The AI4FM approach for proof automation within formal methods

The AI4FM approach for proof automation within formal methods

A Grand Challenge 6 “Dependable Systems Evolution" project

Gudmund Grov
School of Informatics
University of Edinburgh
ggrov@staffmail.ed.ac.uk

Alan Bundy
School of Informatics
University of Edinburgh
bundy@staffmail.ed.ac.uk

Cliff B. Jones
School of Computing
University of Newcastle
cliff.jones@ncl.ac.uk

Andrew Ireland
School of Mathematical &
Computer Sciences
Heriot-Watt University
A.Ireland@hw.ac.uk

1. INTRODUCTION

We are about to embark on a project (AI4FM) that will use Artificial Intelligence (AI) to tackle a core issue for GC6 (Dependable Systems Evolution).

Achieving verified software has been a dream since the birth of computer science and the importance of this objective has become ever greater with the increasing size and complexity of software.[1] Grand Challenge 6 (GC6), “Dependable Systems Evolution”, addresses this by advocating the use of formal methods for software development.[2] The use of formal methods has been successful in safety-critical domains, like railway and aviation and is becoming increasingly popular in other sectors (e.g. Microsoft use formal methods to verify device drivers). A recent paper by Woodcock et al. [7] analyses a large number of recent industrial applications of formal methods.

Formal methods are applied *post facto* (bottom-up) or in *correctness by construction* (top-down). Top-down methods tend to be “posit and prove” where a designer posits a step of development and then seeks to justify it. Such justifications generate proof obligations (POs) — putative lemmas that need proof. Typically, a large proportion of POs can be discharged by automatic theorem provers, but there are still some that require user interaction. Discharging these POs can become a bottleneck in the use of formal methods in practical applications. There are two approaches to dealing with the POs that require user interaction:[3] (1) Follow a *modelling strategy*: change the model/abstraction to a strategy that simplifies the proofs, thus increasing the proportion of POs that are discharged automatically. (2) Follow a *proof strategy*: accept the challenging POs and define a strategy for discharging them.

It is the proof approach that we will take in our AI4FM project. Our aim is to increase the repertoire of techniques for the proof-strategy approach by learning from proof attempts made by humans.

The POs arising from formal methods tend to have different properties from “pure” mathematics. (1) There are often large numbers of detailed POs. To illustrate, the Paris Metro Line 14 and the Roissy Airport shuttle system were both developed using formal methods; the former generated $27,800$ POs (around $2,250$ interactive) while the latter generated $43,610$ POs (around $1,150$ interactive) [1]. (2) POs tend to be less deep, thus less proof effort is required from the user. (3) They are often “similar”, in the sense that they can be grouped into “families” — and the same (high-level) proof approach can be successfully applied to all members of the family.

2. THE AI4FM APPROACH

In many cases where a (correct) PO is not discharged automatically, an expert can easily see how to complete a proof. By exploring the nature of the POs within formal methods we believe that a higher degree of automation can be achieved by relying on expert intervention to do one proof, if this would enable a prover to discharge the others in the same family. Specifically, we hope to build a tool that will learn enough from one proof attempt to improve the chances of proving “similar” results automatically. By “proof attempt” we include things like the steps explored by the user (not just the chain of steps in the final proof). Thus it is central to our goal that we find *high-level* strategies capable of cutting down the search space in proofs. Our hypothesis is:

> *we believe that it is possible (to devise a high-level strategy language for proofs and) to extract strategies from successful hand proofs that will facilitate automatic proofs of related POs.*

[1] We use the term software although the discussion here is valid for any application of formal methods, i.e. to generic system modelling as well as software and hardware.

[2] ‘Formal methods’ use mathematics to specify, develop and reason about software and systems.

[3] Note that the modelling and proof strategies are not mutually exclusive. For example, the numbers quoted below most likely apply after several iterations of the original model. A proof strategy could still be applied after the modelling strategy has reduced the numbers of undischarged POs.

To achieve our goal we plan to use many *dimensions* to analyse the exemplar proof and the POs, e.g. by separating information about data structures and approaches to different patterns of POs. A proof (attempt) might be seen to use "generalise induction hypothesis" (e.g. adding an argument to accumulate values) in a specific proof about, say, sequences; a future use of the same PO might involve a more complicated tree data structure — but if it has an induction rule, the same strategy might work. We would also expect to discover other dimensions, such as the domain of the PO (e.g. does it relate to trains or to railway tracks?).

