Proof Planning: Comparing ΩMEGA, λCLAM and ISAPLANNER *

Lucas Dixon[1] Mateja Jamnik[2] Martin Pollet[3]

lucasd@dai.ed.ac.uk Mateja.Jamnik@cl.cam.ac.uk pollet@ags.uni-sb.de

[1] CISA, School of Informatics, Univ. of Edinburgh, Appleton Tower, 11 Crichton Str., Edinburgh, EH8 9LE, Scotland, UK

[2] University of Cambridge Computer Laboratory, J.J. Thomson Avenue, Cambridge, CB3 0FD, England, UK

[3] Fachbereich Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany

Abstract In this work we propose to compare the approaches taken by the ΩMEGA, λCLAM and ISAPLANNER proof planners in terms of how proof planning methods are written and in terms of the underlying architecture. We investigate the nature of tactics, tacticals, methods and methodicals. The aim is to gain insight into how these objects relate to the search space and whether we can provide a unified view of all the variants. These insights will illuminate the issues in the choice of object language, the organisation and hierarchy of objects in the search space, and the choice of planning and search techniques in automated reasoners.

Introduction

Proof planning was introduced by Alan Bundy [1] as a new paradigm for proof automation. Rather than using low-level logical inference rules, proof is automated using so called *proof methods* which capture common patterns of reasoning. For example, the notion of a proof by induction can be encoded as a proof method.

Proof planners search for a *proof plan* of a theorem. This can then be be executed to derive a fully formal proof in terms of the underlying logical inference rules. The search space tends to be dramatically smaller than that at the level of inference rules [2, 5]. This is due to the more abstract nature of proof methods. Proof planning also focuses on providing a rich structure to organise the application of proof methods. A salient feature of proof planning is that it generates proofs where the reasoning patterns are transparent, and where failure can be patched by proof critics.

A proof method in proof planning typically consists of a triple – preconditions, postconditions and a tactic. This is an idealised view and in practice the structure of methods varies depending on the proof planner in use. More precisely, a proof method is a partial specification (in a meta-language) of a tactic, i.e., it specifies the proof steps that the tactic performs to construct a proof. We will use the term *tactic* to refer to functions that apply logical inference rules to reduce a goal to a number of subgoals.

In our current work, we plan to compare the approaches taken by the following proof planning systems: λCLAM [7] (the successor of the first proof planner CLAM [1]); the ΩMEGA proof planner [8]; and the ISAPLANNER system [4], a generic framework for proof planning in the interactive theorem prover ISABELLE [6]. We want to compare these three proof planners with respect to the following criteria:

Architecture: the underlying structure of each proof planning system. In particular, we are interested in how the search space is organised, how it relates to the encoded techniques, and which search strategies are employed. This also includes examining the choice of planning operators and how this choice effects the expressivity of the language for encoding techniques.

Interface: the interaction between the proof planning system and users who are encoding techniques Thus, here, we are not interested in human-computer interaction issues, such as graphical interfaces etc. Rather, we are interested in the language with which the user can construct new techniques. This includes examining proof methods and how they are structured (pre-, post-conditions, tactics).

Power: what type of theorems is each proof planner able to prove, and why is this the case? In particular, what type of theorems can be proved with some proof planners but not the others? We are also interested in the domains and types of problems to which the different systems have been, and can be applied.

Note that we use the word "technique" to denote a more abstract notion than a proof method. Essentially, a proof method is an embodiment of a technique. In this way, we can be more general in our comparison between different proof planners (and possibly even different theorem provers in the future).

Building Blocks for Comparison

Here, we describe the main components of our proof planners using a unified terminology.

Object Language

The *object language* is defined by a collection of logical rules. These can be applied using *tactics* which can them-

*This work was supported by an EPSRC grant GR/N37314/01, an EPSRC Advanced Research Fellowship GR/R76783, SFB 378 Project grant, and European Commission IHP Calculemus Project grant HPRN-CT-2000-00102.

selves be composed of other tactics using *tacticals*. The important characteristic of tactics is that, being the application of the logical rules, they are to be soundness preserving: their application reduces a goal to a number of subgoals. In terms of power, the object language is important as it provides the mechanisms for representing the problem. It is also important that new theories can easily be defined in such a way that the existing methods in a proof planner can be applied.

