Proof Critics for IsaPlanner

Moa Johansson Lucas Dixon
Alan Bundy
*School of Informatics, University of Edinburgh
Appleton Tower, Crichton St, Edinburgh EH8 9LE, UK
{moa.johansson, lucas.dixon, a.bundy}@ed.ac.uk

Abstract

The discovery of missing lemmas and case-splits are challenging problems for automated theorem proving. Most interactive provers rely on the user for guidance through these proof-steps. Proof-planning critics were introduced by Ireland as a way of automating this. Here, we present ongoing work developing critics for lemma speculation and case-analysis in higher-order logic in the IsaPlanner system.

1 Introduction

Proof-planning is a method for automating proof search, exploiting the fact that there exists families of proofs that share a similar structure (Bundy (1988)). One example is proofs by mathematical induction, which consist of a base-case and a step-case that is proved by appealing to the inductive hypothesis. Rippling is a heuristic commonly used in proof-planning to guide rewriting of the step-case goal towards a form where the hypothesis can be applied (Bundy et al. (2005)). It works by annotating syntactic similarities between the hypothesis and the goal. Only rewrites that preserve the similarities while decreasing the differences are allowed. Proof-planning critics were introduced by Ireland and Bundy (1996) as a method of rescuing failed proof-attempts. Each critic is triggered by a particular failure pattern. This information is then used to suggest a patch, such as the introduction of a missing lemma, a generalisation, a case-split or a new induction rule.

IsaPlanner is a proof-planner for the theorem prover Isabelle (Dixon (2005)). It provides an inductive prover, with an efficient implementation of higher-order rippling. Until recently, IsaPlanner only supported one critic, for lemma calculation. This critic would simply attempt to prove a generalised version of a blocked goal as a lemma. Recent developments of IsaPlanner's proof-representation, including improved support for managing shared meta-variables in proofs and the capability to refer to subgoals by name, have made it possible to start experimenting with more sophisticated critics. Here we describe the implementation of two such critics, for lemma speculation and for case-analysis, inspired by the work of Ireland and Bundy (1996).

2 Lemma Speculation

Rippling is said to be *blocked* if the inductive hypothesis cannot be applied and no more rewrites exists that will decrease the differences towards the step-case goal. This might indicate that an extra lemma is needed. The lemma speculation critic attempts to create a schematic lemma that keeps the similarities and inserts higher-order meta-variables standing for yet unknown term-structure. As an example consider a blocked rippling attempt in the step-case of the proof (by induction on t) of $rev(rev(t) \mathbin{@} l) = rev(l) \mathbin{@} t$[1].

$$\begin{aligned} Given: &\quad \forall l'.\, rev(rev(t) \mathbin{@} l') = rev(l') \mathbin{@} t \\ Goal: &\quad rev(rev(t) \mathbin{@} [h] \mathbin{@} l) = rev(l) \mathbin{@} (h :: t) \end{aligned}$$

The hypothesis (given) is not applicable to the goal and no more rewrite rules can be applied, why the lemma speculation critic would be fired. The first choice point for the critic is to pick a subterm to unblock. For the purpose of this example, we choose the right hand side of the goal. This term will constitute the left hand side of our lemma. As rippling require preservation of similarities between the goal and the given, the right hand side is constructed by inserting meta-variables, standing for term structures yet to be determined, into an instance of the corresponding part of the hypothesis. We prefix meta-variables with '?'.In our example we get the schematic lemma:

$$rev(l) \mathbin{@} (h :: t) = ?F_1((?F_2(rev(l),\ h,\ t,\ l) \mathbin{@} t),\ h,\ t,\ l)$$

[1] rev is the list reversal function, '@' denote append and '::' stands for cons.

