Adapting Piecewise Fertilisation to Reason about Hypotheses

Louise A. Dennis[⋆]
[⋆]Department of Computer Science
University of Liverpool
L.A.Dennis@liverpool.ac.uk

Lucas Dixon[†]
[†]School of Informatics
University of Edinburgh
ldixon@inf.ed.ac.uk

1 Introduction

Fertilisation is the point in an inductive proof when the induction hypothesis is used to discharge (or rewrite) the induction conclusion.

Piecewise Fertilisation was developed by Armando et al. (1999) for handling situations where logical connectives appear in the theorem statement which make the fertilisation process less straightforward.

Of particular interest is the situation where an implication appears in the theorem statement. Typically a theorem of the form

$$\forall x. P(x) \Rightarrow Q(x), \tag{1}$$

produces (in the case where x is a natural number) a step case of the form

$$(P(n) \Rightarrow Q(n)) \wedge P(s(n)) \vdash Q(s(n)), \tag{2}$$

to discharge this piecewise fertilisation breaks this into two sub-problems:

$$P(s(n)) \Rightarrow P(n), \tag{3}$$

and

$$Q(n) \Rightarrow Q(s(n)). \tag{4}$$

This abstract describes preliminary work to take the idea of piecewise fertilisation and combine it with rippling techniques to provide support for reasoning within the hypotheses of inductive proofs.

2 An Example: Counting an element in a List

Let us consider the theorem

$$\forall x, l. x \notin l \Rightarrow cl(x, l) = 0. \tag{5}$$

Where $\in$ is list membership and cl is defined as

$$cl(a, []) = 0, \tag{6}$$

$$cl(a, h :: t) = \text{if } (a = h) \text{ then } s(cl(a, t)) \text{ else } cl(a, t). \tag{7}$$

A proof-planning style induction proof on this theorem proceeds by induction on l. The base case is easily discharged, leaving the step case:

$$(\forall z. z \notin t \Rightarrow cl(z, t) = 0) \wedge y \notin (h :: t) \vdash cl(y, h :: t) = 0, \tag{8}$$

which ripples/rewrites to

$$(\forall z. z \notin t \Rightarrow cl(z, t) = 0) \wedge y \notin (h :: t) \vdash \text{ if } (y = h) \text{ then } s(cl(y, t)) \text{ else } cl(y, t) = 0, \tag{9}$$

and then case splits to

$$(\forall z. z \notin t \Rightarrow cl(z, t) = 0) \wedge y \notin (h :: t) \wedge y \neq h \vdash cl(y, t) = 0, \tag{10}$$

$$(\forall z. z \notin t \Rightarrow cl(z, t) = 0) \wedge y \notin (h :: t) \wedge y = h \vdash s(cl(y, t)) = 0. \tag{11}$$

The second of these has a contradiction in the hypothesis (which can be detected by counter-example finding and discharged easily)

In piecewise fertilisation the case-split of (2) into (3) and (4) is done by identifying an embedding between the antecedant of the induction hypothesis and another hypothesis of the step case. In this case $y \notin h :: t$ embeds into $z \notin t$ (taking into account that z is universally quantified). Instead of forming two sub-problems, as in piecewise fertilisation, we suggest annotating the hypothesis, as for rippling, and rewriting directly.

$$(\forall z. z \notin t \Rightarrow cl(z,t) = 0) \wedge y \notin \boxed{(h :: \underline{t})} \wedge y \neq h \vdash cl(y,t) = 0 \tag{12}$$

$$(\forall z. z \notin t \Rightarrow cl(z,t) = 0) \wedge \boxed{y \neq h \wedge \underline{y \notin t}} \wedge y \neq h \vdash cl(y,t) = 0 \tag{13}$$

Which then lets us infer that $cl(y,t) = 0$ and prove the step case.

3 Current Status

We have most of this process implemented in the IsaPlanner proof planning system (Dixon and Fleuriot, 2003). We are successfully able to perform rippling in the hypothesis of the induction but are not, currently, able to complete the final steps because of some short comings in the implementation of simplification in IsaPlanner – we are working on this.

We believe that this technique will prove useful for simplifying hypotheses in both fully automated and interactive proofs.

Acknowledgements

This work was funded by EPSRC Platform Grant EP/E005713/1.

References

A. Armando, A. Smaill, and I. Green. Automatic synthesis of recursive programs: The proof-planning paradigm. *Automated Software Engineering*, 6(4):329–256, 1999.

L. Dixon and J. D. Fleuriot. IsaPlanner: A prototype proof planner in Isabelle. In *19th International Conference on Automated Deduction (CADE'2003)*, volume 2741 of *LNCS*, 2003.