Planning and Patching Proofs

Alan Bundy & Lucas Dixon

School of Informatics
University of Edinburgh
Why Inductive Reasoning is Necessary

Proof by mathematical induction required for reasoning about repetition, e.g. in:

- recursive datatypes: numbers, lists, sets, trees, etc;
- iterative or recursive computer programs;
- electronic circuits with loops or parameterisation.

So needed for proof obligations in formal methods.
Proof Plans: What Are They?

- Attempt to capture common structure of family of proofs.
- Used to guide search for new proofs from same family.
- Three parts: tactic, method and critics.
  - Tactic is computer program for applying rules of inference.
  - Method is meta-logical specification of tactic.
  - Critic analyses failure and suggests patch.
- Use AI plan formation to construct special-purpose proof plan for conjecture using general-purpose sub-proof plans.
- Allows flexible application of heuristics.
- Understanding gained suggests extensions of heuristics.

Alan Bundy
A Strategy for Inductive Proof: ind_strat

**Preconditions:**

**Declarative:** Rippling must be possible in step cases.

**Procedural:** Look-ahead to choose induction rule that will permit rippling.
Special-Purpose Proof Plan

Commutativity of $+$:

- **ind_strat $s(y)$**
  - Base
  - Step
  - induction
    - $0=0+0$
    - $y=y+0 \rightarrow s(y)=s(y)+0$

- **ind_strat $s(x)$**
  - $x+y=y+x \rightarrow s(x+y)=y+s(x)$

- **ind_strat $s(y)$**
Empirical Success of Proof Planning

- Implemented in \textit{Clam/\lambda Clam}, \Omega mega and IsaPlanner proof planners.
- Successfully tested on a wide range domains: induction, analysis, residue classes, diagonalisation, summing series, equational reasoning, process algebras, transfinite ordinals, completeness proofs, ...
- Applied outwith mathematics: computer configuration and bridge.
- Solution of \textit{eureka} problems using critics, \textit{e.g.} lemma discovery and generalisation.
- Applications to software/hardware verification/synthesis/transformation. \textit{e.g.} verification of Gordon computer, construction of loop invariants via critics.
- \Omega mega linked to 3\textsuperscript{rd} party provers, CAS, constraint solvers, \textit{etc}.
- IsaPlanner part of Isabelle tactic-based theorem prover.
Need for Intermediate Lemmas

Conjecture:

\[ \forall l: list(\tau). \ rev(\ rev(l)) = l \]

Rewrite Rules:

\[ \text{rev}(\text{nil}) \Rightarrow \text{nil} \]
\[ \text{rev}(H :: T) \Rightarrow \text{rev}(T)@(H :: \text{nil}) \]

Step Case:

\[ \text{rev}(\text{rev}(t)) = t \vdash \text{rev}(\text{rev}(h :: t)) = h :: t \]
\[ \vdash \text{rev}(\text{rev}(t)@(h :: \text{nil})) = h :: t \]
\[ \text{blocked} \]
Introducing an Intermediate Lemma

Lemma Required:

\[ \text{rev}(X@Y) \Rightarrow \text{rev}(Y)@\text{rev}(X) \]

Cut Rule: introduces this:

Original: \[ \Gamma \vdash \text{rev}(\text{rev}(l)) = l \]

New:

\[ \Gamma, \text{rev}(X@Y) \Rightarrow \text{rev}(Y)@\text{rev}(X) \vdash \text{rev}(\text{rev}(l)) = l \]

Justification: \[ \Gamma \vdash \text{rev}(X@Y) \Rightarrow \text{rev}(Y)@\text{rev}(X) \]

Heuristics needed: to speculate lemma.
Step Case Unblocked

\[ \text{rev}(\text{rev}(t)) = t \vdash \text{rev}(\text{rev}(h :: t)) = h :: t \]
\[ \vdash \text{rev}(\text{rev}(t)@(h :: \text{nil})) = h :: t \]
\[ \vdash \text{rev}(h :: \text{nil}) \odot \text{rev}(\text{rev}(t)) = h :: t \quad \text{lemma applied} \]
\[ \vdash (\text{rev}(\text{nil})@(h :: \text{nil})) \odot \text{rev}(\text{rev}(t)) = h :: t \]
\[ \vdash (\text{nil}@ (h :: \text{nil})) \odot \text{rev}(\text{rev}(t)) = h :: t \]
\[ \vdash (h :: \text{nil}) \odot \text{rev}(\text{rev}(t)) = h :: t \]
\[ \vdash h :: (\text{nil}@\text{rev}(\text{rev}(t))) = h :: t \]
\[ \vdash h :: \text{rev}(\text{rev}(t)) = h :: t \]
\[ \vdash h = h \land \text{rev}(\text{rev}(t)) = t \]

