Verified Functional Data Structures: Priority Queues in Liquid Haskell

Master's Thesis

by

MohammadMehran Shahidi

June 8, 2025

University of Kaiserslautern-Landau
Department of Computer Science
67663 Kaiserslautern
Germany

Examiner: Prof. Dr. Ralf Hinze
Michael Youssef, M.Sc.

Declaration of Independent Work

I hereby declare that I have written the work I am submitting, titled "Verified Functional Data Structures: Priority Queues in Liquid Haskell", independently. I have fully disclosed all sources and aids used, and I have clearly marked all parts of the work — including tables and figures — that are taken from other works or the internet, whether quoted directly or paraphrased, as borrowed content, indicating the source.

Kaiserslautern, den 8.6.2025

MohammadMehran Shahidi

Abstract

Formal program verification is a powerful approach to ensuring the correctness of software systems. However, traditional verification methods are often tedious, requiring significant manual effort and specialized tools or languages.

This thesis explores Liquid Haskell, a refinement type system for Haskell that integrates SMT (Satisfiability Modulo Theories) solvers to enable automated verification of program properties. We demonstrate how Liquid Haskell can be used to verify correctness of priority queue implementations in Haskell. By combining type specifications with Haskell's expressive language features, we show that Liquid Haskell allows for concise and automated verification with minimal annotation overhead.

Contents

1. Introduction

1.1. Motivation

Why verifying data structures matters; common bugs; importance in safety-critical systems.

1.2. Problem Statement

Challenges in verification; issues with manual proofs or theorem provers.

1.3. Goals and Contributions

What this thesis aims to achieve—verified priority queues, use of Liquid Haskell, etc.

1.4. Structure of the Thesis

Brief summary of the chapters.

2. Background and Related Work

2.1. Functional Data Structures

Overview of functional programming principles and persistent data structures.

2.2. Priority Queues

Definition, applications, and various implementations (e.g., heaps, binomial queues).

2.3. Program Verification Techniques

Overview of Hoare logic, model checking, interactive theorem proving, etc.

2.4. Refinement Types

What refinement types are, and how they relate to program correctness.

2.5. Liquid Haskell

Architecture, refinement typing in Haskell, and how SMT solvers are used.

2.6. Related Work

Comparison with Coq, Agda, Dafny, or other tools verifying similar structures.

3. Liquid Haskell in Practice

3.1. Overview of Liquid Haskell

Setup, syntax, and capabilities.

3.2. Specification Language

How to write and interpret refinement types.

3.3. Verification Workflow

From writing Haskell code to getting verified guarantees via Liquid Haskell.

3.4. Strengths and Limitations

Where it excels and where it struggles (e.g., termination proofs, higher-order functions).

4. Liquid Haskell in Practice

4.1. Overview of Liquid Haskell

Setup, syntax, and capabilities.

4.2. Specification Language

How to write and interpret refinement types.

4.3. Verification Workflow

From writing Haskell code to getting verified guarantees via Liquid Haskell.

4.4. Strengths and Limitations

Where it excels and where it struggles (e.g., termination proofs, higher-order functions).

5. Priority Queue Implementations

5.1. Specification of Priority Queue Properties

Formal requirements: ordering, insert, deleteMin, etc.

5.2. Simple List-Based Implementation

Haskell code, explanation, and verification.

5.3. Binary Heap Implementation

Code structure, key invariants, and proof techniques.

5.4. Skew/Binomial/Pairing Heap (optional)

More advanced structure and proof techniques (if time allows).

5.5. Comparison of Implementations

Performance, expressiveness, and verification effort.

6. Verification in Liquid Haskell

6.1. Encoding Invariants

How structural and behavioral invariants are expressed in refinement types.

6.2. Use of Measures and Predicates

Defining custom properties over data.

6.3. Example Proofs

Walkthroughs of insert and deleteMin correctness.

6.4. Dealing with Termination and Recursion

How Liquid Haskell checks termination.

6.5. Challenges and Workarounds

What was hard to prove and how you solved it.

List of Figures

A. My Code

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

B. My Ideas

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.