Inductive Proofs and Theory Exploration

Docent Lecture

20 March 2015

Moa Johansson

A proof by induction

How do I prove that all slices of cake are tasty using structural induction?

SUPERCHLORINE.com

Step 1. Define a set of cake slices recursively.

is cake.

A proof by induction

A proof by induction

A proof by induction

*For those not participating in the Induction Workshop, we'll verify this proof on Monday 23/3 at 15.15, in the lunchroom.

When do we need inductive proofs?

- Properties involving repetition need induction:
  - Recursive data-types and functions.
  - Loops
  - E.g. functional correctness of Haskell programs.
- Not decidable.

Why is induction hard to automate?

- May need lemmas or generalisations.
- Needing another inductive proof…
- Eureka steps:

  Often assumed to need (human) creativity.
  - What lemma do we need?
  - Should I generalise my conjecture?
  - What induction scheme?

Can an automated theorem prover help with this?

Lemma discovery

- Proof Critics [Bottom-up]:
  - Analyse proof failure.
  - Finds many easy lemmas.
  - Often fails to find more complex or unexpected ones.
- Theory Exploration [Top-down]:
  - Construct richer background theory.
  - Increases power of automated prover.
  - Also useful for human user in interactive setting.

What is theory exploration?

Theory exploration paradigm [Buchberger-00]

- Theorems not proved in isolation.
- Mathematicians explore whole theories:
  - Prove routine lemmas
  - Prove more complex theorems
  - Maybe backtrack: need more lemmas
  - New theories on top of old ones.

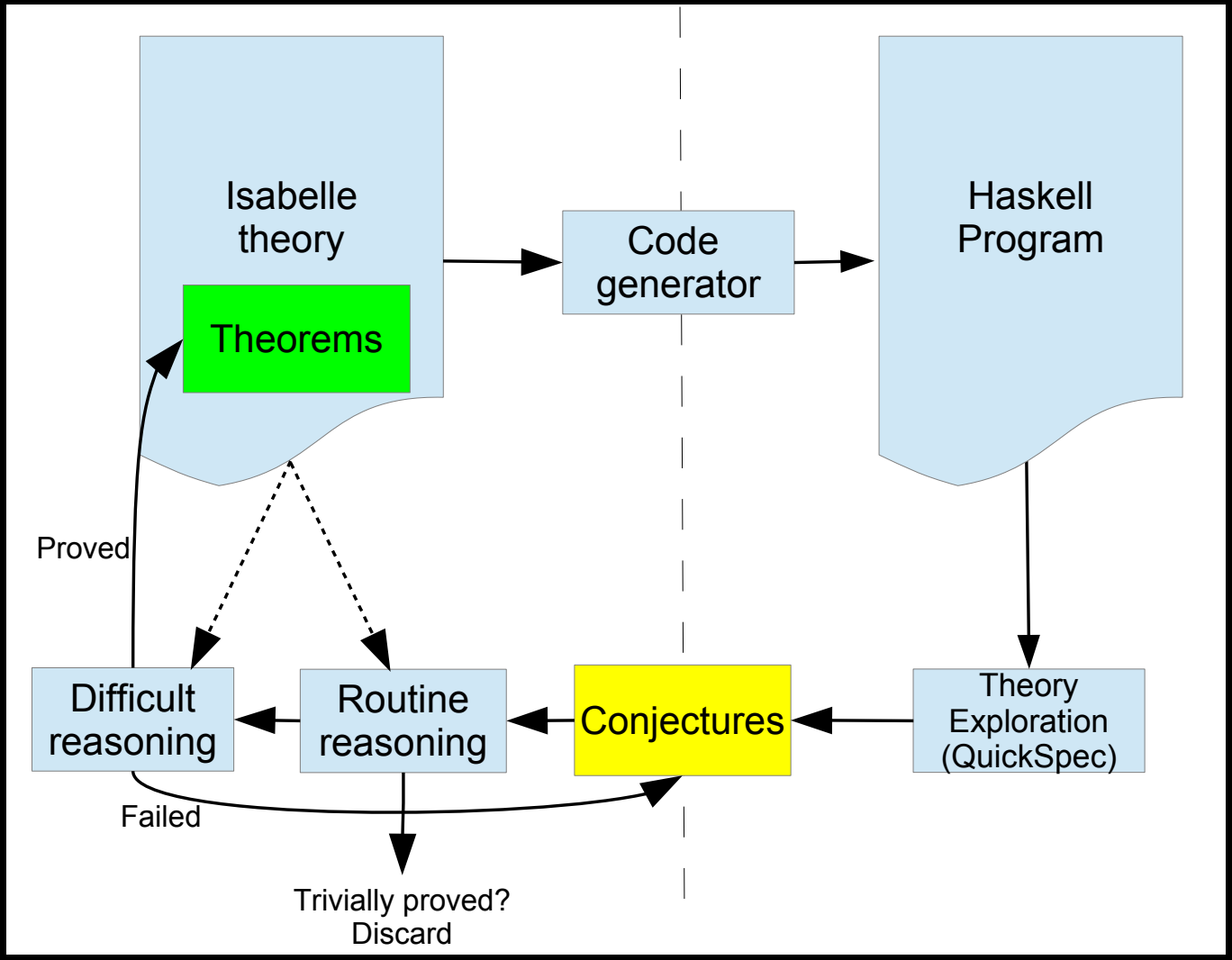
Inductive Theorem Proving and Theory Exploration

- Find interesting lemmas automatically.
- Richer theory for prover to work with.
- Input:
  - Functions, constants, datatypes.
- Testing to avoid generating false ones.
  - Not just simple generate-and-test…
- Keep only interesting, non-trivial.
  - Interesting = “hard” to prove.

Demo:

Let’s explore some trees.

Hipster: Overview



Theory exploration and induction @ Chalmers

- QuickSpec: Speculate laws about Haskell programs
  - Theory exploration to understand programs, find mistakes and aid development.
- HipSpec: Inductive theorem prover (Haskell)
  - Theory exploration in an automated prover.
- Hipster: Theory Exploration for Isabelle/HOL.
  - Theory exploration in an interactive setting.

What's next?

- Learning from proof libraries.
  - Similarities between theories.
  - Proof by analogy.
- Proofs and specs for Haskell programs.
  - Not just QuickCheck properties, also proofs!
  - Programing in a theorem prover?
  - Exploration-driven development?

Summary

- Automating inductive proof is challenging
  - E.g. lemmas (also needing induction)
- One approach: Theory exploration
  - Automatically find and prove routine lemmas
  - Richer background theory enhance power
  - Also applications to
    - Program understanding
    - Interactive theorem proving
    - Maths?