Viewing the term as a tree, meta-variables are inserted above each function symbol. For example, $?F_2$ is inserted 'above' $rev(l)$, taking this as its first argument. To ensure the correct instantiations for meta-variables are allowed, they each also take the variables of the goal (h, t and l) as arguments. Application of this lemma gives the new goal:

$$Goal: \;\; rev(rev(t) \mathbin{@} [h] \mathbin{@} l) = ?F_1((?F_2(rev(l),\ h,\ t,\ l) \mathbin{@} t),\ h,\ t,\ l)$$

After applying the schematic lemma, rippling based rewriting continues, aiming to instantiate the meta-variables. For this rewriting process to be more efficient, it is necessary to restrict higher-order unification. Otherwise, it would be possible to unify a top-level variable, such as $?F_1$, with the left-hand side of almost any rewrite rule, resulting in a huge search space. Instead, we check the goal for function symbols such as rev, and match this to rewrite rules also containing some occurrence of rev. Consider a rewrite rule together with the goal:

$$\underbrace{rev(X)} \mathbin{@} [Y] \Rightarrow rev(Y :: X) \qquad\qquad \cdots = ?F_1((?F_2(\underbrace{rev(l)},\ h,\ t,\ l) \mathbin{@} t),\ h,\ t,\ l)$$

The critic notices a common occurrence of rev in the LHS of the rule and in the RHS of the goal. X can be unified with l, so to make the rule applicable we need to unify the meta-variable $?F_2$ with the remaining term-structure of the LHS of the rule. This suggest the unifier $\lambda z.z \mathbin{@} [?Y_2(h,\ t,\ l)]$ for $?F_2$. After this rewrite the goal takes the form:

$$Goal: \;\; rev(rev(t) \mathbin{@} [h] \mathbin{@} l) = ?F_1((rev(?Y_2(h,\ t,\ l) :: l) \mathbin{@} t), h,\ t,\ l)$$

Although we restrict the rewriting to rules involving common function symbols the search space can still be large, particularly when working in large theories. We hope to develop further heuristics to improve the efficiency of this.

After rewriting, the proof is concluded by exploring the projections of remaining meta-variables, thereby attempting to make the inductive hypothesis applicable. Here, $?F_1$ is projected onto its first argument and $?Y_2$ is projected onto h. The resulting subterm $h :: l$ can now be unified with the universally quantified l' in the hypothesis to find a match. As IsaPlanner now supports shared meta-variables across the proof, the steps described above has the side effect of instantiating the schematic lemma:

$$rev(l) \mathbin{@} (h :: t) = rev(l) \mathbin{@} [h] \mathbin{@} t$$

The lemma can now be generalised and proved using the machinery already existing in IsaPlanner.

3 Case Analysis

Program specifications commonly contain if- and case-statements. As a first step towards automating some software verification proofs, a critic for case-analysis of if-statements has been developed. It automatically introduces a case-split during rippling if the condition of the if-statement cannot be shown to be true or false. This results in two new subgoals that are solved either by further rippling or by simplification if rippling is not applicable. We are also in the process of implementing a similar critic for case-statements.

4 Conclusion and Further Work

We have described some ongoing work implementing proof critics in IsaPlanner to improve its automation, following the ideas of Ireland and Bundy (1996). A first version of a lemma speculation critic has been implemented. We hope to further experiment with this critic to get a better idea of its search space and improve the heuristics for rewriting in the presence of meta-variables.

A critic for case-analysis of if-statements has also been implemented, to aid automation of software verification proofs. Further critics we hope to implement include generalisation of accumulator variables in proofs about tail recursive functions, and critics for revision and synthesis of induction schemes.

References

Alan Bundy. The use of explicit plans to guide inductive proofs. In *Conference on Automated Deduction*, pages 111–120, 1988. URL `citeseer.nj.nec.com/bundy88use.html`.

Alan Bundy, David Basin, Dieter Hutter, and Andrew Ireland. *Rippling: Meta-level Guidance for Mathematical Reasoning*. Springer-Verlag, 2005.

Lucas Dixon. *A Proof Planning Framework for Isabelle*. PhD thesis, University of Edinburgh, 2005.

Andrew Ireland and Alan Bundy. Productive use of failure in inductive proof. *JAR*, 16(1–2):79–111, 1996.