engel, E. Keen, C. Theobalt, and D. Topic. System description: SPASS version 1.0.0. In Ganzinger [Gan99], pages 378–382.

[Wel94] D. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994.

[Wie02] F. Wiedijk. The fifteen provers of the world. Unpublished Draft, 2002.

[ZK02] J. Zimmer and M. Kohlhase. System description: The Mathweb Software Bus for distributed mathematical reasoning. In Voronkov [Vor02], pages 138–142.

[ZZ95] J. Zhang and H. Zhang. SEM: A system for enumerating models. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 298–303, Montreal, Canada, 1995. Morgan Kaufmann, San Mateo, California, USA.