# 184.726 Advanced Multiprocessor Programming
# Project 10: Wait-Free Linked List

Severin Jäger, 01613004

May 29, 2020

## Contents

# 1 Introduction

The scope of this project was the implementation of the basic version of the wait-free linked list presented in [1]. By exhaustive benchmarking, the advantages, but also the cost, of wait-free lists was examined. As a reference algorithm, the lock-free linked list from [2] was implemented as discussed in the lecture.

# 2 The Wait-Free List Data Strcuture

Lock-free data structures can relatively easily be constructed from the atomic CAS instruction, as the CAS only fails if some other thread has made progress. Achieving wait-freedom requires additional synchronisation between the threads. In list implementation here, this is done by excessive helping between the threads.

Basically, the list maintains an array of all pending list operations. Whenever a thread initiates an contains, add or delete operation, it publishes an `OperationDescriptor` to this array. The key to wait-freedom is that every thread executing some list operation in the following iterates over this array and helps previous operations. The order of the operations in determined by a `phase`.

The wait-free list offers the following operations. It is claimed in [1] that all these operations can be implemented in a wait-free and linearizable manner.

- `contains`
- `add`
- `remove`

All three operations internally use the `search` method, which satisfies the same progress and correctness conditions.

The authors admit that the the wait-free list performs significantly worse than the lock-free list presented by [2]. However, they came up with some optimizations, which limit the extent of helping. Furthermore, the present a fast-past-slow-path algorithm combining the wait-free and the lock-free approach while maintaining wait-freedom. They claim that this algorithm almost reaches the performance of the lock-free data structure.

# 3 Implementation

# 4 Benchmarking

# 5 Summary and Outlook

# References


[1] S. Timnat, A. Braginsky, A. Kogan, and E. Petrank, "Wait-free linked-lists," *ACM SIGPLAN Notices*, vol. 47, pp. 309–310, Sep. 2012.

[2] T. L. Harris, "A pragmatic implementation of non-blocking linked-lists," DISC '01, pp. 300–314, 2001.