Proof Development with ΩMEGA

Jörg Siekmann, Christoph Benzmüller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase*, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge**, Carsten Ullrich, Claus-Peter Wirth, and Jürgen Zimmer

`omega@ags.uni-sb.de`
FR 6.2 Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany

The ΩMEGA proof development system [2] is the core of several related and well integrated research projects of the ΩMEGA research group.

ΩMEGA is a mathematical assistant tool that supports proof development in mathematical domains at a user-friendly level of abstraction. It is a modular system with a central data structure and several complementary subsystems. ΩMEGA has many characteristics in common with systems like NUPRL [1], COQ [23], HOL [13], and PVS [9]. However, it differs from these systems with respect to its focus on *proof planning* and in that respect it is similar to the systems at Edinburgh [6,20]. We present an overview of the architecture of the ΩMEGA system and sketch some of its novel features. Special features of ΩMEGA include (1) facilities to access a considerable number of different reasoning systems and to integrate their results into a single proof structure, (2) support for interactive proof development through some non-standard inspection facilities and guidance in the search for a proof, and (3) methods to develop proofs at a knowledge-based level.

1 System Overview

The ΩMEGA system (cf. Fig. 1) is a representative of the new paradigm of *proof planning* and combines interactive and automated proof construction in *mathematical domains*. ΩMEGA's inference mechanism is an interactive theorem prover based on a higher-order natural deduction (ND) variant of a sorted version of Church's simply typed λ-calculus [8]. The user can interactively construct proofs directly at the calculus level or at the more abstract level of *tactics* and *methods*. Proof construction can be supported by already proven assertions and lemmata and by calls to external systems to simplify or solve subproblems (see Sec. 2).

At the core of ΩMEGA is the *proof plan data structure* $\mathcal{PDS}$ [7] in which proofs and *proof plans* are represented at various levels of granularity and abstraction. The proof plans are classified with respect to a taxonomy of mathematical theories, which are currently being replaced by the mathematical data base

* now at Carnegie Mellon University, Pittsburgh, PA, USA
** now at University of Birmingham, Birmingham, UK

