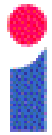
Towards a Strategy Language for Describing Proofs

Alan Bundy

School of informatics

University of Edinburgh

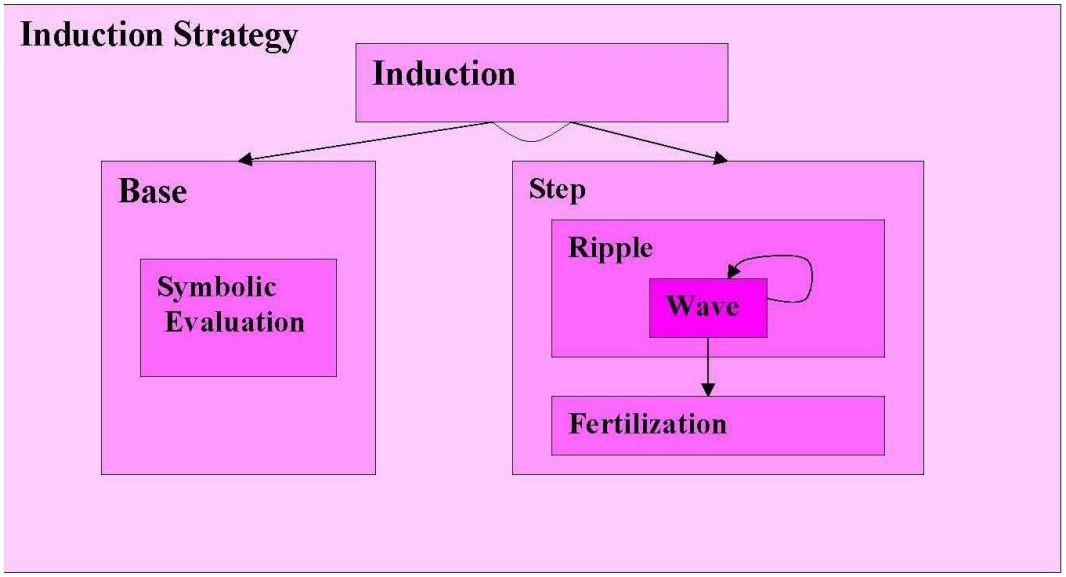
AI4FM Kick-off Meeting



Outline

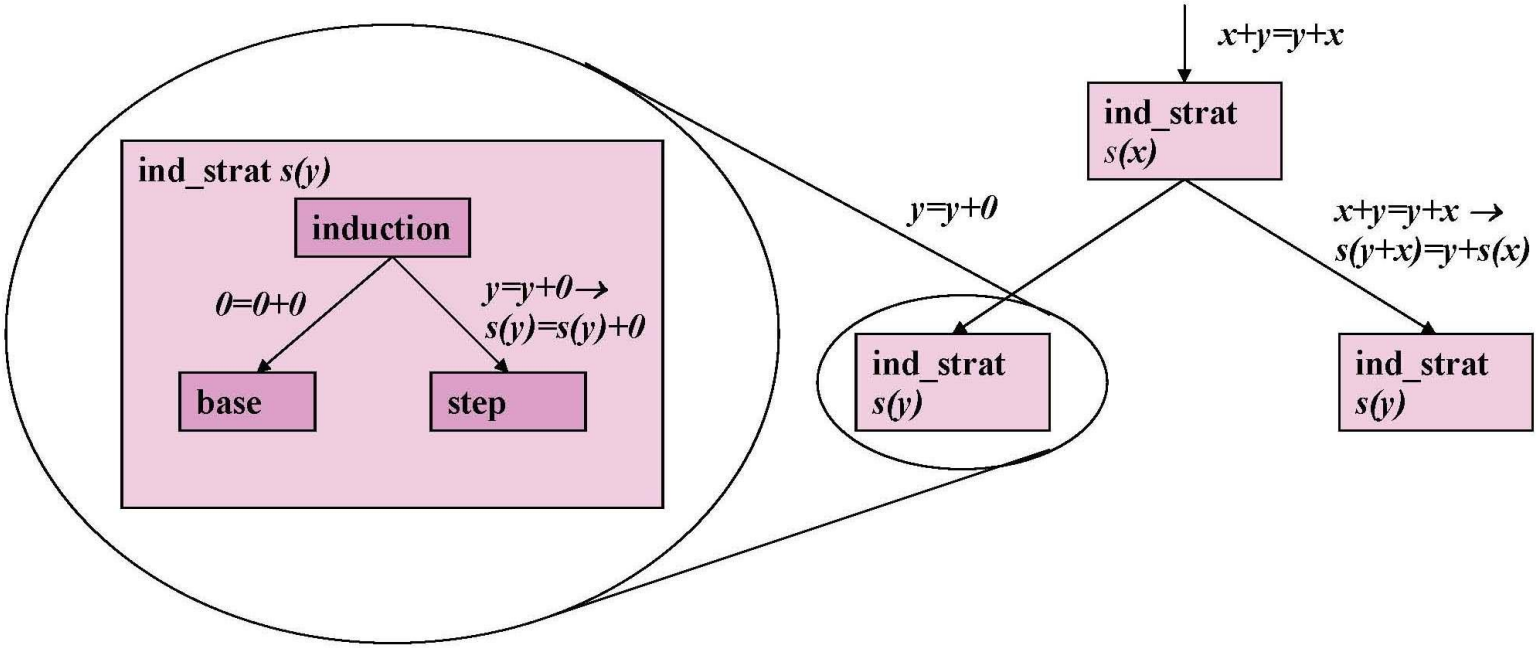
- Introduction to hiproofs
- Hiproofs as candidate strategy language
- Can describe search spaces too
- Consider alternative 'ACL2' methodology