fertilization now possible.
Rippling in the Step Case

\[ t@Y@Z = (t@Y)@Z \]

\[ \vdash h :: t \uparrow @y@z = (h :: t \uparrow @y)@z \]

\[ \vdash h :: t@y@z \uparrow = h :: t@y \uparrow \quad \uparrow \]

\[ \vdash h :: (t@y)@z \uparrow \]

\[ \vdash h = h \land t@y@z = (t@y)@z \uparrow \]

- changing bits in **orange boxes** (wave-fronts).
- unchanging bits in **red** (skeleton).
Wave-Rules

\[
\begin{align*}
H :: T \uparrow \circlearrowleft L & \Rightarrow H :: T \circlearrowleft L \\
\text{rev}(H :: T \uparrow ) & \Rightarrow \text{rev}(T) \circlearrowleft (H :: \text{nil}) \\
X_1 :: X_2 \uparrow & = Y_1 :: Y_2 \uparrow \Rightarrow X_1 = Y_1 \land X_2 = Y_2 \\
X \circlearrowleft (Y \circlearrowleft Z) \uparrow & \Rightarrow (X \circlearrowleft Y) \circlearrowleft Z
\end{align*}
\]

- note preservation of skeleton and
- outward movement of wave-fronts.
- Can be annotated statically or dynamically.
Rippling Sideways and In

\[ rev(t)@L = qrev(t, L) \]

\[ \vdash rev(h :: t^\uparrow)@l = qrev(h :: t^\uparrow, [l]) \]

\[ \vdash (rev(t)@(h :: nil)^\uparrow)@l = qrev(t, h :: [l]^\downarrow) \]

\[ \vdash rev(t)@(h :: nil)[l]^\downarrow = qrev(t, [h :: l]) \]

\[ \vdash rev(t)@[h :: nil][l] = qrev(t, [h :: l]) \]

\[ \vdash rev(t)@[h :: l] = qrev(t, [h :: l]) \]

- Fertilization unifies \( L \) with \( [h :: l] \).
- \( [\textit{Sinks}] \) provide alternative wave-front destination, available when free variables are in hypothesis.
- Wave-fronts have directions: \( out^\uparrow / in^\downarrow \).
- Note that sinks and wave-fronts may need to be simplified, but this is skeleton preserving.
Sideways and Inwards Wave-rules

\[ qrev( H :: T \uparrow, L ) \Rightarrow qrev( T, H :: L \downarrow ) \]

\[ H :: T@L \downarrow \Rightarrow H :: T \downarrow @L \]

\[ ( X@Y \uparrow )@Z \Rightarrow X@ ( Y@Z \downarrow ) \]

- Note that some equations can be annotated in both directions.

\[ H :: T \uparrow @L \Rightarrow H :: T@L \uparrow \]

\[ H :: T@L \uparrow \Rightarrow H :: T@L \uparrow \]

\[ X@ ( Y@Z \uparrow ) \Rightarrow ( X@Y \downarrow )@Z \]
Preconditions of the Wave Method

1. The induction conclusion contains a wave-front,
   \[ \ldots = qrev(h :: t^{\uparrow}, [l]) \].

2. A wave-rule applies to this wave-front.
   \[ qrev(H :: T^{\uparrow}, L) \Rightarrow qrev(T, H :: L^{\downarrow}) \].

3. Any condition is provable.
   \[ X \neq H \rightarrow X \in H :: T^{\uparrow} \Rightarrow X \in T \].

4. Inserted inwards wave-fronts contain a sink or an outwards wave-front.
   \[ \ldots = qrev(t, h :: [l]^{\downarrow}) \].
Advantages of Rippling

**Selective:** not exhaustive rewriting.
- skeleton preserving and measure decreasing.

**Bi-directional:** rewriting.
- different annotations in each direction.

**Termination:** of any set of wave-rules,
- despite bi-directionality.

**Heuristic basis:** for choosing lemmas, generalisations, case splits and inductions.
(Lack of) Cut Elimination

Gentzen’s Cut Rule:

\[ A, \Gamma \vdash \Delta, \quad \Gamma \vdash A \]

\[ \Gamma \vdash \Delta \]

lacks subformula property.

