Theory Exploration for Interactive Theorem Proving

Moa Johansson
Chalmers University of Technology

Abstract

Theory exploration is an automated reasoning technique for discovering and proving interesting properties about some set of given functions, constants and datatypes. In this note we describe ongoing work on integrating the HipSpec theory exploration system with the interactive prover Isabelle. We believe that such an integration would be beneficial for several reasons. In an interactive proof attempt a natural application would be to allow the user to ask for some suggestions of new lemmas that might help the current proof development. Theory exploration may also be used to automatically generate and prove some basic lemmas as a first step in a new theory development. Furthermore, when the theory exploration system is used as a stand-alone system, it should output a checkable proofs, for instance for Isabelle, so that sessions can be saved for future use.

1 Introduction

Theory exploration is an automated reasoning technique for discovering and proving interesting properties about some set of given functions, constants and datatypes. In the context of theorem proving, theory exploration systems are typically used to generate candidate conjectures satisfying some intrestingness criteria, using automated testing or counter-example finding to produce candidates likely to be theorems.

Many theory formation systems, such as MATHsAiD [6], IsaCoSy [5] and IsaScheme [7] are automatic and attempt to produce a background theory with 'interesting' lemmas which are likely to be useful to in later proof attempts. However, while both IsaCoSy and IsaScheme are built on top of the interactive prover Isabelle [8], neither can be called from an interactive proof attempt. This problem is due to long runtimes, IsaCoSy and IsaScheme are generally to slow to wait for in an interactive proof attempt. We have recently developed a new theory exploration system called HipSpec, which discovers and proves properties about Haskell programs by induction [2]. HipSpec is considerably faster than previous systems, and thus a candidate for integration in an interactive system.

Large theories with many functions and datatypes is a major challenge for automated theory exploration systems. To deal with theories with many hundreds, or even thousands, of functions requires some form of modular exploration and division into smaller sub-theories. In the interactive scenario we can to some extent circumvent this problem. Where the user asks the system for suggestions of lemmas applicable to the current subgoal, the input to the theory exploration system can be directly specified by the user or limited to functions relevant to that subgoal.

In this note, we describe a new project aiming at integrating theory exploration and interactive theorem provers:

- Theory exploration has so far mainly been used in conjunction with automated theorem provers. Many proof developments are however too complex to be fully automated. By integrating theory exploration with an interactive theorem prover we make theory exploration available to a wider audience.
- We aim to develop a tool integrated with the interactive proof assistant Isabelle which allow a user who is stuck in a proof attempt to ask for some candidate lemmas which could help the current proof attempt. It is crucial that the system responds quickly and does not produce too many (or too few) candidates.
- We would like theory exploration systems such as HipSpec to produce a sensible output format, so that proofs can be independently checked and lemmas recorded in a library. Producing Isabelle theory files is one option.

2 Background: The HipSpec System

HipSpec is an inductive theorem prover and theory exploration system for discovering and proving properties about Haskell programs. HipSpec takes a Haskell program annotated with properties which the user wish to prove and first generate and prove a set of equational theorems about the functions and datatypes present. These are added to its background theory and user specified properties are then proved in this richer theory. Experimental results have been encouraging, HipSpec performs very well and proved more theorems automatically than other state-of-the-art inductive theorem provers [2].

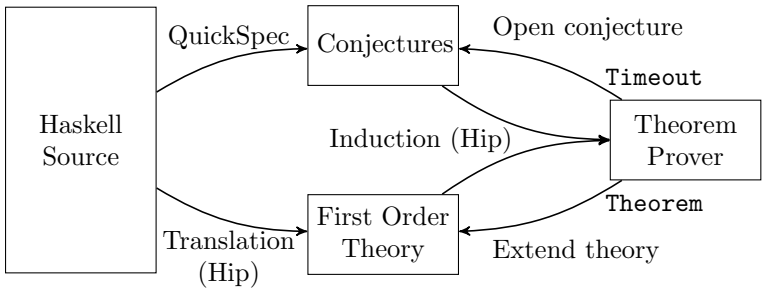

Figure 1: Architecture of the HipSpec system.

HipSpec combines the inductive prover Hip [10] with the QuickSpec system [3]. An overview of HipSpec is shown in Figure 1. HipSpec first translates the function definitions and datatypes from the Haskell program into first-order logic to form an initial theory about the program. QuickSpec also reads in the available function symbols and datatypes and proceeds to generate new terms from these. Using the automated testing framework from the QuickCheck system [1], QuickSpec divides these terms into equivalence classes, from which equational conjectures are extracted. These conjectures are passed back to Hip, which applies a a suitable induction scheme and feeds the resulting proof-obligations, along with the first-order background theory, to an automated first-order prover, for instance Z3 [4]. If the proof succeeds, the new theorem is added to the background theory, and may be used as a lemma in subsequent proof attempts. If a proof fails, HipSpec tries other conjectures and may return to open ones later, when more lemmas have been added to the background theory.

