A (very) small experiment in Event-B rippling

Gudmund Grov, Alan Bundy & Lucas Dixon

Rodin Workshop, Dusseldorf 2010
Talk outline

▶ “Abrial’s MMPE rule” : \( n \times x/100 \times f \times p \times 20 \)
  ▶ 100,000 loc \( \leadsto \) 3-12 Man Months of Proof Effort
▶ BUT, \( \exists \) families of Event-B UPOs based on proof strategy
  ▶ AI4FM tries to explore such families to increase automation.
▶ This requires high-level proof strategies
  ▶ rippling is an example of a high level proof strategy
  ▶ implemented in Isaplanner.
▶ In this talk we will:
  ▶ give an overview of the AI4FM project
  ▶ describe a simple Event-B experiments with rippling/Isaplanner
  ▶ beg for help!
AI4FM overview

- The user manually proves one exemplar proof.

\[ A \vdash B \]

- The theorem prover uses the additional information from the exemplar proof to discharge “similar” proofs:
The project

- 4 years UK EPSRC funding – started 1 April 2010
- The team
  - Newcastle
    - Cliff Jones, Leo Freitas, Andrius Velykis
  - Edinburgh
    - Alan Bundy, Gudmund Grov, Yuhui Lin
  - Heriot-Watt
    - Andrew Ireland
  - Southampton
    - Michael Butler.
Nature of Event-B POs/proofs

- Rarely deep.
  - we are *not* trying to prove real math theorems!
- Complexity reduced by layering abstractions.
- Lots of POs
  - ... which can often be grouped into “(proof) families”.
- Lots of detail (on larger examples)
The **AI4FM** process

- (Somehow) classify POs into families.
- Require the expert user to prove 1 PO manually/interactively
  - preferable the “simplest”.
- This extra information is used to discharge rest of families
- Requires abstracting proof into a higher-level strategy
  - ... and we must design a strategy language to capture this.
Towards a strategy language

- The strategy language needs to be robust to cope with changes.
- A strategy may describe a sequence of intermediate lemmas that could be spawned: prove lemmas $L_1, L_2, \cdots, L_n$ s.t. GOAL follows.
- Abstract over many “dimensions”
  - data-structure, domain, etc.
- Include notions like generalisation and lemma discovery.
- Rippling provides evidence for a high-level strategy language.
Rippling (in a hurry)

- Proof plans: high-level description of proofs
  - captures common patterns of reasoning
- Rippling: a proof plan, which
  - works when one of the givens can be embedded in the goal (e.g. inductive step cases)
    - for example in an Event-B INV type PO:
      \[
      \text{com} = \text{Cls} \triangleleft (cl; \text{nm}) \vdash \text{com} = \text{Cls} \triangleleft (\text{cl} \leftrightarrow \{ s \mapsto (\text{sze} \mapsto [])\}) \uparrow; \text{nm}
      \]
  - annotations to guide rewriting (towards given)
    - language of wave-fronts and skeletons
  - (direction) guarantees termination
Rippling illustration (step case of $t@(Y@Z) = (t@Y)@Z$)

Rewrite rules:

$$H#T \uparrow \Theta L \Rightarrow H#T\Theta L$$ \hspace{1cm} (1)

$$X_1#X_2 \uparrow = Y_1#Y_2 \uparrow \Rightarrow X_1 = Y_1 \land X_2 = Y_2 \uparrow$$ \hspace{1cm} (2)

Rippling proof:

$$t@Y@Z = (t@Y)@Z$$ \hspace{1cm} IH

$$h#t \uparrow@Y@Z = (h#t \uparrow@Y)@Z$$ \hspace{1cm} apply (1) 2 times

$$h#t@Y@Z \uparrow = h#(t@Y)@Z \uparrow$$ \hspace{1cm} apply (1)

$$h#t@Y@Z \uparrow = h#(t@Y)@Z \uparrow$$ \hspace{1cm} apply (2)

$$h = h \land t@Y@Z = (t@Y)@Z$$ \hspace{1cm} apply IH
IsaPlanner

- IsaPlanner is a proof planner built on top of Isabelle
  - enables use of existing Isabelle automation
  - soundness ensured by Isabelle.
- Reasoning techniques which generates proof plans
  - lazy search over possible ways to apply technique
- Implemented in the ML language
  - language also used to write new reasoning techniques
  - provides many ML functions to aid developing new techniques
- Rippling technique encoded (among others)
The experiment