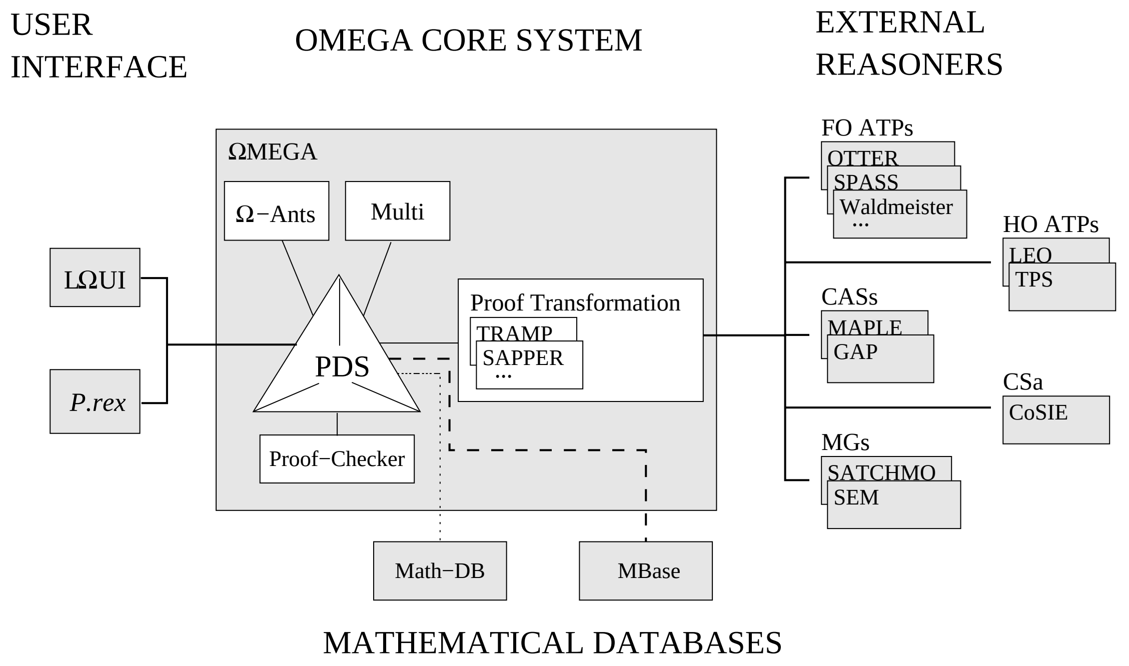

Fig. 1. The architecture of the ΩMEGA proof assistant. Thin lines denote internal interfaces, thick lines denote communication via MATHWEB-SB. The thick dashed line indicates that MBASE is soon to be integrated via MATHWEB-SB. It will replace the current mathematical database (thin dotted line).

MBASE [12]. The user of ΩMEGA, or the proof planner MULTI [17], or the suggestion mechanism Ω-ANTS [3] modify the $\mathcal{PDS}$ during proof development. They can invoke external reasoning systems whose results are included in the $\mathcal{PDS}$ after appropriate transformation. After expansion of these high level proofs to the underlying ND calculus, the $\mathcal{PDS}$ can be checked by ΩMEGA's proof checker. User interaction is supported by the graphical user interface $\mathcal{L}\Omega\mathcal{UI}$ [21] and the proof explainer *P.rex* [10].

Fig. 1 illustrates the architecture of ΩMEGA: the previously monolithic system was split up and separated into several independent modules. These modules are connected via the mathematical software bus MATHWEB-SB [11]. An important benefit is that MATHWEB modules can be distributed over the Internet and are then accessible by other distant research groups as well.

2 External Systems

Proof problems require many different skills for their solution and it is therefore 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 generators (MGs). Their use is twofold: they may provide a solution to a subproblem, or they may give hints for the control of the proof search.

The output of an incorporated reasoning system is translated and inserted as a sub-proof into the $\mathcal{PDS}$, which maintains the proof plan. This is beneficial for interfacing systems that operate at different levels of abstraction, as well as for a human oriented display and inspection of a partial proof. When integrating partial results, it is important to check the correctness of each contribution. In ΩMEGA, this is accomplished by transforming the solution into a subproof, which is then refined to a logic-level proof to be examined by ΩMEGA's proof checker.

The integrated external systems in ΩMEGA are currently the following:

CASs provide symbolic computation, which can be used in two ways: to compute hints to guide the proof search (e.g., witnesses for existentially quantified variables); and to perform 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 module SAPPER [22]. Currently, ΩMEGA uses the systems MAPLE and GAP.

ATPs are employed to solve subgoals. Currently, ΩMEGA uses the first-order ATPs BLIKSEM, EQP, OTTER, PROTEIN, SPASS, WALDMEISTER, and the higher-order systems TPS and $\mathcal{LEO}$. The first-order ATPs are connected via TRAMP [15], 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.

MGs guide the proof search. A model generator provides witnesses for existentially quantified variables or counter-models that show that some subgoal is not a theorem. Currently, ΩMEGA uses the MGs SATCHMO and SEM.

CSs construct mathematical objects with theory-specific properties, such as witnesses for existentially quantified variables. Moreover, a constraint solver can help to reduce the proof search by checking for inconsistencies of constraints. Currently, ΩMEGA employs $\mathcal{C}o\mathcal{SIE}$ [19], a constraint solver for inequalities and equations over the field of real numbers.

3 Support for Proof Development

ΩMEGA supports the user while inspecting the state of a proof, and provides some guidance in the search.

ΩMEGA's graphical user interface $\mathcal{L}\Omega\mathcal{UI}$ displays information on the current proof state in multiple (cross-linked) modalities: a graphical map of the proof tree, a linearized presentation of the proof nodes with their formulae and justifications, and a term browser. When inspecting portions of a proof by these facilities, 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, a natural language explanation of the proof is provided by the system *P.rex* [10], which is interactive and adaptive. The system explains a proof step at the most abstract level (which that user is assumed to know) and it reacts flexibly to questions and requests. While the explanation is in progress, the user can interrupt *P.rex* anytime, if the current explanation is not satisfactory. *P.rex* analyzes the user's interaction and enters into a clarification dialog when needed to identify the reason why the explanation was not satisfactory and re-plans a better explanation, for example, by switching to another level of abstraction.