Cut Elimination Theorem:

Gentzen showed Cut Rule redundant in FOL.
Kreisel showed necessary in inductive theories.

Practical Consequences:

Need to generalise conjectures.
Need to introduce lemmas.
Ripple-Based Heuristics

Induction Rules: choose induction which best supports rippling.

Lemmas: design wave-rule to unblock ripple.

Generalisation: generalise goal to allow wave-rule to apply.
Lemma Calculation 1

- Conjecture:

\[ \forall k, l: \text{list}(\tau). \ rev(k@l) = rev(l)@rev(k) \]

- Wave-Rules:

\[
\begin{align*}
H :: T \uparrow @L & \Rightarrow H :: T@L \\
rev( H :: T \uparrow ) & \Rightarrow rev(T)@L \uparrow \\
\end{align*}
\]

- Rippling in Step Case:

\[
\begin{align*}
rev( h :: t \uparrow @l ) & = rev(l)@rev( h :: t \uparrow ) \\
rev( h :: t@l ) & = rev(l)@rev(t)@L \uparrow \\
\end{align*}
\]

\[
\begin{align*}
rev(t@l)@L \uparrow (h :: nil) & \Rightarrow \text{blocked} \\
rev(l)@rev(t)@L \uparrow (h :: nil) & \Rightarrow \text{blocked}
\end{align*}
\]
Lemma Calculation 2

- **Induction Hypothesis:**
  
  \[ \text{rev}(t \cdot l) = \text{rev}(l) \cdot \text{rev}(t) \]

- **Blocked Step Case:**
  
  \[ \text{rev}(t \cdot l) \cdot (h :: \text{nil}) = \text{rev}(l) \cdot (\text{rev}(t) \cdot (h :: \text{nil})) \]

- **After Weak Fertilization:**
  
  \[ (\text{rev}(l) \cdot \text{rev}(t)) \cdot (h :: \text{nil}) = \text{rev}(l) \cdot (\text{rev}(t) \cdot (h :: \text{nil})) \]

- **Generalise Common Subterm:**
  
  \[ (u \cdot v) \cdot w = u \cdot (v \cdot w) \]

  i.e., we have calculated that we need associative of \( \cdot \) as a lemma.
Critic: Lemma Speculation

- Conjecture:
  $$\forall l:list(\tau). \ rev(\ rev(l)) = l$$

- Wave-Rule:
  $$\text{rev}(H :: T^\uparrow) \Rightarrow \text{rev}(T) @ H :: \text{nil}^\uparrow$$

- Induction Conclusion:
  $$\text{rev}(\text{rev}(h :: t^\uparrow)) = h :: t^\uparrow$$
  $$\text{rev}(\text{rev}(t) @ (h :: \text{nil})^\uparrow) = h :: t^\uparrow$$
  blocked

- Pattern Sought:
  $$\text{rev}(X @ Y^\uparrow) \Rightarrow F(\text{rev}(X), X, Y)^\uparrow$$

- Lemma Discovered:
  $$\text{rev}(X @ Y^\uparrow) \Rightarrow \text{rev}(Y) @ \text{rev}(X)^\uparrow$$
Failure of Ripple Preconditions

• Precondition 1 is true:
  1. The induction conclusion contains a wave-front.

    \[ \text{rev}( \text{rev}(t) @(h :: \text{nil}) ) = h :: t \] (in fact, two)

• Precondition 2 is false:
  2. A wave-rule applies to this wave-front.
     (to neither of them)

• Preconditions 3 and 4 are inapplicable.
  3. Any condition is provable.
  4. Inserted inwards wave-fronts contain a sink or an outwards wave-front.
Overview of Lemma Speculation Critic

Critic Invocation:  ⇒  Critic Applied:

Ind_Strat 1

Induction

Base

Step

Ripple

Wave Blocked

Lemma Critic

Cut

Ind_Strat 1

Induction

Base

Step

Ripple

Wave Unblocked

Lemma
Rippling Failure: Missing Sink

Conjecture:

$$\forall l : \text{list}(\tau). \ rev(l) = qrev(l, \text{nil})$$

Wave-Rules:

$$\text{rev}(H :: T \uparrow) \Rightarrow \text{rev}(T)\@H :: \text{nil} \uparrow$$

$$\text{qrev}(H :: T \uparrow, L) \Rightarrow \text{qrev}(T, H :: L \downarrow)$$

