System Description: MULTI
A Multi-Strategy Proof Planner

Andreas Meier and Erica Melis

German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
{ameier,melis}@dfki.de, http://www.activemath.org/{~ameier,~melis}

1 Introduction

The CASC competitions among automated theorem provers show that there is no single system that outperforms all other systems in all domains. One reaction to this observation is the combination of several systems in a competitive (e.g., the SSCPA system) or cooperative manner (e.g., the CSSCPA and TECHS systems). Thereby, general-purpose heuristics select promising systems to be executed and promising intermediate results to be exchanged. Typically, exchanged results are restricted to clauses or equations that are accepted by all systems. The use of particular domain-specific services for particular subtasks such as the construction of mathematical objects and their flexible cooperation with other services guided by mathematically motivated control knowledge is not possible.

Multi-strategy proof planning [10] provides a framework for the flexible collaboration of independent services, so-called *strategies*, that can realize various operations on proof plans guided by explicitly represented control knowledge. This paper describes the realization of multi-strategy proof planning in the MULTI system and illustrates its application in conducted case studies. Although MULTI focuses to proof planning techniques, its architecture and concepts are generally applicable for a flexible and knowledge-based cooperation of independent services in theorem proving.

2 Proof Planning and Multi-Strategy Proof Planning

Proof planning [1, 11] is a theorem proving technique, which constructs a proof at the abstract level of so-called *methods*, i.e., tactics enriched by explicit pre- and postconditions. Methods result from the analysis of common structures or common procedures of a family of proofs. They can encode not only general proof steps but also steps particular to a mathematical domain. Mathematically motivated heuristics are encoded in the control knowledge employed in the search for a solution plan consisting of a sequence or hierarchy of methods.

The idea of multi-strategy proof planning is to combine services instead of performing simple proof planning at the level of methods only. These separate but flexibly collaborating services, so-called *strategies*, realize various proof plan modifications and refinements. For instance, strategies can realize different kinds of backtracking or they can provide particular services of external systems

that can be flexibly integrated with strategies that apply methods. Thereby, the search for applicable strategies introduces an additional hierarchical level into proof planning that allows for the incorporation of further mathematically motivated knowledge.

3 The MULTI System

Algorithms and Strategies MULTI distinguishes algorithms and strategies. *Algorithms* are independent and parameterized proof plan refinement and modification processes. Currently, MULTI employs 6 different algorithms that all work on partial proof plans: PPlanner for method introduction, InstVar for the instantiation of variables, BackTrack for the deletion of steps from the plan under construction, Exp for the expansion of complex methods, ATP for closing a goal by calling traditional automated theorem provers, and CPlanner for case-based planning. The former three algorithms are decoupled and parameterized functionalities of a simple proof planning process, whereas the latter three algorithms introduce new functionalities. The framework is open to introduce further algorithms that can contribute to proof plan construction.

Strategies specify different services or behaviors of the algorithms. Strategies result from instantiations of the parameters of an algorithm. For instance, the parameters of PPlanner include a set of methods. When such a strategy is executed, then PPlanner introduces only steps that use the methods specified in the strategy. A parameter of InstVar is the function that determines how the instantiation for a variable is computed. A parameter of BackTrack is the function that computes a set of refinement steps that will be deleted from the proof plan.

Technically, strategies are implemented as data structures with three slots: (1) an application condition stating when the strategy is applicable, i.e., the legal applicability knowledge, (2) the algorithm which is employed by the strategy, and (3) the parameter setting with which the algorithm is employed. Examples of strategies can be found in [7, 8].

Strategic Control The heuristic knowledge of the utility of the application of strategies under certain conditions is separated from the strategies. It is declaratively encoded in *strategic control rules*, which provide the basis for meta-reasoning and a global guidance in MULTI. Technically, a control rule is an IF-THEN pair, where the IF-part is a predicate about the proof planning status and the THEN-part is an action that ranks or prunes possible strategy applications.

Architecture In order to allow for the flexible cooperation of the independent strategies guided by meta-reasoning, MULTI uses a blackboard architecture. In blackboard systems [5], independent components, so-called knowledge sources, collaborate to solve a problem whose solution state is placed on a blackboard that all knowledge sources can access. Figure 1 shows the blackboard architecture of MULTI, which is similar to the *BB1* blackboard system (see [5]). The architecture

has two blackboards and two kinds of knowledge sources, one for the proof planning problem and one for the control problem, i.e., the problem of deciding which strategy to apply next.