What are Hiproofs?



- **Higraphs**: graphs whose nodes are higraphs.
- **Hiproofs**: use of higraphs to represent proofs.
  - http://dream.inf.ed.ac.uk/projects/higraphs/
- **Nodes** represent tactics; **arcs** pass subgoals.

Represent Proof at Different Levels of Detail



Zoom in and out of proof

Motivation

- Can describe transferable, high-level structure of proofs.
- Can parse source proof to level of detail permitted by known tactics.
  - 'black box' unparsable sub-proofs.
- Can generate target proof to varying levels of detail.
  - If low-level fails, try higher-level.
- Can describe partial proofs and search spaces too.

History

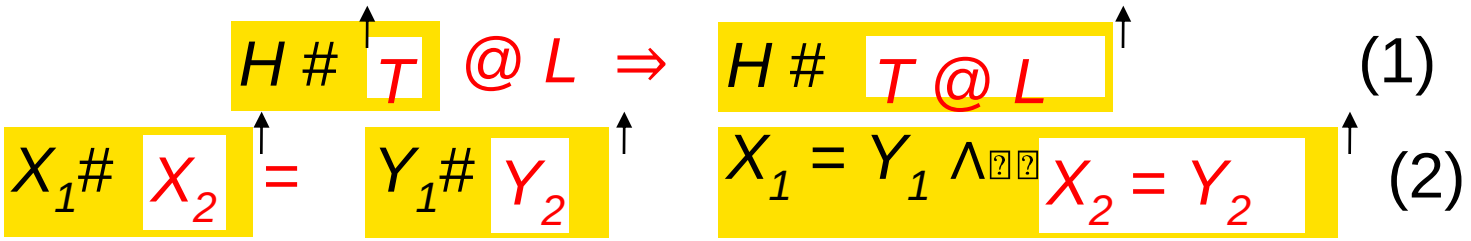
- Bundy & Ireland used box diagrams informally to describe proof plans.
- Denney, Power and Tourlas developed hiproof theory.
- Various hiproof viewers developed for Dixon's IsaPlanner.
- Aspinall, Denney & Lüth developed Hitac language and semantics.
- Plan to use hiproof proof constructor.

Bluffer's Guide to Rippling

- Use when 'given' embeds in 'goal'
  - e.g., inductive proofs
- Annotate to distinguish embedding from rest
  - language of wave-fronts and skeletons
- Manipulate goal so that embedding is complete sub-term
  - use annotated rewrite rules
- Use given to rewrite this sub-term,
  - e.g., to true.

