

Inferring intent from proof attempts: AI4FM project

Cliff Jones
Computing Science
Newcastle University



Contents

Intro

Examples

Where next?



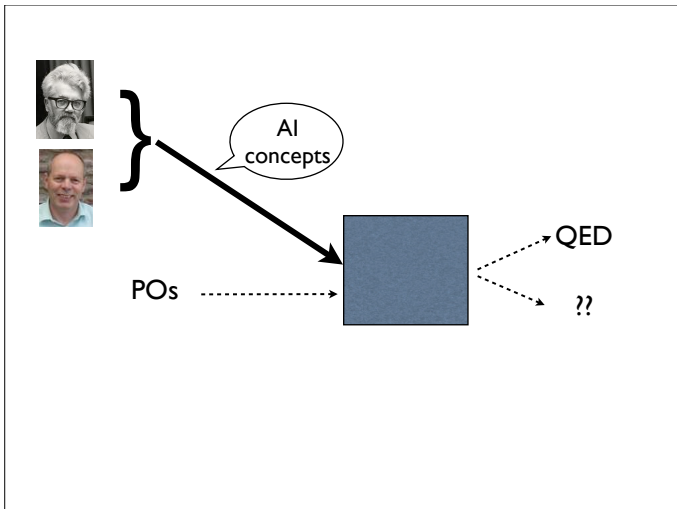
Observations

- (IMHO): “formal methods” pay off *during* development
 - *post facto* proof — not a smart way to debug
- (my belated) acceptance that tools are essential
- sadly, full formal reification rarely used in industry (possibly imaginary) data:
 - a development step generates 500 proof obligations
 - of which, 400 discharged automatically
 - by search, clever heuristics, theories (courtesy of AI)
 - discharging 100 POs by engineering staff is not acceptable
 - *but* 5 mini ideas cover the remaining proof tasks!
 - typically, these ideas are specific to the data structures etc.
- can we:
 - learn the “5 ideas” from an expert; apply to other 95 POs
 - accept that this will/may not help with “next 100”



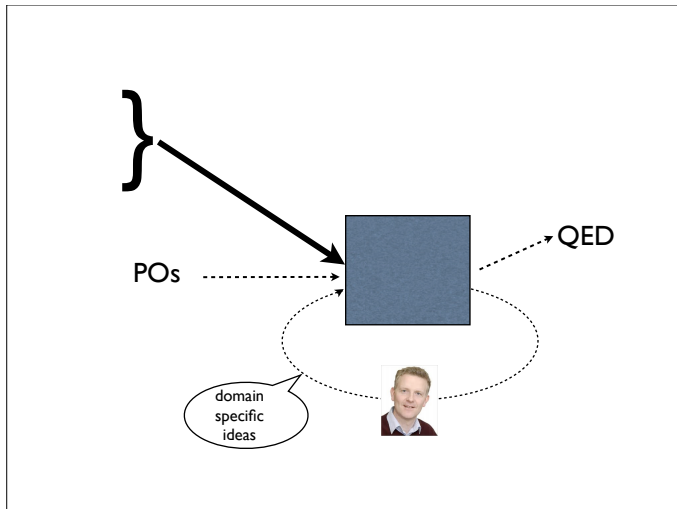
Approach

Show where “smarts” come in: traditional

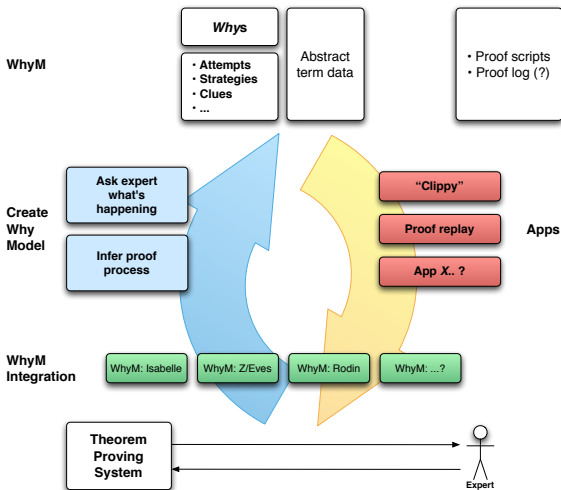


Approach

Show where “smarts” come in: AI4FM



Architecture (Leo/Andrius)



Notes on architecture (i)

