ΩMEGA: Towards a Mathematical Assistant

Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Karsten Konrad, Andreas Meier, Erica Melis, Wolf Schaarschmidt, Jörg Siekmann, Volker Sorge*

Fachbereich Informatik, Universität des Saarlandes
D-66041 Saarbrücken, Germany. `http://jswww.cs.uni-sb.de/`

Abstract. ΩMEGA is a mixed-initiative system with the ultimate purpose of supporting theorem proving in main-stream mathematics and mathematics education. The current system consists of a proof planner and an integrated collection of tools for formulating problems, proving subproblems, and proof presentation.

1 Introduction

The dream of machine assistance in mathematical problem solving by far predates the advent of electronic computers. Current classical automated theorem provers can prove some non-trivial mathematical theorems. However, while humans can cope with long and complex proofs and possess strategies to avoid less promising proof paths, automated theorem proving suffers from exponential search spaces. Consequently, a combination of the power of automated tools with human-like abilities and/or user interaction is necessary in order to be able to prove main-stream mathematical problems.

Proof planning, introduced by Bundy for induction theorem proving [Bun88], seems to provide a promising framework for systems that truly assist mathematicians. It offers a cognitively adequate model for integrating powerful domain-specific methods with meta-level (automated and human) control.

In the following, we will describe the prototypical system ΩMEGA that explores the use of proof planning together with tools that provide high-level proof tools and user support to come closer to the vision of a mathematical assistant system. Note that these tools are essential for the projected usefulness of a mathematical assistant system as a whole.

2 The Architecture and Components of ΩMEGA

The entire process of theorem proving in ΩMEGA can be viewed as an interleaving process of proof planning, plan execution and verification.

2.1 Hierarchical Plan Data Structure

The central data structure for the overall process is the *Proof plan Data Structure* ($\mathcal{PDS}$). This is a hierarchical data structure that represents a (partial) proof at different levels of abstraction (called *proof plans*). It is represented as a directed acyclic graph, where the nodes are justified by (LCF-style) tactic applications.

* This work was supported by the Deutsche Forschungsgemeinschaft, SFB 378

Conceptually, each such justification represents a proof plan (the *expansion* of the justification) at a lower level of abstraction that is computed when the tactic is executed[1]. In ΩMEGA, we explicitly keep the original proof plan in an expansion hierarchy. Thus the $\mathcal{PDS}$ makes the hierarchical structure of proof plans explicit and retains it for further applications such as proof explanation or analogical transfer of plans.

Once a proof plan is completed, its justifications can successively be expanded to verify the well-formedness of the ensuing $\mathcal{PDS}$. This verification phase is necessary, since the correctness of the different components (in particular, that of the external ones) cannot be guaranteed. When the expansion process is carried out down to the underlying ND-calculus, the soundness of the overall system relies solely on the correctness of the verifier and of ND. This also provides a basis for the controlled integration of external reasoning components if each reasoner's results can (on demand) be transformed into a sub-$\mathcal{PDS}$.

A $\mathcal{PDS}$ can be constructed by automated or mixed-initiative planning, or pure user interaction that can make use of the integrated tools. In particular, new pieces of $\mathcal{PDS}$ can be added by directly calling tactics, by inserting facts from a data base, or by calling some external reasoner (cf. 3.1) such as an automated theorem prover or a computer algebra system. Automated proof planning is only adequate for problem classes for which method- and control knowledge have already been established.

2.2 Proof Planning

ΩMEGA's proof planner is based on an extension of the well-known STRIPS algorithm that can be evoked to construct a proof plan for a node g (the *goal node*) from a set I of *supporting nodes* (the initial state) using a set Ops of proof planning operators, here called methods. A *method* is a (partial) specification of a tactic in a meta-level language.

Proof planning starts with a null plan that consists of a finish-node containing no postcondition and the (open) goal g as precondition and start-nodes, each containing an assumption from I as postcondition and no precondition. The finish-node belongs to the trailer of the plan and the start-nodes belong to the header. The null plan's head-state is I and its tail-state consists of g. Backward proof planning starts with an open goal g. It searches for a method M applicable to g and introduces a node with M into the trailer of the proof plan. g gets status *closed*, if the subgoals g_i produced by the application of M are not in the head state, they become the new *open* subgoals. The planner continues to search for a method applicable to one of the open subgoals and terminates if there are no more open goals. Forward planning is possible too. It produces new assumptions and enlarges the header of a plan. If an open subgoal agrees with an assumption, it is closed. The planner can be used for stepwise as well as for continued planning. Currently, ΩMEGA uses forward and backward state-space search. Planning is combined with hierarchical expansion of methods and precondition abstraction.

[1] This proof plan can be recursively expanded, until we have reached a proof plan, which is in fact a fully explicit proof, since all nodes are justified by the inference rules of a higher-order variant of Gentzen's calculus of natural deduction (ND).

