Abstract. This paper describes Hipster, a system integrating theory exploration with the proof assistant Isabelle/HOL. Theory exploration is a technique for automatically discovering new interesting lemmas in a given theory development. Hipster can be used in two main modes. The first is exploratory mode, used for automatically generating basic lemmas about a given set of datatypes and functions in a new theory development. The second is proof mode, used in a particular proof attempt, trying to discover the missing lemmas which would allow the current goal to be proved. Hipster’s proof mode complements and boosts existing proof automation techniques that rely on automatically selecting existing lemmas, by inventing new lemmas that need induction to be proved. We show example uses of both modes.

1 Introduction

The concept of theory exploration was first introduced by Buchberger [2]. He argues that in contrast to automated theorem provers that focus on proving one theorem at a time in isolation, mathematicians instead typically proceed by exploring entire theories, by conjecturing and proving layers of increasingly complex propositions. For each layer, appropriate proof methods are identified, and previously proved lemmas may be used to prove later conjectures. When a new concept (e.g. a new function) is introduced, we should prove a set of new conjectures which, ideally, “completely” relates the new with the old, after which other propositions in this layer can be proved easily by “routine” reasoning. Mathematical software should be designed to support this workflow. This is arguably the mode of use supported by many interactive proof assistants, such as Theorema [3] and Isabelle [17]. However, they leave the generation of new conjectures relating different concepts largely to the user. Recently, a number of different systems have been implemented to address the conjecture synthesis aspect of theory exploration [13,15,16,5]. Our work goes one step further by integrating the discovery and proof of new conjectures in the workflow of the interactive theorem prover Isabelle/HOL. Our system, called Hipster, is based on our previous work on HipSpec [5], a theory exploration system for Haskell programs. In that work, we showed that HipSpec is able to automatically discover many of the kind of equational theorems present in, for example, Isabelle/HOL’s libraries for
natural numbers and lists. In this article we show how similar techniques can be used to speed up and facilitate the development of new theories in Isabelle/HOL by discovering basic lemmas automatically.

Hipster translates Isabelle/HOL theories into Haskell and generates equational conjectures by testing and evaluating the Haskell program. These conjectures are then imported back into Isabelle and proved automatically. Hipster can be used in two ways: in *exploratory mode* it quickly discovers basic properties about a newly defined function and its relationship to already existing ones. Hipster can also be used in *proof mode*, to provide lemma hints for an ongoing proof attempt when the user is stuck.

Our work complements Sledgehammer [18], a popular Isabelle tool allowing the user to call various external automated provers. Sledgehammer uses *relevance filtering* to select among the available lemmas those likely to be useful for proving a given conjecture [14]. However, if a crucial lemma is missing, the proof attempt will fail. If theory exploration is employed, we can increase the success rate of Isabelle/HOL’s automatic tactics with little user effort.

As an introductory example, we consider the example from section 2.3 of the Isabelle tutorial [17]: proving that reversing a list twice produces the same list. We first apply structural induction on the list $xs$.

```isabelle
theorem rev_rev : "rev(rev xs) = xs"
apply (induct xs)
```

The base case follows trivially from the definition of $rev$, but Isabelle/HOL’s automated tactics simp, auto and sledgehammer all fail to prove the step case. We can simplify the step case to:

$$rev(rev xs) = xs \implies rev((rev xs) @ [x]) = x#xs$$

At this point, we are stuck. This is where Hipster comes into the picture. If we call Hipster at this point in the proof, asking for lemmas about $rev$ and append (@), it suggests and proves three lemmas:

```isabelle
lemma lemma_a: "xs @ [] = xs"
lemma lemma_aa : "(xs @ ys) @ zs = xs @ (ys @ zs)"
lemma lemma_ab : "(rev xs) @ (rev ys) = rev (ys @ xs)"
```

To complete the proof of the stuck subgoal, we need lemma ab. Lemma ab in turn, needs lemma a for its base case, and lemma aa for its step case. With these three lemmas present, Isabelle/HOL’s tactics can take care of the rest. For example, when we call Sledgehammer in the step case, it suggests a proof by Isabelle/HOL’s first-order reasoning tactic metis [11], using the relevant function definitions as well as lemma_ab:

```isabelle
theorem rev_rev : "rev(rev xs) = xs"
apply (induct xs)
apply simp
sledgehammer
by (metis rev.simps(1) rev.simps(2) app.simps(1) app.simps(2) lemma_ab)
```