All three systems support higher order object languages and a theory hierarchy. In ΩMEGA, the proof plans can be expanded, i.e., executed upon request. λCLAM is not currently linked to an object level theorem prover. In ISAPLANNER, proof planning is interleaved with the proof plan's execution.

Technique Language

While complex techniques can be encoded in tactics, one of the central purposes of the proof planning paradigm is to provide a more convenient and powerful way to encode knowledge. The three systems provide different support for the encoding of mathematical knowledge.

λCLAM has atomic and compound methods. Atomic methods are written by specifying related higher order patterns, in particular a set of preconditions, postconditions, and a tactic pattern are defined. Compounds methods structure submethods using methodicals such as THEN which applies the first method and then the second one. λCLAM also has proof critics which are used to great effect with rippling [3].

ΩMEGA also uses atomic methods, but instead of providing a way to combine the methods, it provides two further levels of organisation: control rules and strategies. Control rules specify how the search with methods is performed, and strategies specify collections of control rules and methods to govern their interaction.

ISAPLANNER uses a single an extensible language for encoding techniques. This provides the flexibility to write λCLAM style methods, methodicals, and proof critics. However, the methodical language is restricted and can be extended as needed.

We feel that an important step in comparing these systems is to understand how techniques in one system can be defined in another. This will also serve as a tool to allow the migration of techniques developed in one system to others.

Planning Algorithm

To examine the approaches in still further depth, we intend to examine the algorithms for applying the various styles of encoded knowledge. Understanding the planning algorithm is the central problem in understanding the various architectures. Additionally, we hope this will shed light on the expressivity of the technique languages used by the various systems.

In ISAPLANNER the algorithm is very simple, each technique is a function from a reasoning state to a sequence of reasoning states. This defines a search tree which can be explored using depth first, breadth first, iterative deepening, or best first. Additionally, search strategies can be set by techniques, overriding the general search strategy at a local level.

The planning algorithm used by λCLAM analyses the proof method and selects the first atomic method to apply. The algorithm is a function that interprets and applies the proof methods. Currently only depth first search is supported.

In ΩMEGA, the planning algorithm is more complicated and involves interpreting the strategy which in turn involves the control rules, and finally the proof methods. ΩMEGA uses a depth-first search strategy with respect to the results of control rules.

Work in Progress

We presented some basic components of the ΩMEGA, λCLAM and ISAPLANNER systems that we plan to compare using a common terminology. These proof planners have been developed without much interaction and largely independently. From these systems, we want to devise a unified view of proof planning systems.

We will study the three systems with respect to three criteria: comparing interface and architecture will be beneficial to the developers of the proof planning systems; comparing the power of each system will be of more interest to the users of the proof planning systems.

We plan to approach this comparative study from several levels. First, on the practical side, we will carry out a case study of a particular theory formation in all three proof planners (new domain, new methods, etc.). Second, on the theoretical level, we will assess what kind of efficiency measures can and cannot be implemented. Third, on the practical *and* theoretical side, we will look into what type of difficult and diverse theorems can and cannot be proved by each of the three proof planners.

References

[1] A. Bundy. The use of explicit plans to guide inductive proofs. In *9th CADE*, number 310 in LNCS, pages 111–120. Springer, 1988.

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

[3] A. Bundy, et al. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.

[4] L. Dixon and J.D. Fleuriot. IsaPlanner: A prototype proof planner in Isabelle. In *19th CADE*, number 2741 in LNCS, pages 279–283. Springer, 2003.

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

[6] L.C. Paulson. *Isabelle: A generic theorem prover*. Number 828 in LNCS. Springer, 1994.

[7] J.D.C. Richardson, A. Smaill, and I. Green. System description: proof planning in higher-order logic with lambdaclam. In *15th CADE*, number 1421 in LNAI, pages 129–133. Springer, 1998.

[8] J. Siekmann, et al. Proof development with ΩMEGA. In *18th CADE*, number 2392 in LNAI, pages 144–149. Springer, 2002.