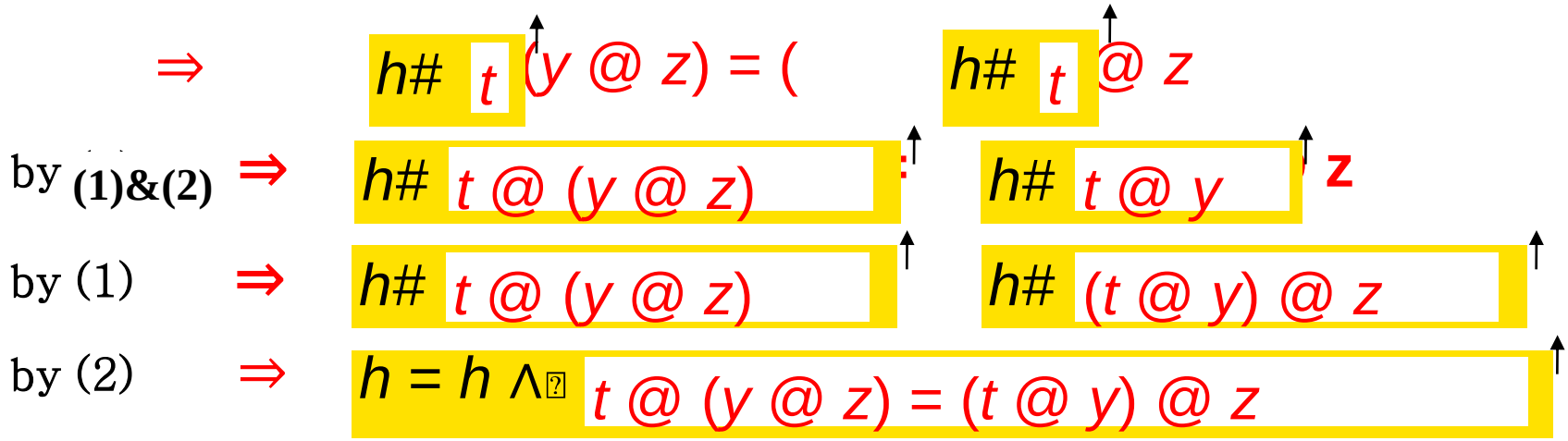
Example of Rippling

Wave-rules

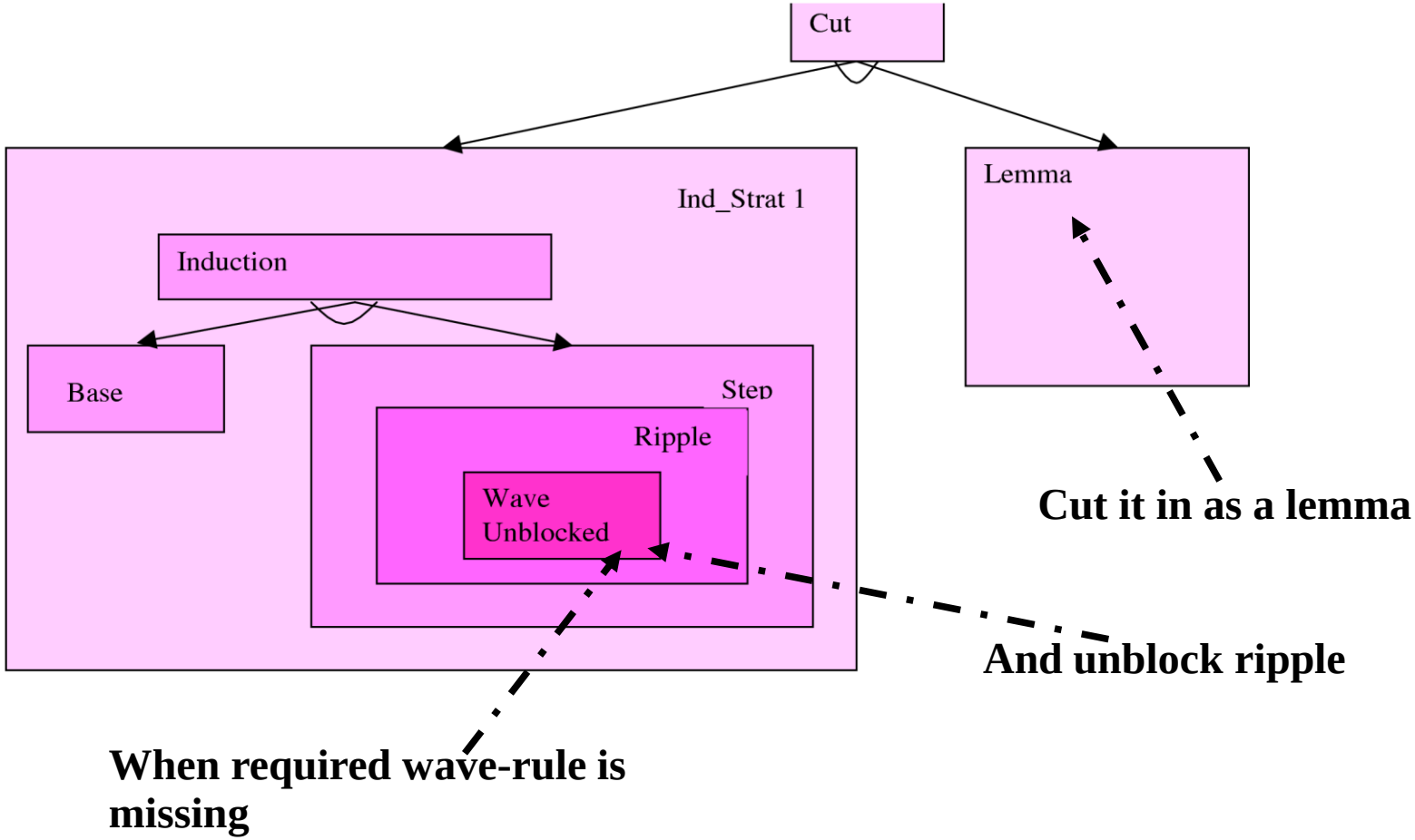


