Theorem Provers Part 2: Inefficient Computation

Thomas Sewell

Dec 2024

Theorem Provers & Inefficient Computation

In part 2 of this lecture series, we'll look at some puzzles using a theorem prover.

Recap from talk 1. What is a mechanized logic?

- A language of terms & propositions.
- A language of proofs of propositions.
- A checker for these proofs.
- There are lots of possible design decisions.

Skipped for now: What are our motivating goals?

Proving in Isabelle/HOL

Isabelle, like other interactive proof tools, is internally quite complex, but tries to be self-documenting.

- There are many facts in its database.
  - From background libraries.
  - Created by definitional tools.
- There are various query mechanisms.
  - `find_theorems`
  - `try`

Two Easy Demonstrations

Let's have a look at two fairly easy problems.

- A proof of a sufficient condition to show that the recursive transitive closure of a relation R is the total set.
- A proof that the tail-recursive implementation of reverse preserves the set of elements.

Some of these exercises have been uploaded to the web:

- `https://www.cse.unsw.edu.au/~tsewell/lss/`

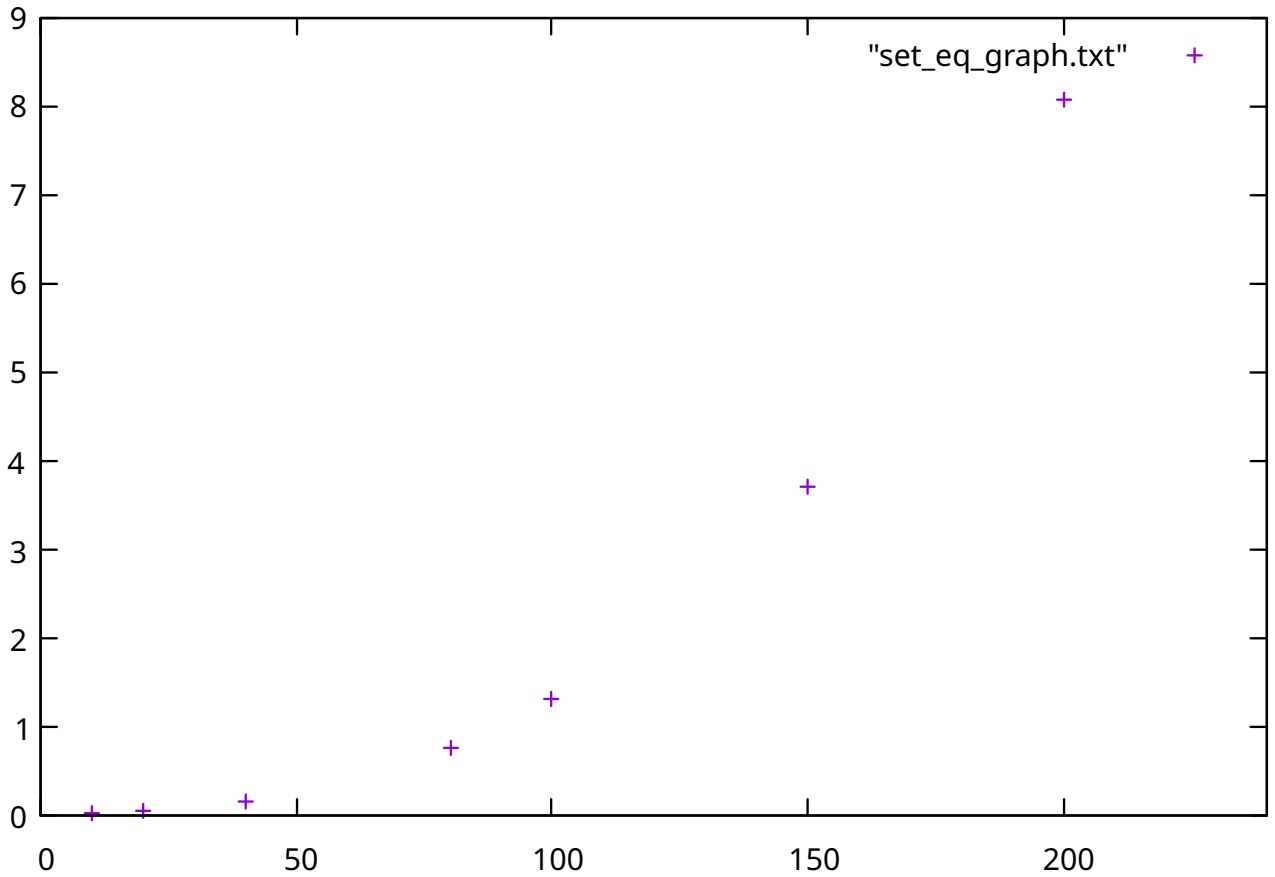
A Tricky Demonstration

Here is a lemma we can prove fairly easily:

$$\{x, 3, 4, 5, 6\} = \{6, 5, 4, 3, x\}$$

What about generalising 6 to larger integers?

A Graph



A Family of Puzzles

This is part of a family of similarly-difficult puzzles:

$$\{x, 3, 4, 5, 6\} = \{6, 5, 4, 3, x\}$$

$$f\ x + f\ 3 + f\ 4 + f\ 5 + f\ 6 = f\ 6 + f\ 5 + f\ 4 + f\ 3 + f\ x$$

$$\text{perm}\ [x, 3, 4, 5, 6]\ [6, 5, 4, 3, x] \longleftrightarrow x = y$$

Puzzles as Learning Tools

Here is one more perspective on proofs-as-programs:

Sometimes our proof approach is essentially a program. But which one?

The goal of the exercise is to think about what is possible. What is the best we can do within our system?

I have used these puzzles to learn and teach Isabelle, HOL4, ACL2 and Coq. Some day soon I will get to Lean.

Puzzle Automation

$$\{x, 3, 4, 5, 6\} = \{6, 5, 4, 3, x\}$$

The best we can do, for now, is to implement an $n \log(n)$ sorting algorithm in proof steps.

Recap

Recap. How do we learn to use a tool like Isabelle/HOL?

- How do we get started? How can the tool help educate us?
- How do we master the tool? How do we learn what it can and cannot do?

Recap

Recap. How do we learn to use a tool like Isabelle/HOL?

- How do we get started? How can the tool help educate us?
- How do we master the tool? How do we learn what it can and cannot do?

That's all for today. Thanks for listening.

Questions?