



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.



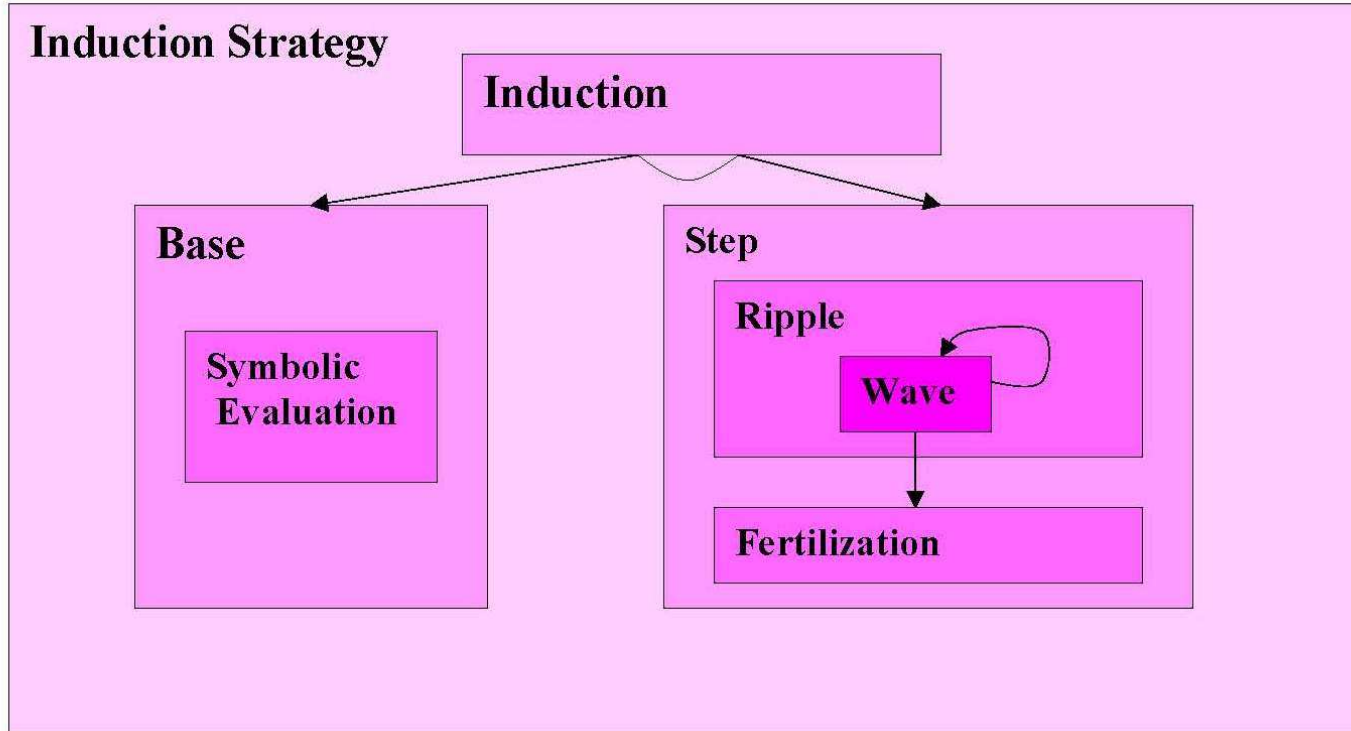
Proof Plans: What Are They?

- 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.