- Goal is to check how rippling works in an Event-B setting
- Longer term we hope to see how easy it is to encode new techniques/patterns in Isaplanner.
- Set representation in Isabelle/HOL used
  - ... and slightly extended
- Modelled an example system in Rodin
  - translated POs into Isabelle/HOL representation
- Addressed INV type POs
Representing Event-B POs in Isabelle/HOL

- Event-B POs (+ theory) must be represented in Isabelle/HOL
- Uses Isabelle/HOL’s set theory
- Extended with “Event-B operators”
  - e.g. \(\triangleleft, \triangleright, \ll, \gg, \ldots\)
- Functions as relations (HOL functions are total)
  - \(\rightarrow, \leftrightarrow, \mapsto, \ldots\) defined
  - function application uses definite description operator (i.e. \(\iota\))
- Proof rules (theorems) derived by need
- Drawbacks
  - Most general type for functions
    - little mileage from HOL type checking
  - WDs ignored
  - Ignores Event-B proof rules
The system: a telephone system

- Based on Z model by Woodcock
- The variables
  - \( \text{call} \in \text{Subs} \leftrightarrow (\text{Status} \leftrightarrow \text{Subs}) \)
  - \( \text{connected} \in \text{Subs} \rightarrow \text{Subs} \)
  - \( \text{st} \) and \( \text{num} \) projections on \( (\text{Status} \leftrightarrow \text{Subs}) \)
  - \( \text{Free} \subseteq \text{Subs} \)
- Invariants
  - \( \text{inv1} : \text{Callers} = \text{dom}((\text{call}; \text{st}) \triangleright \text{Connected}) \)
  - \( \text{inv2} : \text{connected} = \text{Callers} \triangleleft (\text{call}; \text{num}) \)
- where \( \text{Connected} \subseteq \text{Status} \) for "connected calls".
The system: a telephone system (events)

- Event’s to lift handle, dial, answer, etc..
- We will only discuss two events
  - EVENT \( \text{LiftFree} \triangleq \)
    \[
    \text{ANY } s \text{ WHERE } s \in \text{Free} \quad \text{THEN } \text{Free} := \text{Free} \setminus \{s\} \\
    \text{call}(s) := (\text{seize} \mapsto \text{empty})
    \]
    ...

- EVENT \( \text{LiftSuspended} \triangleq \)
  \[
  \text{ANY } s \text{ WHERE } (s \mapsto \text{suspended}) \in \text{connected}^{-1}; \text{call}; \text{st} \\
  \text{THEN } \text{call}(\text{connected}^{-1}(s)) := (\text{speech} \mapsto s)
  \]
  ...

PO LiftFree/inv1/INV

- All required rewrite (wave) rules are existing
- Proof strategy
  - Rippling followed by application of IH (inv1)
  - Then manually discharge reminding (non-rippling) goals
    - i.e. the conditions from conditional rewrite rules
- PO LiftFree/inv1/INV:

  $Callers = dom\left( call; st \triangleright Connected \right), s \in Free \vdash$

  $Callers = dom\left( (call \leftrightarrow \{ s \mapsto (seize \leftrightarrow empty) \}); st \triangleright Connected \right)$

  (in one branch) gives the following two (provable) sub-goals

  $s \notin dom\left( call \right) \quad ((s \mapsto (seize \leftrightarrow [])); st) \triangleright Connected = {}$

  note there are also several (sometimes unprovable) branches
PO LiftFree/inv2/INV

- **PO LiftFree/inv2/INV:**

  \[
  \text{connected} = \text{Callers} \triangleleft (\text{call}; \text{num}), s \in \text{Free} \vdash \\
  \text{connected} = \text{Callers} \triangleleft ((\text{call} \leftrightarrow \{s \mapsto \text{(seize} \mapsto \text{empty})\}); \text{num})
  \]

- (in one branch) gives the following two (provable) sub-goals

  \[
  s \notin \text{dom}(\text{call}) \quad (\text{Callers} \triangleleft \{(s \mapsto \text{(seize} \mapsto [])\}); \text{num}) = \{
  \]