The proof blackboard contains the current proof plan as well as the history of the proof planning process. The strategies are the knowledge sources that work on this proof blackboard. The control blackboard contains job offers, demands, and a memory. When a strategy's condition part is satisfied, the strategy posts its applicability information, i.e., a *job offer*, onto the control blackboard. The *MetaReasoner* is the knowledge source working on the control blackboard. It ranks the job offers by evaluating the strategic control-rules. The *Scheduler* looks up the control blackboard, takes the highest ranked job offer and executes it.

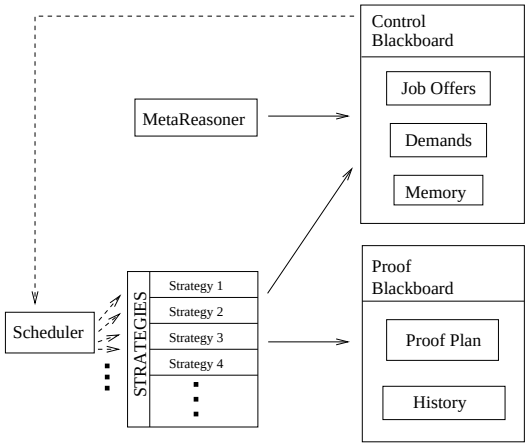

Fig. 1. MULTI's blackboard architecture.

A strategy that is executed changes the proof plan and records its steps in the history. Executed strategies can also post demands and memory entries onto the control blackboard, which allows to interleave strategy executions. For instance, if the currently executed strategy S can continue only, if another strategy S' is executed first, then S is interrupted, its status is saved in the memory, and a demand for S' is placed onto the control blackboard. After the execution of S' the strategy S can be re-invoked from the memory.

How MULTI Operates In a nutshell, MULTI operates according to the following cycle, in which no order or sequence of strategies is hard-coded:

- *Job Offers:* Applicable strategies post their applicability (for the current partial plan) as 'job offers' onto the control blackboard.
- *Guidance:* Strategic control rules are evaluated to rank the job offers.
- *Invocation:* The strategy with the highest ranked job offer is invoked.
- *Execution:* The strategy works on the proof blackboard and can place new demands and memory entries onto the control blackboard.

Discussion To achieve more autonomous strategies the heuristic knowledge could be part of the strategies as well. This would result in a multi-agent approach where the strategies negotiate with each other which strategy to apply next. Our reason for separating heuristic utility and legal feasibility knowledge and for implementing a blackboard architecture is the availability of global control knowledge that favors a central control mechanism over many interacting and locally deciding agents.

4 Experiences and Case Studies

MULTI's strategy level allows for the formalization and incorporation of proof knowledge in strategies and strategic control rules that is beyond the method-level and its control. The knowledge encoded can be diverse. For instance, strategies can describe different techniques to prove a class of problems. Strategies can also describe different ways of backtracking or different ways of constructing mathematical objects to instantiate variables. Strategic control rules can describe, for instance, in which order to try several applicable strategies. They can also guide failure handling. In order to illustrate the advantages of MULTI's strategy level we briefly discuss its application to the residue class domain (see [8] for a detailed description). Other major case studies conducted with MULTI tackle ϵ–δ–proofs [7] and permutation group problems [2].

Residue class conjectures classify given residue class structures wrt. their algebraic category. An example theorem is "the residue class structure $(\mathbb{Z}_5, \bar{+})$ is associative". Other problems from this domain concern the isomorphy of two algebraic structures, e.g., "the residue class structures $(\mathbb{Z}_5, \bar{+})$ and $(\mathbb{Z}_5, \bar{*})$ are not isomorphic". We proved about 19.000 residue class conjectures with MULTI.

Although the problems in this domain are within the range of difficulty a traditional automated theorem prover can handle (see experiments in [8]), it is nevertheless an interesting case study for proof planning, since with MULTI it is possible to generate substantially different and intuitive proofs based on entirely different proof ideas. The parameters of PPlanner can configure strategies taking advantage of different proof settings that mirror proof techniques existing in mathematics. For instance, we realized the three intuitive techniques *naive case-split*, *equational reasoning*, and *reduction to known facts* for this domain in three PPlanner strategies, with different sets of methods and control rules. The availability of three techniques to tackle the problems of the domain extends the robustness of the combined proof planning approach in the sense that if one strategy does not succeed, then another one may continue. A strategic control rule attempts to use the generally most efficient strategy first and the most reliable one last. Moreover, since MULTI supports the switching of strategies, different subproblems in one proof process can be tackled by different strategies.