General-Purpose Proof Plans

A Strategy for Inductive Proof: `ind_strat`



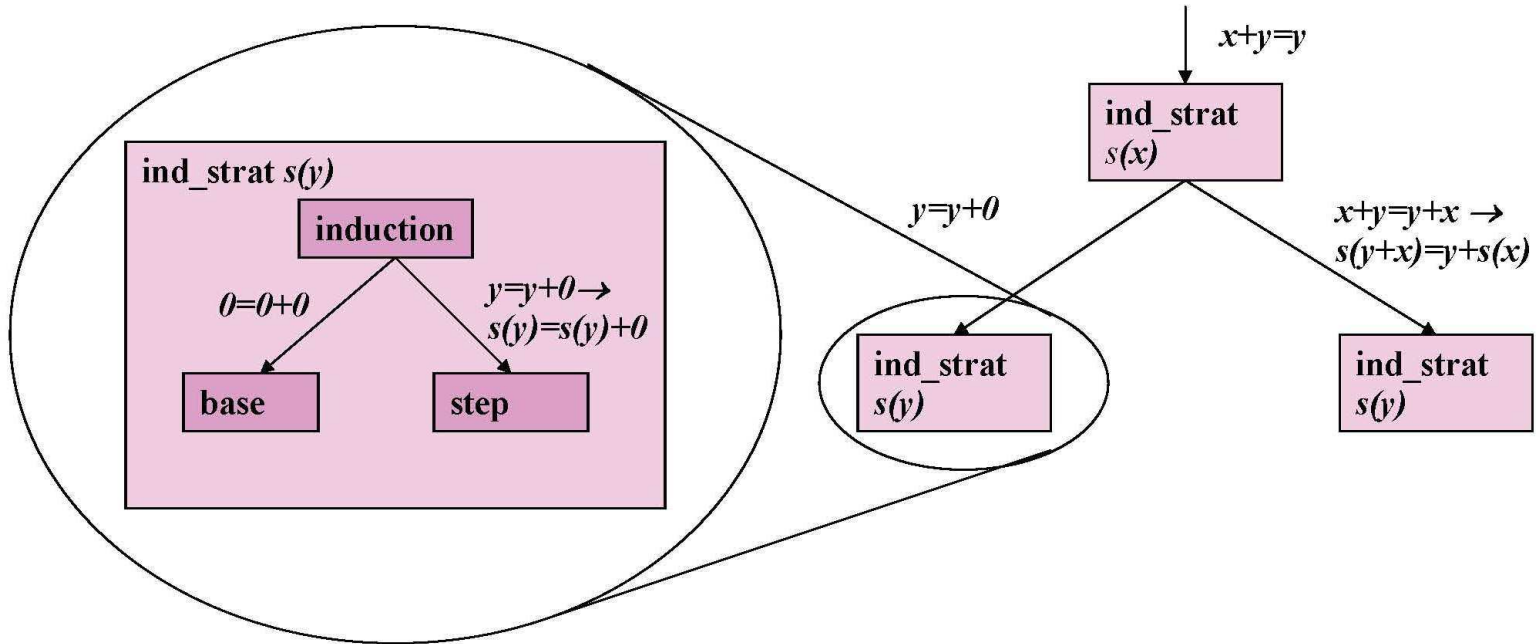
Preconditions:

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 +:



Empirical Success of Proof Planning

- **Implemented** in *Clam*/ λ *Clam*, Ω 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 **eureka** problems using **critics**,
e.g. lemma discovery and generalisation.
- **Applications** to software/hardware verification/synthesis/transformation.
e.g. verification of Gordon computer, construction of loop invariants via critics.
- Ω mega linked to **3rd party provers**, CAS, constraint solvers, *etc.*
- IsaPlanner part of Isabelle tactic-based theorem prover.



Need for Intermediate Lemmas

Conjecture:

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

Rewrite Rules:

$$\text{rev}(\text{nil}) \Rightarrow \text{nil}$$

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

Step Case:

$$\begin{aligned} \text{rev}(\text{rev}(t)) = t \vdash \text{rev}(\text{rev}(h :: t)) &= h :: t \\ \vdash \underbrace{\text{rev}(\text{rev}(t)@(h :: \text{nil}))}_{\text{blocked}} &= h :: t \end{aligned}$$

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

$$rev(rev(t)) = t \vdash rev(\mathbf{rev}(h :: t)) = h :: t$$

$$\vdash \mathbf{rev}(rev(t) @ (h :: nil)) = h :: t$$

$$\vdash \mathbf{rev}(h :: nil) @ rev(rev(t)) = h :: t \quad \textit{lemma applied}$$

$$\vdash (\mathbf{rev}(nil) @ (h :: nil)) @ rev(rev(t)) = h :: t$$

$$\vdash (\mathbf{nil} @ (h :: nil)) @ rev(rev(t)) = h :: t$$

$$\vdash (\mathbf{h} :: nil) @ rev(rev(t)) = h :: t$$

$$\vdash h :: (\mathbf{nil} @ rev(rev(t))) = h :: t$$

$$\vdash \mathbf{h} :: rev(rev(t)) = h :: t$$

$$\vdash h = h \wedge \mathbf{rev}(rev(t)) = t$$

fertilization now possible.