3 Integrating HipSpec and Isabelle

Isabelle's higher-order logic is essentially a (terminating and finite) subset of the functional programming language Haskell, and Isabelle's code generator can translate Isabelle theories into Haskell programs. HipSpec then needs to monomorphise any polymorphic types in order to translate it to first-order logic, after which it can generate lemmas about the functions corresponding to the Isabelle theory. A very early prototype allowing HipSpec to be called from Isabelle has been implemented, but a lot of implementational work remain, in particular to import lemmas discovered back into Isabelle.

HipSpec does not produce proofs, as it relies on external provers as black boxes. We have however experimented with using Z3's capability to produce unsatisfiable cores in order to report back which lemmas were used in a proof. Using this information, we plan to experiment with a lightweight inductive tactic for Isabelle, which should apply induction and simplification using the relevant lemmas, thus verifying the proof in Isabelle as an added soundness check. Given the right lemmas, we expect such a simple induction tactic would succeed in proving many of the conjectures which HipSpec has discovered. This is similar to how the Sledgehammer system works [9], it sends a conejcture from Isabelle to external provers and then replays the proof using Isabelle's internal prover Metis. Once we have an Isabelle tactic for HipSpec we can use it in several ways:

Automated induction: HipSpec takes the current goal and applies theory exploration and induction. If it succeeds, it reports which induction scheme and lemmas were used to its corresponding tactic, which replays the proof in Isabelle. If any new lemmas were found, information to replay their proofs should also be produced.

Generate background lemmas: The user specifies a set of Isabelle functions and datatypes which are of interest. HipSpec applies theory exploration to these and generate a set of interesting lemmas along with instructions to verify the proofs inside Isabelle. This could be done as a first step in a new theory development, to automatically generate many basic lemmas before the user tackle more complicated theorems. Alternatively, the user may call theory exploration to help in an ongoing proof-attempt where the user is stuck. In this case, theory exploration is even more restricted, it should only generate lemmas which really do apply to the goal, anything else is uninteresting. The user should be presented with a short list of options to choose from.

HipSpec use random testing to generate conjectures, hence all conjectures have passed a large number of tests before being submitted to the prover. In an interactive setting, even conjectures HipSpec has failed to prove automatically can be of interest, with the user interactively supplying the proof, should she/he find the conjecture is useful.

Produce Isabelle theories about Haskell programs: Ultimately we would like HipSpec to be useful for verifying real Haskell programs. As mentioned, HipSpec currently use a black-box external prover and does not output any proofs. For verification purposes we may want some additional reassurance. Therefore, we suggest that HipSpec should have the option to output its results Isabelle theories. The user can then inspect the Isabelle theory, and recheck the proofs independently if required. Lemmas about libraries can also more easily be imported if required. Translating from Haskell to Isabelle is however not as straight forward as the other way around, Haskell is a lazy language and also support infinite datatypes and non-terminating functions, which Isabelle/HOL does not. Hence, in the first instance we will be limited to the finite and terminating subset of Haskell.

4 Summary

We believe that theory exploration would be very useful also in an interactive setting. It could assist the user with suggestions of lemmas relevant to the current proof attempt, or simply to generate basic lemmas in a new theory. By letting the user specify the functions and datatypes passed to theory exploration, we can control the search space and explore larger and more complex theories that are currently beyond the scope of the fully automatic version. We are currently working on integrating the theory exploration system HipSpec with the interactive prover Isabelle.

References

[1] K. Claessen and J. Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. In *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, ICFP '00, pages 268–279, New York, NY, USA, 2000. ACM.

[2] K. Claessen, M. Johansson, D. Rosén, and N. Smallbone. Automating inductive proofs using theory exploration. In *Proceedings of the Conference on Auomtated Deduction (CADE)*, 2013.

[3] K. Claessen, N. Smallbone, and J. Hughes. QuickSpec: guessing formal specifications using testing. In *Proceedings of the 4th international conference on Tests and proofs*, TAP'10, pages 6–21, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] L. De Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proceedings of TACAS*, TACAS'08/ETAPS'08, pages 337–340. Springer-Verlag, 2008.

[5] M. Johansson, L. Dixon, and A. Bundy. Conjecture synthesis for inductive theories. *Journal of Automated Reasoning*, 47(3):251–289, 2011.

[6] R. L. McCasland, A. Bundy, and P. F. Smith. Ascertaining mathematical theorems. *Electron. Notes Theor. Comput. Sci.*, 151(1):21–38, Mar. 2006.

[7] O. Montano-Rivas, R. McCasland, L. Dixon, and A. Bundy. Scheme-based theorem discovery and concept invention. *Expert Systems with Applications*, 39(2):1637–1646, Feb. 2012.

[8] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[9] L. Paulson and J. Blanchette. Three years of experience with Sledgehammer, a practical link between automation and interactive theorem provers. In *Proceedings of IWIL-2010*, 2010.

[10] D. Rosén. Proving equational Haskell properties using automated theorem provers. Master's thesis, University of Gothenburgh, 2012.