Designing a strategy language capable of capturing such properties (in an abstract form) is key to the success of AI4FM. Evidence for the possibility of such strategy language is *rippling* [2] – its generality can be illustrated by the domains to which it has been applied, e.g. verification of functional, logical and imperative programs; synthesis of theorems, programs and witnesses; correction of faulty specifications and hardware verification. We believe that several items from rippling will play a major part in the design of our strategy language:

- **Some "standard" proof plans and known deviations from and patches to them.** [2] uses rippling to describe a "standard" proof plan for inductive proofs and shows how each different pattern of failure in rippling suggests a different way of patching a failed proof attempt. We hypothesise that expert-provided proofs of undischarged POs will typically exhibit either a new proof plan or a new patch to an existing plan.

- **Choices of unusual induction rules and variables, choices of loop invariants.** Choosing an alternative non-standard induction rule is one of the patches to the standard induction proof plan which is described in [2]; patching a failed loop invariant is the patch described in [6].

- **Choices of intermediate lemmas.** Designing, constructing and proving a key intermediate lemma is another of the patches to the standard induction proof plan described in [2].

- **Generalisation of the PO.** [2] also describes ways of generalising the PO or the current goal to patch a failed proof.

3. CONCLUSION

AI4FM addresses proof automation for formal methods which is a bottleneck for their (industrial) application. The target of the project is to increase the overall proof automation within formal methods, thus increasing their applicability. Hence, it relates to any grand challenge where formal methods are, or can be,[4] applied. Thus, it is within the scope of the existing computing Grand Challenge 6: "Dependable Systems Evolution".

[4]Although formal methods have mainly been applied to safety-critical domains, it has been argued that they are in fact cheaper than conventional software development techniques. The argument is that bugs are detected earlier, thus reducing (much of) the need for testing and reworking.

Of the proposed societal grand challenges, reliable software is at least crucial for *security*. Moreover, the formality and mathematical rigour of formal methods could have a role to play when modelling and developing systems to support *assisted living* and *health*.[5]

Finally, note that the motivation for AI4FM has come from actual industrial needs (in particular from industrial partners of the EU-funded Deploy project[6]). Our longer term hope is that it can help to stimulate other AI ideas for formal methods. An example of this is [5], where AI is used to suggest changes to formal models from proof failures. For more details about the AI4FM project, please see `www.ai4fm.org` and [3, 4].

Acknowledgements

The initial work has been supported by the EPSRC Platform Grants *EP/E005713/1 The Integration and Interaction of Multiple Mathematical Reasoning Processes* and *EP/E035329/1 Trustworthy Ambient Systems (TrAmS)*. From April 2010, it will be supported by the EPSRC *(EP/H024050/1, EP/H023852/1, EP/H024204/1) AI4FM: using AI to aid automation of proof search in Formal Methods* grant.

4. REFERENCES

[1] J.-R. Abrial. Formal methods: Theory becoming practice. *Journal of Universal Computer Science*, 13(5):619–628, 2007.

[2] A. Bundy, D. Basin, D. Hutter, and A. Ireland. *Rippling: Meta-level Guidance for Mathematical Reasoning*, volume 56 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2005.

[3] A. Bundy, G. Grov, and C. B. Jones. Learning from experts to aid the automation of proof search. In L. O'Reilly and M. Roggenbach, editors, *AVoCS'09 – PreProceedings of the Ninth International Workshop on Automated Verification of Critical Systems*, Technical Report of Computer Science CSR-2-2009, pages 229–232. Swansea University, Wales, UK, 2009.

[4] A. Bundy, G. Grov, and C. B. Jones. An outline of a proposed system that learns from experts how to discharge proof obligations automatically. In *Proceedings of Dagstuhl Seminar 09381: Refinement Based Methods for the Construction of Dependable Systems*, 2009.

[5] A. Ireland, G. Grov, and M. Butler. Reasoned Modelling Critics: Turning Failed Proofs into Modelling Guidance. In *Proceedings of ABZ'10*, number 5977 in LNCS. Springer-Verlag, 2010.

[6] J. Stark and A. Ireland. Invariant discovery via failed proof attempts. In P. Flener, editor, *Logic-based Program Synthesis and Transformation*, number 1559 in LNCS, pages 271–288. Springer-Verlag, 1998.

[7] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal Methods: Practice and Experience. *ACM Computing Surveys*, 41(4), Oct 2009.

[5]If software is to be used to help people in their daily life, it should not make them do something that is potentially dangerous.

[6]See `www.deploy-project.eu/`