Formally verified algorithms and data structures in Coq: concepts and techniques

(Formalnie zweryfikowane algorytmy i struktury danych w Coqu: koncepty i techniki)

Wojciech Kołowski

Praca magisterska

Promotor: narazienikt

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

Czerwiec '20 chyba że koronawirus

Abstract

Okasaki but better

Okasaki tylko lepiej

Contents

Chapter 1

A man, a plan, a canal - thesis

1.1 Things to write about

- Introduction: differences between imperative and functional algorithms, differences between performance-oriented design and formal-correctness-oriented design.
- Literature review, Okasaki is old and bad for Coq, SF3 is shallow.
- Binary search trees: a case study to show the basic workflow and that it's not that obvious how to get basic stuff right.
- Design: we shouldn't require proofs in order to run programs. Ways of doing general recursion and, connected with it, functional induction as the way-to-go proof technique. Maybe something about the equations plugin. A word about classes, records and modules.
- Quicksort: in functional languages we have so powerful abstractions that we can actually implement algorithms and not just programs.
- Braun mergesort: in order not to waste resources, we sometimes have to reify abstract patterns, like the splitting in mergesort.
- Cool data structures: ternary search trees, finger trees.

Bibliography