universität innsbruck

Department of
Computer Science

Formalizing an Amortized Cost Analyis of Binomial Heaps and Fibonacci Heaps in Liquid Haskell

Jamie Hochrainer, B.Sc.

M.Sc. Thesis Exposé

Supervisor: Univ.-Prof. Dr. Georg Moser
3rd October 2023

1 Motivation

In computer science, heap data structures play an important role in efficiently managing priority queues. Two popular and powerful types of heaps are Binomial heaps and Fibonacci heaps. Binomial heaps are a collection of binomial trees that satisfies the binomial heap properties. This means each binomial tree in the heap has a unique degree and additionally the key of a node is greater or equal than the key of its parent. One of the advantages of Binomial heaps is their efficient merge operation, which allows for the combination of two heaps with a total number of n nodes in $O(\log n)$ time. Furthermore, insertion and deletion operations are also efficient, taking $O(\log n)$ time.
Fibonacci heaps are another type of heap data structure. They can be derived from Binomial heaps and have a very good amortized time complexity. Contrary to Binomial Heaps, Fibonacci heaps allow multiple Fibonacci trees of the same rank. The main idea of Fibonacci heaps is to postpone expensive Binomial heap operations as long as possible and execute them at a later point. They support deletion from a heap with n nodes in $O(\log n)$ amortized time and other standard heap operations in $O(1)$ amortized time.
Amortized cost analysis is a crucial aspect of understanding the performance of these heap data structures. While individual operations may occasionally be expensive, the amortized cost analysis considers a sequence of operations and calculates their average cost over time. Both Binomial heaps and Fibonacci heaps exhibit exceptional amortized costs for various operations.
To formalize an amortized cost analysis of those two data structures we use Liquid Haskell. Liquid Haskell is a powerful extension of the Haskell programming language that brings a new level of safety and precision to the code. It extends Haskell with formal verification techniques to help writing correct and robust software. We can be confident that the code will behave as expected, reducing the risk of runtime errors, crashes, and security vulnerabilities. Liquid Haskell can prevent errors before they happen. By specifying precise types and invariants, bugs can be caught at compile time, reducing the need for extensive testing and debugging. Developers who are already proficient in Haskell may be able to readily use Liquid Haskell without having to learn an entirely new formal specification language or theorem prover. This reduces the learning curve and makes it accessible to a broader audience of Haskell programmers.

2 Problem Statement

Binomial Heaps and Fibonacci Heaps exhibit complex dynamic behavior due to their nature as self-adjusting data structures. Their performance depends on various factors, including the sequence of operations, the order in which they are executed, and the initial state of the heap. Analyzing their amortized costs manually becomes impractical and error-prone as the complexity of operations and heap size increases. Binomial Heaps and Fibonacci Heaps involve a range of complex operations, including insertion, deletion, and merging of nodes. Analyzing the amortized cost of these operations while accounting for their dynamic behavior can be challenging.

The main objective of the thesis will be to formalize an amortized cost analysis of binomial and Fibonacci heaps in Liquid Haskell. Amortized cost analysis often involves maintaining a potential function to track resources. Developing a suitable resource accounting mechanism that integrates seamlessly with Liquid Haskell is a challenge. Liquid Haskell is a tool for refinement types and formal verification of Haskell code. Developing an automated amortized cost analysis within Liquid Haskell ensures that the analysis is conducted within a safe and formally verified framework. It helps guarantee that the analysis accurately represents the behavior of the heaps. Automated amortized cost analysis allows developers to identify potential bottlenecks and inefficiencies in Binomial and Fibonacci Heaps within the Liquid Haskell environment. This information can guide optimization efforts to improve the practical performance of these data structures in real-world applications. Liquid Haskell relies on refinement types for formal verification. Expressing amortized cost properties as refinement types that accurately capture the complexity of heap operations can be non-trivial.

3 Evaluation Plan and Time Schedule

The plan for this thesis to start research and reading papers in March 2023. Since Liquid Haskell has a lot of functionalities, we will continuously search for useful methods in the literature. The first implementation of basic operations of binomial and Fibonacci heaps will be done in April 2023 and we will extend this implementation with refinements and a complexity analysis in June 2023. This will probably take up to three month. As a next step we can additionally implement the delete method of Fibonacci heaps in October 2023 which will be a challenge in a functional programming language without pointers. Moreover, in October we will start to write down the progress that is done so far. Additionally, as a next step we could think of formalizing another data structure in Liquid Haskell, for example red-black trees and maybe also optimize its delete function as supposed by Mathew Might. The plan is to finish the project and the writing part latest in January 2024.

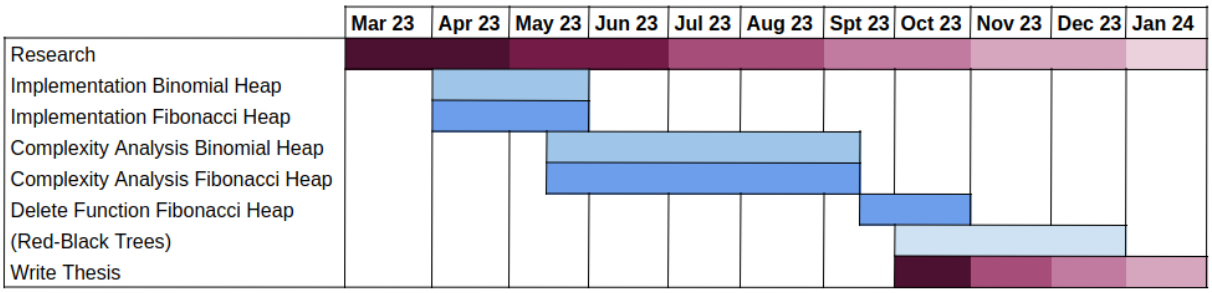

Figure 3.1: Time schedule for master thesis.

Literature

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms, third edition*. MIT Press, 3rd edition, 2009.

Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

Simon Griebel. Verification of the decrease-key operation in fibonacci heaps in imperative hol. Master's thesis in informatics, Technical University of Munic, May 2020.

Martin AT Handley, Niki Vazou, and Graham Hutton. Liquidate your assets: reasoning about resource usage in liquid haskell. *Proceedings of the ACM on Programming Languages*, 4 (POPL):1–27, 2019.

Cohen Josh and Paul Palmer. Verified binomial and skew heaps using liquidhaskell, April 2020.

Bharti Vivek Nitin and P. Renjith. Binomial heap, June 2017.

Chris Okasaki. *Purely functional data structures*. Cambridge University Press, 1999.

Choubey Pranjal and P. Renjith. Amortized analysis, June 2017.

Reddy Roopesh and P. Renjith. Fibonacci heap, June 2017.

Daniel Stüwe. Verification of fibonacci heaps in imperative hol. Master's thesis in informatics, Technical University of Munic, April 2019.

Robert Endre Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic Discrete Methods*, 6(2):306–318, 1985.

Niki Vazou and Michael Greenberg. How to safely use extensionality in liquid haskell. In *Proceedings of the 15th ACM SIGPLAN International Haskell Symposium*, pages 13–26, 2022.

Niki Vazou, Eric L Seidel, and Ranjit Jhala. Liquidhaskell: Experience with refinement types in the real world. In *Proceedings of the 2014 ACM SIGPLAN Symposium on Haskell*, pages 39–51, 2014.