Another feature of ΩMEGA is the guidance mechanism provided by the suggestion module Ω-ANTS [3]. This module finds a set of possible actions that may be helpful in finding a proof and orders them in a preference list. These actions are an application of particular calculus rules, tactics, or proof methods as well as external reasoners or facts from the knowledge base. Ω-ANTS is based on a hierarchical blackboard upon which data about the current proof state is collected. It is computed by concurrent computational threads, which communicate with the blackboard. This data is then processed by a multi-agent system, which compiles a list of possible actions. This provides not only a very flexible and robust mechanism but also the user does not have to wait until all possible next proof steps have been computed, but intermediate results are shown as they come up. The computation is anytime in the sense that the more time is spent on the computation the better and more complete is the result. A proof step can be executed interactively at any time either from the list of suggested actions or freely chosen by the user. Ω-ANTS can also be used in an automated mode. In this case the actions are ranked heuristically and the best rated action is automatically executed with a possibility to backtrack. These actions may perform proof steps at the natural deduction and tactic level as well as the automatic application of the various integrated external systems.

4 Proof Planning

ΩMEGA's main focus is on knowledge-based proof planning [5,18], where proofs are not conceived in terms of low level calculus rules but at a higher level of abstraction that highlights the main ideas and deemphasizes minor logical or mathematical manipulations on formulae. This viewpoint is realized in the system by proof tactics and abstract proof methods. In contrast to, for instance, the LCF philosophy, our tactics and methods are not necessarily always correct as they have heuristic elements incorporated that account for their strength, so that an informed use of these methods is unlikely to run into failures too often. Since an abstract proof plan may be incorrect for a specific case, its correctness has to be tested by refining it into a logical ND-proof in ΩMEGA's core calculus. This can then be verified by ΩMEGA's proof checker.

Tactics are annotated with partial specifications and then used as *methods* for the proof planner MULTI. Explicitly represented control knowledge helps to find high level proof plans. Traditional proof planning is enhanced in MULTI by using mathematical knowledge to prune the search. Methods are combined to form strategies, which then perform different proof techniques.

5 Case Studies

The novelties of the ΩMEGA system have been tested in several case studies. They particularly illustrate the useful interplay of the various components, such as proof planning supported by heterogeneous external reasoning systems.

A typical example for a class of problems that can not be solved by traditional automated theorem provers is the class of ϵ–δ–proofs [18]. This class was originally proposed by W. Beldsoe [4] 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 a similar statement for multiplication. The difficulty of this domain arises from the need for arithmetic computation and suitable instantiation of meta-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 computations are done with the computer algebra system MAPLE, and an appropriate instantiation for δ is computed by the constraint solver $\mathcal{C}o\mathcal{SIE}$.

Another class of problems we tackled with proof planning is concerned with residue class problems [16]. In this domain we show theorems as e.g. the fact that the residue class structure $(\mathbb{Z}_5, \bar{+})$ is associative, has a unit element etc., 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. Moreover, we prove for two given structures whether they are isomorphic or not. Although the problems in this domain are in the range of traditional automated theorem provers it is nevertheless an interesting domain for proof planning since multi strategy proof planning generates substantially different proofs based on different proof ideas. For instance, one strategy we realized in MULTI converts statements on residue classes into statements on numbers and then applies an exhaustive case analysis. Another strategy tries to reduce the original goal into sets of equations to which MAPLE is applied to check whether the equality actually holds. Moreover, the computer algebra systems MAPLE and GAP are employed to compute witnesses for particular elements, for instance, to compute $\bar{0}_5$, the unit element of $(\mathbb{Z}_5, \bar{+})$.