Rippling in the Step Case

$$t@(Y@Z) = (t@Y)@Z$$

$$\vdash h :: t^\uparrow @(y@z) = (h :: t^\uparrow @y)@z$$

$$\vdash h :: t@(y@z)^\uparrow = h :: t@y^\uparrow @z$$

$$\vdash h :: t@(y@z)^\uparrow = h :: (t@y)@z^\uparrow$$

$$\vdash h = h \wedge t@(y@z)^\uparrow = (t@y)@z^\uparrow$$

- changing bits in *orange boxes*[↑] (wave-fronts).
- unchanging bits in *red* (skeleton).



Wave-Rules

$$H :: T \uparrow @L \Rightarrow H :: T@L \uparrow$$

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

$$X_1 :: X_2 \uparrow = Y_1 :: Y_2 \uparrow \Rightarrow X_1 = Y_1 \wedge X_2 = Y_2 \uparrow$$

$$X@(Y@Z) \uparrow \Rightarrow (X@Y)@Z \uparrow$$

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



Rippling Sideways and In

$$\mathit{rev}(t)@L = \mathit{qrev}(t, L)$$

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

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

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

$$\vdash \mathit{rev}(t)@(\lfloor h :: \mathit{nil} \rfloor @ l) = \mathit{qrev}(t, [h :: l])$$

$$\vdash \mathit{rev}(t)@(\lfloor h :: l \rfloor) = \mathit{qrev}(t, [h :: l])$$

- Fertilization unifies L with $[h :: l]$.
- $\lfloor \mathit{Sinks} \rfloor$ 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.

$$H :: T^{\uparrow} @L \Rightarrow H :: T@L^{\uparrow}$$

$$H :: T@L^{\uparrow} \Rightarrow H :: T^{\uparrow} @L$$

$$X@(Y@Z^{\uparrow}) \Rightarrow (X@Y^{\downarrow})@Z$$



Preconditions of the Wave Method

1. The induction conclusion contains a wave-front,

e.g. $\dots = qrev(h :: t^\uparrow, [l]).$

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

e.g. $qrev(H :: T^\uparrow, L) \Rightarrow qrev(T, H :: L^\downarrow).$

3. Any condition is provable.

e.g. $X \neq H \rightarrow X \in H :: T^\uparrow \Rightarrow X \in T.$

4. Inserted inwards wave-fronts contain a sink or an outwards wave-front.

e.g. $\dots = qrev(t, h :: [l]^\downarrow).$



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:

$$\frac{A, \Gamma \vdash \Delta, \quad \Gamma \vdash A}{\Gamma \vdash \Delta}$$

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:

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

- Wave-Rules:

$$H :: T \uparrow @L \Rightarrow H :: T@L \uparrow$$

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

- Rippling in Step Case:

$$\text{rev}(h :: t \uparrow @l) = \text{rev}(l)@\text{rev}(h :: t \uparrow)$$

$$\text{rev}(h :: t@l \uparrow) = \underbrace{\text{rev}(l)@(\text{rev}(t)@(h :: \text{nil}) \uparrow)}_{\text{blocked}}$$

$$\text{rev}(t@l)@(h :: \text{nil}) \uparrow = \underbrace{\text{rev}(l)@(\text{rev}(t)@(h :: \text{nil}) \uparrow)}_{\text{blocked}}$$



Lemma Calculation 2

- **Induction Hypothesis:**

$$\mathit{rev}(t@l) = \mathit{rev}(l)@\mathit{rev}(t)$$

- **Blocked Step Case:**

$$\mathit{rev}(t@l)@(h :: \mathit{nil}) = \mathit{rev}(l)@(\mathit{rev}(t)@(h :: \mathit{nil}))$$

- **After Weak Fertilization:**

$$(\mathit{rev}(l)@\mathit{rev}(t))@(h :: \mathit{nil}) = \mathit{rev}(l)@(\mathit{rev}(t)@(h :: \mathit{nil}))$$

- **Generalise Common Subterm:**

$$(u@v)@w = u@(v@w)$$

i.e., we have calculated that we need associative of @ as a lemma.