Reasoning about impasses is a natural ingredient of meta-reasoning at the strategic level: strategic control rules can analyze and exploit failures to guide subsequent strategy applications (see also [7]). In the residue class domain, there exists knowledge of suitable backtrack points for certain failures and proof situations. This knowledge is encoded into strategic control rules, which guide the application of different BackTrack strategies that realize the appropriate backtracking. This considerably prunes the traversed search space.

MULTI's concept of strategies and algorithms explicitly supports the incorporation of external systems to provide services and to solve subtasks. For instance, to prove that two given residue class structures are not isomorphic, we realized discriminating in MULTI, which requires finding a property P such that P holds for the one structure but not for the other. Discriminating is mainly realized by a main PPlanner strategy, which, however, is interleaved with the call of 3 different

external systems in different strategies (see [9] for details). The computation of P is provided by the theory formation system HR [3], which is called in an InstVar strategy. In order to show that P holds for the one structure but not for the other, computer algebra systems are employed within another InstVar strategy. Finally, the resolution prover SPASS is used in a strategy of ATP to prove that $\forall X_{\blacksquare} \forall Y_{\blacksquare} P(X) \wedge \neg P(Y) \Rightarrow X \not\sim Y$ holds (while this step is fairly obvious for a mathematician, it is crucial for a formal proof).

5 Conclusion and Availability

We presented the multi-strategy proof planner MULTI. The most important features of MULTI are the usage of independent strategies realizing various proof plan services and their flexible cooperation guided by declaratively represented strategic meta-reasoning. MULTI differs from other approaches of proof planning embodied by $\lambda CLAM$ [12] and ISAPLANNER [4] (1) by its notion of strategies that covers diverse services such as different kinds of backtracking or calls of external systems and (2) by its flexible combination of strategies that is not pre-defined in compound hierarchies of steps as in $\lambda CLAM$ and ISAPLANNER.

MULTI is implemented in Allegro Common Lisp as a component of the mathematical assistant system ΩMEGA [6]. Its source code is available as part of the ΩMEGA source code, see `http://www.ags.uni-sb.de/~omega`. Further information can be found at: `http://www.ags.uni-sb.de/~ameier/multi.html`.

References

1. A. Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In *Proc. of CADE–9*, pages 111–120, 1988.
2. A. Cohen, S. Murray, M. Pollet, and V. Sorge. Certifying solutions to permutation group problems. In *Proc. of CADE–19*, pages 258–273, 2003.
3. S. Colton. The HR program for theorem generation. In *Proc. of CADE–18*, pages 280–284, 2002.
4. L. Dixon and J.D. Fleuriot. IsaPlanner: A prototype proof planner in Isabelle. In *Proc. of CADE–19*, pages 279–283, 2003.
5. R. Engelmore and T. Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.
6. The OMEGA Group. Proof Development with OMEGA. In *Proc. of CADE–18*, pages 144–149, 2002.
7. A. Meier and E. Melis. Failure reasoning in multiple-strategy proof planning. *Electronic Notes in Theoretical Computer Science*, To appear 2005.
8. A. Meier, M. Pollet, and V. Sorge. Comparing Approaches to Explore the Domain of Residue Classes. *Journal of Symbolic Computation*, 34(4):287–306, 2002.
9. A. Meier, V. Sorge, and S. Colton. Employing Theory Formation to Guide Proof Planning. In *Proc. of Joint AISC and Calculemus 2002*, pages 275–289, 2002.
10. E. Melis and A. Meier. Proof planning with multiple strategies. In *Proc. of the First Intern. Conference on Computational Logic (CL2000)*, pages 644–659, 2000.
11. E. Melis and J. Siekmann. Knowledge-Based Proof Planning. *Artificial Intelligence*, 115(1):65–105, 1999.
12. J.D.C. Richardson, A. Smaill, and I.M. Green. System description: Proof planning in higher-order logic with $\lambda Clam$. In *Proc. of CADE–15*, pages 129–133, 1998.