Rippling

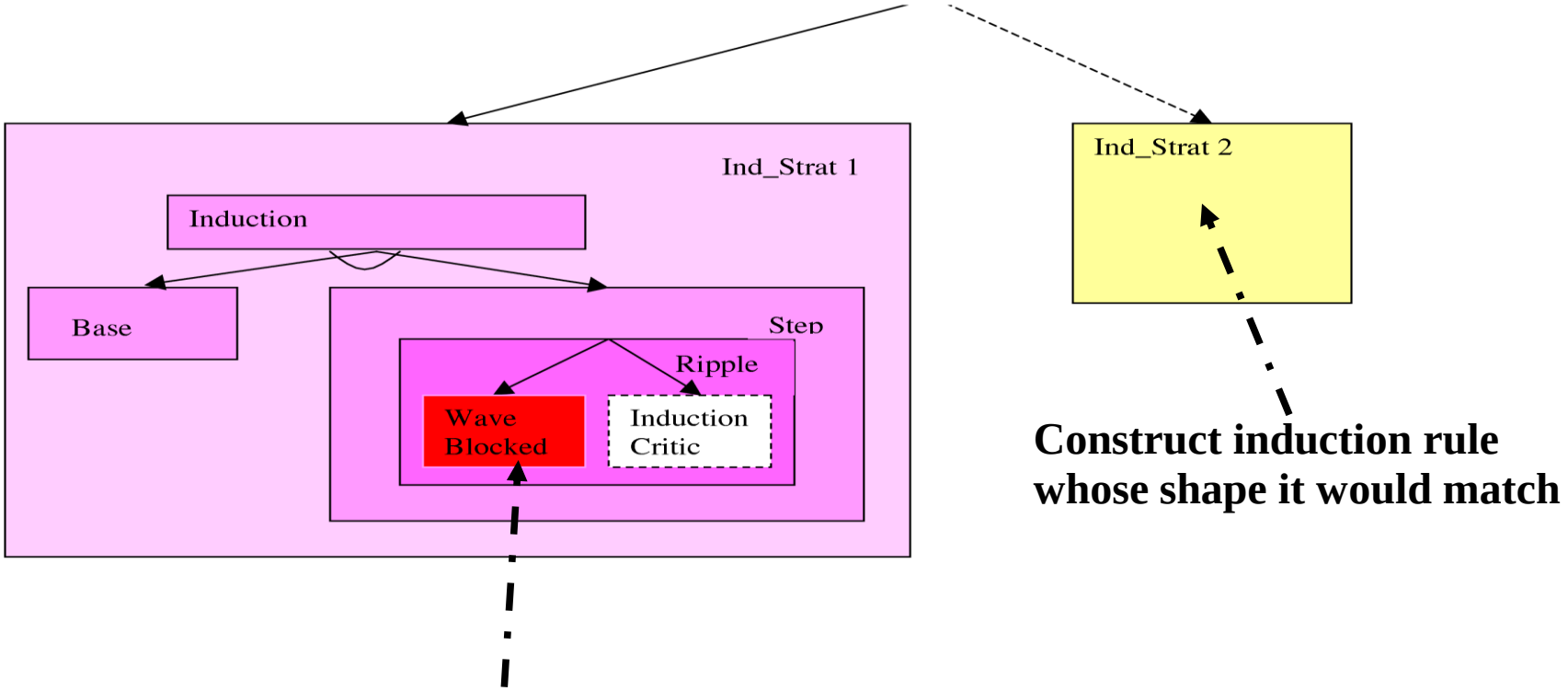
$t \mathbin{@} (Y \mathbin{@} Z) = (t \mathbin{@} Y) \mathbin{@} Z$