Induction Conclusion:

$$\text{rev}(h :: t \uparrow) = \text{qrev}(h :: t \uparrow, \text{nil})$$

$$\text{rev}(t)\@h :: \text{nil} \uparrow = \text{qrev}(h :: t \uparrow, \text{nil})$$

missing sink
Failure of Ripple Preconditions

• Preconditions 1, 2 and 3 are true:
  1. The induction conclusion contains a wave-front.

\[ \ldots = \text{qrev}(h :: t^{\uparrow}, \text{nil}) \]

  2. A wave-rule applies to this wave-front.

\[ \text{qrev}(H :: T^{\uparrow}, L) \Rightarrow \text{qrev}(T, H :: L^{\downarrow}) \]

  3. Any condition is provable — trivially, no condition.

• Precondition 4 is false:
  4. Inserted inwards wave-fronts contain a sink or an outwards wave-front.

\[ \ldots = \text{qrev}(t, h :: \text{nil}^{\downarrow}) \]
Overview of Generalisation Critic

Critic Invocation: $\Rightarrow$ Critic Applied:
Original Conjecture:
\[ \forall t : \text{list}(\tau). \ rev(t) = qrev(t, nil) \]

Disallowed Ripple:
\[ \ldots = qrev(t, h :: \text{nil}) \]

Schematic Conjecture:
\[ \forall t : \text{list}(\tau). \forall l : \text{list}(\tau). F(\rev(t), l) = qrev(t, G(l)) \]

Induction Hypothesis:
\[ F(\rev(t), L) = qrev(t, G(L)) \]

where \( F, G \) and \( L \) are meta-variables.
New Step Case:

\[
F(\text{rev}(h :: t), [l]) = \text{qrev}(h :: t, G([l]))
\]

\[
F(\text{rev}(t) @ h :: \text{nil}, [l]) = \text{qrev}(t, h :: G([l]))
\]

\[
\text{rev}(t) @ (h :: \text{nil} @ F'(\text{rev}(t) @ (h :: \text{nil}), [l]))
\]

\[
= \text{qrev}(t, h :: G([l]))
\]

\[
\text{rev}(t) @ (F'(\text{rev}(t) @ (h :: \text{nil}), [l]))
\]

\[
= \text{qrev}(t, h :: G([l]))
\]

\[
\text{rev}(t) @ ([h :: l]) = \text{qrev}(t, [h :: l])
\]

where \( F = @, F' = \lambda X. \lambda Y. Y \) and \( G = \lambda X. X \).

**Key Wave-Rule:** \( (X @ Y) @ Z \Rightarrow X @ (Y @ Z) \)

**Generalised Conjecture:** \( \forall t: \text{list}(\tau). \forall l: \text{list}(\tau). \text{rev}(t) @ l = \text{qrev}(t, l) \)
Pattern of Failure Suggests Patch

<table>
<thead>
<tr>
<th></th>
<th>PC 1</th>
<th>PC 2</th>
<th>PC 3</th>
<th>PC 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalization</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Case Split</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Induction Revision</td>
<td>√</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lemma Discovery</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

√ = success  ? = partial success  × = failure
Advantages of Proof Planning

• **Reduction** in search: larger steps, fewer options.

• Multi-level proof **explanation**: supports interaction.

• Non-standard proof **exploration**,
  least-commitment devices: meta-variables and constraints.
  agent-based proof planning.

• Framework for **inter-operating** reasoners.
Disadvantages of Proof Planning

- Loss of *completeness*: limited to methods and critics.
- Lack of *serendipity*: only anticipated patterns.
- **Hard work** to discover new proof plans,
  need to invent new concepts, *e.g.* wave-fronts,
  which are then a barrier to human understanding.
- Danger of *over-tuning*:
  a proof plan for every proof.
Summary

- Negative theoretical results create special search problems.
- These problems common in practice:
  induction rule choice, lemmas & generalisations.
- Proof plan for induction based on rippling.
- Rippling: selective; bidirectional; terminating and offers heuristic solution to special problems.
- Ripple breakdowns suggest: induction revision; lemma speculation or generalisation.
  Different patterns of proof breakdown suggest different patches.
- Implemented via proof planning with critics.