- TP systems good at low-level search
 - domain of tactics/tacticals
 - unlikely to be the cause of failure
 - strengths differ in various TP systems
- what the expert sees
 - high-level strategy: “Why”
 - “missing” lemmas/side-conditions/generalisations
 - (we hope) higher level will be similar
- note: **mine proof “process”**
 - rather than finished proofs
- spotting what user is doing
 - “parse” + “clippy”
 - “matching” to spot what is missing
- indication that tactics are too low level:
 - work once
 - fail after minor change to definitions
 - cf. Mehta’s thesis / RodinTools



Notes on architecture (ii)

- use parallelism
 - at least as in RodinTools (parallel with user thoughts)
 - explore multiple strategies
 - control issues non-trivial?
- inclusion of (parallel) “disproving”
 - counter examples may have been underrated!
 - e.g. concatenation not commutative
 - vs. extra side-conditions
- *work in progress*: there now follow some examples:



Some common points

- *not* aiming at general maths proofs
 - precisely: **FM POs**
 - remember that POG is already a decomposition of an (unstated?) equivalence theorem
 - rarely require deep proofs
 - but they do vary from one application to another
- tailored lemmas are important (see below)
- role of “proof critics”



example: “adequacy” PO

(as in VDM data reification)

$$\forall a \in A \cdot \exists r \in R \cdot a = retr(r)$$

sometimes messy because of embedded existential quantifier

consider:

- application area
- POG (name)
- is type A of bounded size?
- know we'll need a witness for r

choose:

- choice function ($a = retr(choose(a))$)
- switch to *reln*



low level example: *Conjecture*

consider:

- application area
- POG (name)
- ...
- hyp/goal use operators from different *Theorys*

choose:

- definitions of operators
- lemmas relating operators
- map to different data type



example: (any) *Conjecture*

“matching” / shape

consider:

- application area
- POG (name)
- shape

choose:

- look for lemma
- form new lemma (cf. “shape” in similar proofs)
- generalise? (cf. “shape” in similar proofs)
- split (cut rule)? (cf. “shape” in similar proofs)



low-level example: *Conjecture*

consider:

- hyp/goal have operators in different orders

choose:

- normal form
- distribution lemmas
- commutative lemmas



example: record structure

consider:

- big?

choose:

- structure with fewer fields (cf. J's "toy problem")
- eliminate dependencies (*inv*)



example: on seeing a new *Theory*

consider:

- sibling theories

choose:

- lemmas “by analogy”
- (rarely!) morphisms?



example: induction

consider:

- ??

choose:

- standard
- “k-induction”
- complete
- lemma shape for multiple operators ($a \overset{\curvearrowright}{\rightarrow} b \overset{\curvearrowright}{\rightarrow} c$)



example: new *function* definition

consider:

- domain given

choose:

- prove where defined
- generate induction lemmas



example: *Term involving function*

(recursive) reference

$rev == \dots$

$rev(rev(s)) = s$

$len\ rev(s) = len\ s$

consider:

- sibling theories

choose:

- lemma relating $len(s1 \curvearrowright s2) = len\ s1 + len\ s2$



Handling failure

... can we reduce “failure” to analysis of generated *Conjectures*

consider:

- looks “broken”

choose:

- suspend?
- reduce weights



What we are doing *now*

- “how to say why” + “models of why”
- remember — at this stage of AI4FM:
 - think about how to record high-level proof strategies
 - the issue of how to learn them comes next
 - *weights* above are but a small nod to learning!
- understand role of “failure” in proof
 - I love the “proof critic” idea (Edinburgh)
 - ... but do wonder if tracking failure is too sequential?
- there are many other things that inhibit adoption of FMs:
 - tools themselves (mono-lingual)
 - no one universal method
 - “concept overload”



More information

- www.ai4fm.org
- open mailing list: ai4fm-info@jiscmail.ac.uk
- *mural* — see [JJLM91]
- Rippling — see [BBHI05]
- AI4FM itself — see [JGB10, FJ10]
- Dagstuhl event 2012-07-02/06





A. Bundy, D. Basin, D. Hutter, and A. Ireland.

Rippling: Meta-level Guidance for Mathematical Reasoning.

Cambridge University Press, 2005.



Leo Freitas and Cliff B. Jones.

Learning from an expert's proof: AI4FM.

In Tom Ball, Lenore Zuck, and Natarajan Shankar, editors, *UV10 (Usable Verification)*, November 2010.



Cliff B. Jones, Gudmund Grov, and Alan Bundy.

Some facets of a strategy language for proofs.

In *5th Automated Formal Methods workshop (AFM'10)*, July 2010.

Also available as Edinburgh University, School of Informatics technical report EDI-INF-RR-1377.



C. B. Jones, K. D. Jones, P. A. Lindsay, and R. Moore.

mural: A Formal Development Support System.

Springer-Verlag, 1991.