- The “idea” is similar to LiftFree/inv1/INV
  - “ripple out” to “isolate” the new part — and show it is \{
  - BUT the rules used, i.e. the proof, differs (e.g. \triangleright vs. \triangleleft)
  - AND one of the sub-goals is harder to prove
LiftSuspended POs

- **PO LiftSuspended/inv1/INV**

\[ \text{Callers} = \text{dom}(\text{call}; \text{st} \triangleright \text{Connected}), \]
\[ (s \mapsto \text{suspended}) \in \text{connected}^{-1}; \text{call}; \text{st} \vdash \]
\[ \text{Callers} = \text{dom}((\text{call} \iff \{ \text{connected}^{-1}(s) \mapsto (\text{speech} \mapsto s) \}); \text{st} \triangleright \text{Connected}) \]

- **PO LiftSuspended/inv2/INV**

\[ \text{connected} = \text{Callers} \triangleleft (\text{call}; \text{num}), \]
\[ (s \mapsto \text{suspended}) \in \text{connected}^{-1}; \text{call}; \text{st} \vdash \]
\[ \text{connected} = \text{Callers} \triangleleft ((\text{call} \iff \text{connected}^{-1}(s) \mapsto (\text{speech} \mapsto s)); \text{num}) \]

- Previous approach: \( s \) was not in \( \text{Callers} \), and is still not
- Here, the “dual” is true:
  - \( \text{connected}^{-1}(s) \) was in \( \text{Callers} \) and still is.
- Requires “theory” properties of \( \text{num}, \text{st}, \text{call} \) and \( \text{connected} \)
Some observations: difference with existing rippling work

- Hard to make any conclusions from a small case study
- Rippling seems promising for INV POs
- It manages to reduce the PO to a smaller, simpler goal – however:
  - much more conditional rewriting than other domains
  - .. and WD (which will add further conditions) ignored
  - not clear how to discharge these (non-rippling) goal
- Still lots of room for improvement:
  - several branches – use counter-example finders to filter out most obvious ones
  - better use of existing techniques (simp, blast, etc)
  - productive use of failure.
Productive use of failure

- One advantage of rippling is Ireland’s proof critics.
- Particular failures (or partial matches) of rippling triggers certain “exception cases”
  - e.g. a missing lemma is speculated, or the conjecture is generalised.
- More robust than lower-level tactics.
- These could be applied in the Event-B setting
  - but not clear how to prove discovered/generalised lemmas
    - induction+rippling traditionally used
    - ... this may not be the case in Event-B
  - we may need to discover new proof strategies here
  - ... or learn them from a given exemplar proof ...
Rippling experiment vs. AI4FM agenda

- AI4FM is about learning strategies from an exemplar proof
- Rippling is an existing strategy
- Maybe a general strategy for INV is to use rippling
- and the special purpose strategies learned from exemplar proofs are used to
  - discharge sub-goals from conditional rewrite rules in target proofs
  - .. and (internally) of source proof, use proof of one such sub-goal to discharge other
  - prove discovered lemmas.
The **AI4FM plan**

- **Analyse** a lot of
  - POs
    - preferably from real-world applications
  - ... their (expert-provided) proofs attempts
  - and analysing “families”.

- **Based on analysis, develop a strategy language**
  - more examples ⇒ more robustness
  - thus, the need for examples!
  - we expect an **iterative** development of the language.

- **Provide tool support**
  - to extract a strategy out of an exemplar proof
  - to interpret strategies to discharge “similar POs”.
We need your help (so we can help you)!

- We have experimented with our own little case-study.
- Steve Wright have already volunteered to participate!
- You can help by providing case-studies with (non-trivial) POs:
  - ... industrial-sized preferably
  - ideally, with (given) families of proofs
  - even better with with proof history (including dead-ends)
  - the full proof process can tell us more than the finished article

- Examples from difference sources will increase robustness
- ... which will (hopefully) give you **better proof automation**!
Summary

- AI4FM is a new project trying to learn and recycle proof strategies.
- Rippling is a high-level proof strategy
- We have shown promising results when rippling Event-B POs
  - a simple example
  - user-interaction still required
- (Your?) relevant example(s) will be of great help!
Thank you!

http://www.ai4fm.org