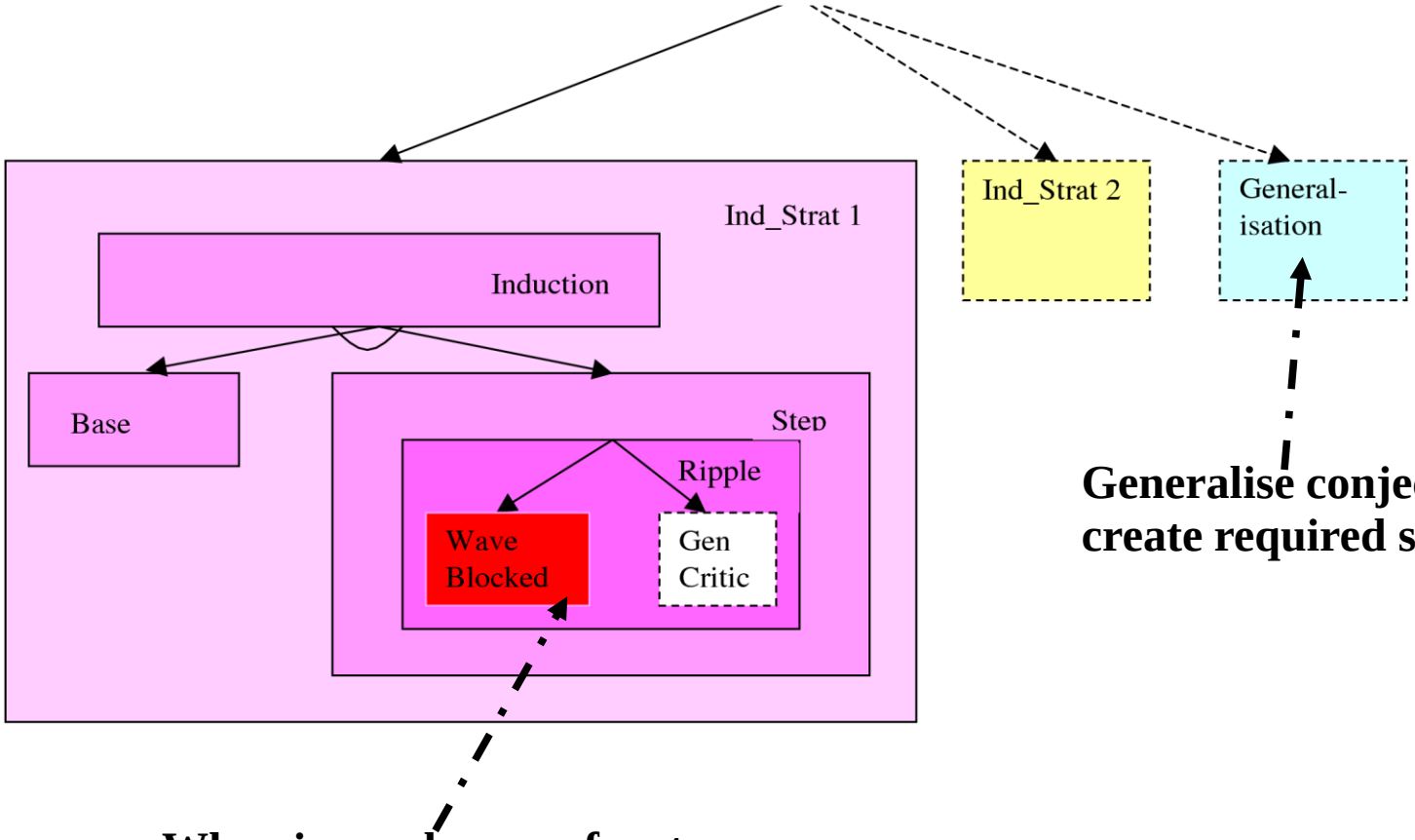
Introduce Intermediate Lemma



Change Induction Rule



Generalise Conjecture



Alternative *Unstructured* Option

- Follow 'ACL2' methodology
- 'Strategy' as provision of key lemmas
- Identify lemmas provided in source proof
  - Maybe of 'toy' conjecture whose proof fails in the 'same' way.
- Construct corresponding lemma
  - Maybe the same lemma or similar one
- Prove and add lemma – recurse
  - Maybe to particular rule set

Conclusion

- Hiproof as 'strategy language' for describing proof structure
- Parse source proof then use to guide target proof
- Limited by available tactics
  - but 'black box' unparsable sub-proofs
- Alternative is 'ACL2' methodology
  - no structure – identify and add lemmas