Another more recent case study is the proof of the *Irrationality of* $\sqrt{2}$. Here the user proposes interactively the main conceptual steps. Simple but painful logical subderivations are then passed to the ATPs and simple computations can be done by the CASs.

The ΩMEGA system is available at `http://www.ags.uni-sb.de/~omega`.

References

1. S. Allen, R. Constable, R. Eaton, C. Kreitz, and L. Lorigo. The Nuprl open logical environment. In *Proc. of CADE-17*, LNAI 1831. Springer, 2000.
2. C. Benzmüller et al. ΩMEGA: Towards a mathematical assistant. In *Proc. of CADE-14*, LNAI 1249. Springer, 1997.

3. C. Benzmüller and V. Sorge. Ω-ANTS – An open Approach at Combining Interactive and Automated Theorem Proving. In *Proc. of Calculemus-2000.* AK Peters, 2001.
4. W.W. Bledsoe. Challenge problems in elementary analysis. *Journal of Automated Reasoning*, 6:341–359, 1990.
5. A. Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In *Proc. of CADE-9*, LNCS 310. Springer, 1988.
6. A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303-324, 1991. Earlier version available from Edinburgh as DAI Research Paper No 413.
7. L. Cheikhrouhou and V. Sorge. $\mathcal{PDS}$ — A Three-Dimensional Data Structure for Proof Plans. In *Proc. of ACIDCA'2000*, 2000.
8. A. Church. A Formulation of the Simple Theory of Types. *The Journal of Symbolic Logic*, 5:56–68, 1940.
9. S. Owre et al. PVS: Combining specification, proof checking and model checking. In *Proc. of CAV-96*, LNCS 1102. Springer, 1996.
10. A. Fiedler. *P.rex*: An interactive proof explainer. In *Proc. of IJCAR 2001*, LNAI 2083. Springer, 2001.
11. M. Kohlhase and J. Zimmer. System description: The MathWeb Software Bus for Distributed Mathmatical Reasoning. In *Proc. of CADE-18*, LNAI. Springer, 2002.
12. A. Franke and M. Kohlhase. System description: MBASE, an open mathematical knowledge base. In *Proc. of CADE-17*, LNAI 1831. Springer, 2000.
13. M. Gordon and T. Melham. *Introduction to HOL – A theorem proving environment for higher order logic.* Cambridge University Press, 1993.
14. W. McCune. Otter 3.0 reference manual and guide. Technical Report ANL-94-6, Argonne National Laboratory, Argonne, Illinois 60439, USA, 1994.
15. A. Meier. TRAMP: Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level. In *Proc. of CADE-17*, LNAI 1831. Springer, 2000.
16. A. Meier, M. Pollet, and V. Sorge. Comparing Approaches to Explore the Domain of Residue Classes. *Journal of Symbolic Computations*, 2002. forthcoming.
17. E. Melis and A. Meier. Proof Planning with Multiple Strategies. In *Proc. of CL-2000*, LNAI 1861. Springer, 2000.
18. E. Melis and J. Siekmann. Knowledge-Based Proof Planning. *Artificial Intelligence*, 115(1):65–105, 1999.
19. E. Melis, J. Zimmer, and T. Müller. Integrating constraint solving into proof planning. In *Proc. of FroCoS 2000*, LNAI 1794. Springer, 2000.
20. J.D.C Richardson, A. Smaill, and I.M. Green. System description: Proof planning in higher-order logic with λ-CLAM. In *Proc. of CADE-15*, LNAI 1421, Springer, 1998.
21. J. Siekmann et al. $\mathcal{LOUI}$: $\mathcal{L}$ovely ΩMEGA $\mathcal{U}$ser $\mathcal{I}$nterface. *Formal Aspects of Computing*, 11:326–342, 1999.
22. V. Sorge. Non-Trivial Computations in Proof Planning. In *Proc. of FroCoS 2000*, LNAI 1794. Springer, 2000.
23. Coq Development Team. *The Coq Proof Assistant Reference Manual.* INRIA. see `http://coq.inria.fr/doc/main.html`.