The plans found by this procedure are directly incorporated into the $\mathcal{PDS}$ as a separate level of abstraction. Furthermore, the proof planner also stores the reasons for its decisions for later use in proof explanation and analogy.

In this model, the actual reasoning competence of the planner and the user builds upon the availability of appropriate methods together with meta-level control knowledge that guides the planning. At the moment, ΩMEGA provides user-defined method ratings as a means of control and can use analogy as a control strategy of the planner.

3 Support Systems

Several integrated tools support the user in interacting with the system. Some of them are also available to the planner.

3.1 Integration of External Reasoners

A reasoner $\mathcal{R}$ can be integrated into ΩMEGA in three different settings: In

- **interactive calls**, $\mathcal{R}$ is represented as a command `call-RSys` that invokes the reasoner on a particular sub-problem and returns the result,
- **proof planning**, $\mathcal{R}$ is represented as a method whose specification contains knowledge about the problem solving behavior and option settings for $\mathcal{R}$.
- **methods**, $\mathcal{R}$ can serve as a justification of a declaratively given subgoal that is left to be proved by $\mathcal{R}$.

In any case, the proof found by $\mathcal{R}$ must be transformed into a $\mathcal{PDS}$ eventually, since this is the proof-theoretic basis of ΩMEGA.

The current ΩMEGA supports the automated theorem prover OTTER[McC94] and an experimental computer algebra system as external reasoners. We have described the integration of OTTER in [HKK$^+$94] and the proof transformation in [HF96], so we will concentrate on the computer algebra integration.

3.2 Integration of Computer Algebra

Traditional deduction systems are weak in computing with concrete mathematical objects, such as natural numbers. In contrast to that, computer algebra systems (CASs) manipulate highly optimized representations of the objects and are therefore very useful for solving subgoals in mathematical deduction. Moreover, a look into mathematical textbooks reveals that usually neither computation nor purely logical deduction dominates proofs. ΩMEGA integrates an experimental CAS that can manipulate polynomials over rational numbers.

CASs are very complex programs and only trustworthy to a limited extent. Since ΩMEGA aims at correct proofs, it uses the mathematical knowledge implicit in the CAS to extract proof plans that correspond to the mathematical computation in the CAS. Note that this approach necessitates a data base of the mathematics behind the algorithms to guarantee correctness and explanations (for details cf. [KKS96]).

3.3 Theory Data Base

Since methods and control knowledge used in proof planning are mostly domain-specific, ΩMEGA organizes the mathematical knowledge in a hierarchy of theories.

Theories represent signature extensions, axioms, definitions, theorems, lemmata, and finally methods that make up typical established mathematical domains. Each theorem has its home theory and therefore has access to the theory's signature extensions, axioms, definitions, and lemmata without explicitly introducing them.

A simple inheritance mechanism allows the user to incrementally build larger theories from smaller parts. Currently, ΩMEGA comes with a small hierarchy of basic theories for didactic purposes.

3.4 Proof Explanation

Proof presentation is one important feature of a mathematical assistant that has been neglected by traditional deduction systems. ΩMEGA employs an extension of the PROVERB system [HF96] developed by our group that allows for presenting proofs and proof plans in natural language. In order to produce coherent texts that resemble those found in mathematical textbooks, PROVERB employs state-of-the-art techniques of natural language processing.

Due to the possibly hierarchical nature of $\mathcal{PDS}$ proofs, these can be verbalized at more than one level of abstraction, which can be selected by the user. Since a user will normally want to vary the level of abstraction in the course of a proof, the current verbalization facility will be extended to one that explains proofs to users guided by their feedback in the future.

4 Progress and Availability

The ΩMEGA system is an experiment in designing and integrating components of mathematical assistant systems. It is still in an incomplete, prototypical state. However, the integration of external reasoners, and some control strategies have extended the capability of ΩMEGA to a level that is beyond the scope of monolithic automated deduction systems, as has been demonstrated, e.g., in the case of optimization questions in an economics masters exam.

ΩMEGA runs on Common Lisp together with it's object system Clos and has been tested on Allegro, and Gnu Common Lisp. The source code is available from `ftp://jswww.cs.uni-sb.de/pub/omega/omega.tgz`.

References

[Bun88] A. Bundy. The use of explicit plans to guide inductive proofs. *CADE 9*, LNCS 310, pages 111–120, 1988.

[HF96] X. Huang and A. Fiedler. Presenting machine-found proofs. In *CADE 13*, LNCS 1104, pages 221–225, 1996.

[HKK+94] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts, and J. Siekmann. *Ω*-MKRP a proof development environment. *CADE12*, LNAI 814, pages 788–792, 1994.

[KKS96] M. Kerber, M. Kohlhase, and V. Sorge. Integrating computer algebra with proof planning. *DISCO'96*, LNCS 1128, pages 204–215, 1996.

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