# $\lambda$*Clam* : A Hierarchical Proof Planning System with Proof Critics

Louise A. Dennis[1] and Ewen Maclean[2]

[1] School of Computer Science and Information Technology, University of Nottingham, Nottingham, NG8 1BB `lad@cs.nott.ac.uk`
[2] Division of Informatics, University of Edinburgh, Appleton Tower, Edinburgh

This abstract describes the implementation of the most recent version of the *Clam* proof planning system. It is called $\lambda$*Clam* and is implemented in the $\lambda$Prolog programming language.

Proof planning is an AI-search based technique for automated proof. It has been applied with particular success to proofs by mathematical induction.

Proof planning was first suggested by Bundy [1]. A proof plan in $\lambda$*Clam* is a proof of a theorem at some level of abstraction presented as a tree. A proof plan is generated using AI-style planning techniques. The planning operators used by a proof planner are called *proof methods*.

Proof methods are defined by their pre- and post-conditions which are used by the planner to form the proof plan. The defining feature of a proof method is this presentation in terms of pre- and post- conditions. Methods themselves need not contain any guarantee of soundness and generally refer to a tactic in an LCF-style theorem prover which can be run to create a sound proof.

Our implementation of the logical parts of $\lambda$*Clam* governing the syntactic structures, goals and methods, is heavily based on the work of Felty [2].

Proof method application is governed by the preconditions (which may be either legal or heuristic in nature) and by a *proof strategy* which restricts the methods available depending on the progress through the proof. For instance we may wish to simplify a goal as much as possible by applying a rewriting method exhaustively before considering other procedures such as checking for tautologies.

Proof plans can be further refined using *proof critics* [3] which inspect partial proof plans and may revise earlier decisions based on information that becomes available later in the planning process.

The original critics planner used a "flat" planner rather than the hierarchical version implemented in $\lambda$*Clam*. In order to allow us to revise our proof plans as we go we need to be able to attach to each node in the plan a record of its current position in the method heirarchy - this allows us to edit this *methodical* and restart the planning from that point. We call this approach to proof planning a *partial data structure planner* and its implementation is based on the theory of continuations [4]. The expressions made up of methodicals form the *proof strategies* for each class of proof and allow strategies to be described in an intuitive fashion in terms of loops (repeats), sequences and branches.

We have similarly extended the language of critics with *critic expressions* which are defined as critics composed via criticals and handled using continuations[1]. In this way it is possible to implement more flexible sequences of critics and to associate a critic expression with more than one method (if appropriate). It also more clearly defines critics as operating upon and mutating partial plans since once activated they then alter the existing plan structure and the proof strategies contained within it.


## 1 Acknowledgments

The authors of this abstract are, at present, the main developers of the $\lambda$*Clam* system but many other people have, and indeed still are, working on it and many of the original features of the planner are attributable to these people. Developers include, but are not limited to, James Brotherston, Ewen Denney, Jeremy Gow, Ian Green, Julian Richardson, Gordon Reid, Andreas Schöter and Alan Smaill.


## References


1. A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.
2. A. Felty. Implementing tactics and tacticals in a higher-order logic programming language. *Journal of Automated Reasoning*, 11(1):43–81, 1993.
3. A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning – LPAR 92, St. Petersburg*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 178–189. Springer-Verlag, 1992.
4. J. Richardson and A. Smaill. Continuations of proof strategies. In M. P. Bonacina and B. Gramlich, editors, *Proc. 4th International Workshop on Strategies in Automated Deduction (STRATEGIES 2001)*, Siena, Italy, june 2001. Available from `http://www.logic.at/strategies/strategies01/`.
5. Julian Richardson and Alan Smaill. Continuations of proof strategies. In Tobias Nipkov Rajeev Gor, Alexander Leitsch, editor, *Proceedings of International Joint Conference on Automated Reasoning*, pages 130–139, 2001.


[1] The implementation of critics in $\lambda$*Clam* is based heavily on the work of Jeremy Gow for his PhD thesis.