Critic: Lemma Speculation

- Conjecture:

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

- Wave-Rule:

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

- Induction Conclusion:

$$\begin{aligned} \text{rev}(\text{rev}(h :: t^\uparrow)) &= h :: t^\uparrow \\ \underbrace{\text{rev}(\text{rev}(t) @ (h :: \text{nil})^\uparrow)}_{\text{blocked}} &= h :: t^\uparrow \end{aligned}$$

- 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.

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

(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.

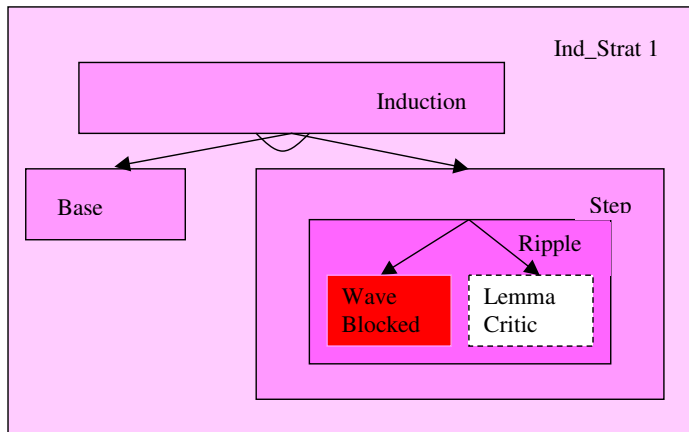
3. Any condition is provable.

4. Inserted inwards wave-fronts contain a sink or an outwards wave-front.



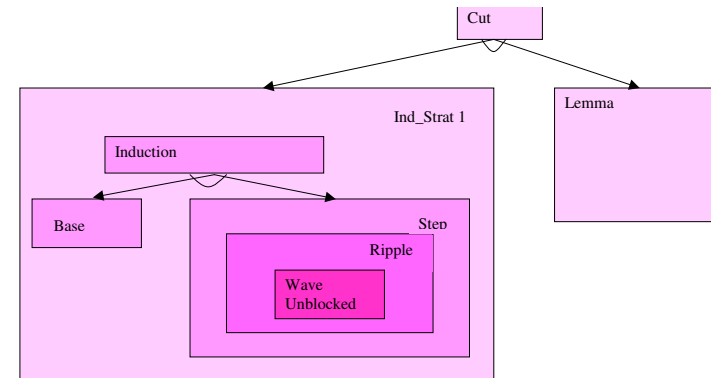
Overview of Lemma Speculation Critic

Critic Invocation:



⇒

Critic Applied:



Rippling Failure: Missing Sink

Conjecture:

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

Wave-Rules:

$$rev(H :: T^\uparrow) \Rightarrow rev(T) @ H :: nil^\uparrow$$

$$qrev(H :: T^\uparrow, L) \Rightarrow qrev(T, H :: L^\downarrow)$$

Induction Conclusion:

$$rev(h :: t^\uparrow) = qrev(h :: t^\uparrow, nil)$$

$$rev(t) @ h :: nil^\uparrow = \underbrace{qrev(h :: t^\uparrow, nil)}_{\text{missing sink}}$$



Failure of Ripple Preconditions

- Preconditions 1, 2 and 3 are **true**:

1. The induction conclusion contains a wave-front.

$$\dots = \mathit{qrev}(h :: t^\uparrow, \mathit{nil})$$

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

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

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

- Precondition 4 is **false**:

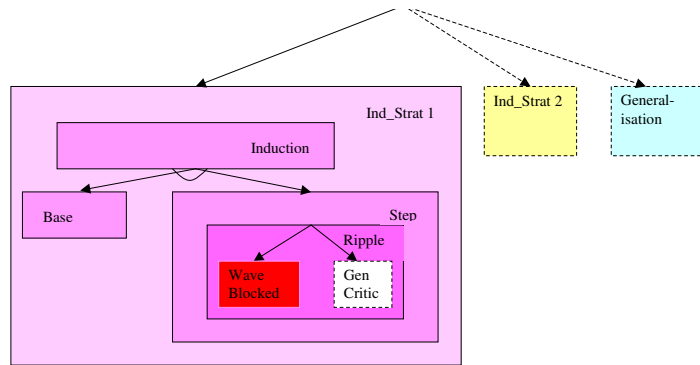
4. Inserted inwards wave-fronts contain a sink or an outwards wave-front.

$$\dots = \mathit{qrev}(t, h :: \mathit{nil}^\downarrow)$$



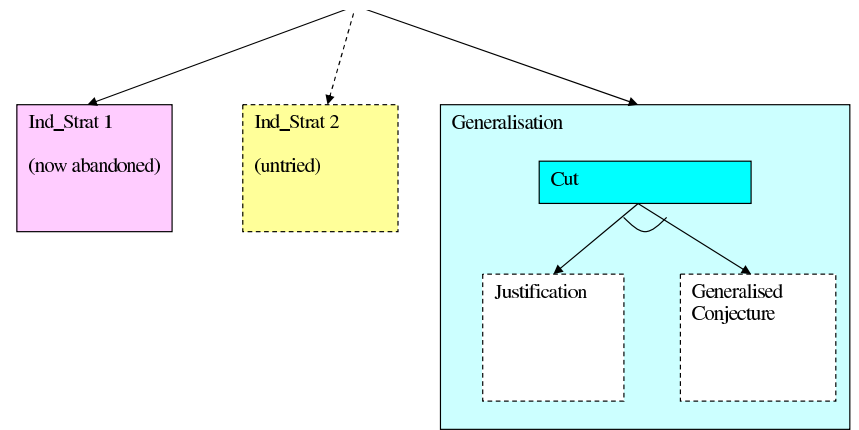
Overview of Generalisation Critic

Critic Invocation:



⇒

Critic Applied:



Patch: Sink Speculation

Original Conjecture:

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

Disallowed Ripple:

$$\dots = \text{qrev}(t, h :: \text{nil}^\downarrow)$$

Schematic Conjecture:

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

Induction Hypothesis:

$$F(\text{rev}(t), L) = \text{qrev}(t, G(L))$$

where F , G and L are meta-variables.



Patch: Instantiating the Meta-Variables

New Step Case:

$$\begin{aligned}
F(\text{rev}(h :: t^\uparrow), [l]) &= \text{qrev}(h :: t^\uparrow, G([l])) \\
F(\text{rev}(t)@h :: \text{nil}^\uparrow, [l]) &= \text{qrev}(t, h :: G([l])^\downarrow) \\
\text{rev}(t)@(h :: \text{nil} @ F'(\text{rev}(t)@(h :: \text{nil})^\uparrow, [l])^\downarrow) & \\
&= \text{qrev}(t, h :: G([l])^\downarrow) \\
\text{rev}(t)@(h :: F'(\text{rev}(t)@(h :: \text{nil})^\uparrow, [l])^\downarrow) & \\
&= \text{qrev}(t, h :: G([l])^\downarrow) \\
\text{rev}(t)@([h :: l]) &= \text{qrev}(t, [h :: l])
\end{aligned}$$

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

Key Wave-Rule: $(X@Y^\uparrow)@Z \Rightarrow X@(Y@Z)^\downarrow$

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

	PC 1	PC 2	PC 3	PC 4
Generalization	✓	✓	✓	×
Case Split	✓	✓	×	
Induction Revision	✓	?		
Lemma Discovery	✓	×		

✓ = 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.

