

# Parallel Implementation of the A\* Pathfinding Algorithm on a Purely-Functional Programming Language

Prince Bernie B. Colis

Bachelor of Science in Computer Science

John Kenneth S. Lesaba

Bachelor of Science in Computer Science

Jon Ariel N. Maravilla

Bachelor of Science in Computer Science

Karl Frederick R. Roldan

Bachelor of Science in Computer Science

Senior project submitted to the faculty of the

Department of Computer Science

College of Computer Studies, Ateneo de Naga University

in partial fulfillment of the requirements for their respective

Bachelor of Science degrees

---

Project Advisor: Adrian Leo Pajarillo

First Panel Member

Second Panel Member

Third Panel Member

Month Day, 2020

Naga City, Philippines

Keywords: parallel programming, functional programming, graph theory

Copyright 2020, Prince Bernie B. Colis, John Kenneth S. Lesaba, Jon Ariel N. Maravilla, and Karl  
Frederick R. Roldan

The Senior Project entitled

**Parallel Implementation of the A\* Pathfinding Algorithm on a  
Purely-Functional Programming Language**

developed by

**Prince Bernie B. Colis**

Bachelor of Science in Computer Science

**John Kenneth S. Lesaba**

Bachelor of Science in Computer Science

**Jon Ariel N. Maravilla**

Bachelor of Science in Computer Science

**Karl Frederick R. Roldan**

Bachelor of Science in Computer Science

and submitted in partial fulfillment of the requirements of their respective Bachelor of Science degrees  
has been rigorously examined and recommended for approval and acceptance.

**First Panel Member**

Panel Member

Date signed: \_\_\_\_\_

**Second Panel Member**

Panel Member

Date signed: \_\_\_\_\_

**Third Panel Member**

Panel Member

Date signed: \_\_\_\_\_

**Adrian Leo Pajarillo**

Project Advisor

Date signed: \_\_\_\_\_

The Senior Project entitled

**Parallel Implementation of the A\* Pathfinding Algorithm on a  
Purely-Functional Programming Language**

developed by

**Prince Bernie B. Colis**

Bachelor of Science in Computer Science

**John Kenneth S. Lesaba**

Bachelor of Science in Computer Science

**Jon Ariel N. Maravilla**

Bachelor of Science in Computer Science

**Karl Frederick R. Roldan**

Bachelor of Science in Computer Science

and submitted in partial fulfillment of the requirements of their respective Bachelor of Science degrees  
is hereby approved and accepted by the Department of Computer Science, College of Computer  
Studies, Ateneo de Naga University.

**Marianne P. Ang, MS**

Chair, Department of Computer Science

Date signed: \_\_\_\_\_

**Joshua C. Martinez, MIT**

Dean, College of Computer Studies

Date signed: \_\_\_\_\_

# Declaration of Original Work

We declare that the Senior Project entitled

## **Parallel Implementation of the A\* Pathfinding Algorithm on a Purely-Functional Programming Language**

which we submitted to the faculty of the

### **Department of Computer Science, Ateneo de Naga University**

is our own work. To the best of our knowledge, it does not contain materials published or written by another person, except where due citation and acknowledgement is made in our senior project documentation. The contributions of other people whom we worked with to complete this senior project are explicitly cited and acknowledged in our senior project documentation.

We also declare that the intellectual content of this senior project is the product of our own work. We conceptualized, designed, encoded, and debugged the source code of the core programs in our senior project. The source code of third party APIs and library functions used in my program are explicitly cited and acknowledged in our senior project documentation. Also duly acknowledged are the assistance of others in minor details of editing and reproduction of the documentation.

In our honor, we declare that we did not pass off as our own the work done by another person. We are the only persons who encoded the source code of our software. We understand that we may get a failing mark if the source code of our program is in fact the work of another person.

**Prince Bernie B. Colis**

3 - Bachelor of Science in Computer Science

**John Kenneth S. Lesaba**

3 - Bachelor of Science in Computer Science

**Jon Ariel N. Maravilla**

3 - Bachelor of Science in Computer Science

**Karl Frederick R. Roldan**

3 - Bachelor of Science in Computer Science

This declaration is witnessed by:

**Adrian Leo Pajarillo**

Project Advisor

# **Parallel Implementation of the A\* Pathfinding Algorithm on a Purely-Functional Programming Language**

by

Prince Bernie B. Colis, John Kenneth S. Lesaba, Jon Ariel N. Maravilla, and Karl Frederick R.

Roldan

Project Advisor: Adrian Leo Pajarillo

Department of Computer Science

## **EXECUTIVE SUMMARY**

To be filled in later. /\*TODO\*/.

I dedicate this research work to all of humanity.

# ACKNOWLEDGEMENTS

I thank everyone who helped me finish this thesis.



# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Project Context . . . . .	2
1.2	Purpose and Description . . . . .	2
1.3	Objectives . . . . .	3
1.4	Scope and Limitations . . . . .	3
<b>A</b>	<b>Code Listing</b>	<b>4</b>

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

## Introduction

Pathfinding algorithms are methods of finding a path between two vertices in a graph. Most pathfinding problems are concerned with finding the shortest path between two vertices, if there are more than one possible paths. There had been many pathfinding algorithms that had been developed throughout the years such as Minimum Spanning Tree (MST), Prim's Algorithm[14], and the A\* algorithm, which will be used in this paper.[10]

Functional programming is one of the major programming paradigms where computations are done by function composition. Along with this, purely-functional programming is a subparadigm of functional programming where there are no side-effects (e.g, variable mutability). One of the major challenges of parallel programming is controlling the order of execution to prevent *race conditions*, which can often lead to bugs and are hard to maintain. However, since some purely-functional programming languages such as Haskell[1] have no mutability and computations lead to the same result regardless of the order, they are a perfect candidate for writing parallel programs.[9] This research aims to find a parallel implementation of the existing A\* pathfinding algorithm using a purely-functional setting with attention to program performance in terms of time and space complexity. [8, 18] In turn, this helps in the advancement of different programming languages that feature functional programming and lambda expressions, not just purely-functional languages.

## 1.1 Project Context

The A\* pathfinding algorithm is now mostly used as a pathfinding algorithm for video games. While most video games are written in an imperative and object-oriented programming languages such as C#, C++, and JavaScript, it is entirely possible to write video games in functional programming languages using a reactive functional programming paradigm. [5]

Functional programming had been getting more popular recently with more people using ReactJS, TypeScript, PureScript, and Scala. However, imperative programming still dominates the industry, thus more algorithms are written for imperative programs. It can be observed that most algorithm books are written with imperative programming in mind such as *Introduction to Algorithms*[6], *The Algorithm Design Manual*[16], and *The Art of Computer Programming*[12]. Thus, the need for familiarity for functional approaches for some of the most popular algorithms should be discovered, since people are more familiar with imperative approaches and as such, it is more often used in developing video games than functional programming.

One good reason for writing programs in a functional language is that pure functions are easy to reason about and can especially be aided with using a dependently-typed proof assistant such as Coq or Agda.[4, 17, 7]. Programs written in functional programming can be easily reasoned out by reasoning about the smaller components and composing two or more proven functions into a single function and it should also give the correct result. [3]

## 1.2 Purpose and Description

This research aims to utilize the existing A\* pathfinding algorithm [8, 18] and find a way to develop a reasonably-efficient purely-functional implementation of the algorithm using parallel data structures such as STMs or MVars[13].

The A\* Pathfinding algorithm is used heavily in video games, telephone traffic, and other graph traversal problems[10]. This research aims to aid in the development of video games using the functional programming paradigm in the future as video game development is dominated by imperative languages.

## 1.3 Objectives

The research aims to find an efficient parallel implementation of the A\* pathfinding algorithm using a purely-functional programming language such as Haskell. Likewise, concrete comparisons between the number of cores and logical threads will be used to measure the most efficient performance runtime of the algorithm.

The researchers aim to create two programs that will be used in the research. The first program will be used to generate a large-enough maze that will be relatively hard to solve without the aid of computers in a short amount of time. Another program will be written for solving the maze. The program will be a browser-based application that will show the maze graphically and the generated path. This pathfinding program must have an efficient performance while solving the maze. To measure the performance of the program, the application ThreadScope will be used to monitor the thread and core activities while the program is being run. [2]

## 1.4 Scope and Limitations

The research will only cover solvable-mazes as the A\* algorithm does not halt when there is no reachable end goal (e.g, the start vertex and end vertex lie on different components of one graph).[10] Likewise, there will be no generality and all programs will be written in Haskell. Translation to other functional programming languages is not a priority and thus, lambda notation will not be used. Other concurrent data structures besides MVar and Software Transactional Memory will not be utilized.

The concrete implementation and analysis is planned to be tested only on four CPUs such as Intel Core i7-9750H and Intel Core i5-9400. Other CPU architectures are not planned to be tested on.

## Appendix A

### Code Listing

# REFERENCES

- [1] *Haskell programming language*. <https://haskell.org>.
- [2] *Threadscope*. <https://github.com/haskell/ThreadScope>.
- [3] A. ABEL, M. BENKE, A. BOVE, J. HUGHES, AND U. NORELL, *Verifying haskell programs using constructive type theory*, 01 2005, pp. 62–73.
- [4] J. BREITNER, A. SPECTOR-ZABUSKY, Y. LI, C. RIZKALLAH, J. WIEGLEY, AND S. WEIRICH, *Ready, set, verify! applying hs-to-coq to real-world haskell code*, 2018.
- [5] M. H. CHEONG, *Functional programming and 3d games*, (2006).
- [6] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to Algorithms, Third Edition*, The MIT Press, 3rd ed., 2009.
- [7] Y. EL BAKOUNY, T. CROLARD, AND D. MEZHER, *A coq-based synthesis of scala programs which are correct-by-construction*, Proceedings of the 19th Workshop on Formal Techniques for Java-like Programs, (2017).
- [8] Z. ET AL., *Parallelizing a\* path finding algorithm*, International Journal Of Engineering And Computer Science, 6 (2017), pp. 22469–22476.
- [9] K. HAMMOND, *Why parallel functional programming matters: Panel statement*, in Reliable Software Technologies - Ada-Europe 2011, A. Romanovsky and T. Vardanega, eds., Berlin, Heidelberg, 2011, Springer Berlin Heidelberg, pp. 201–205.
- [10] P. E. HART, N. J. NILLSON, AND R. BETRAM, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Transactions of Systems Science and Cybernetics, SSC-4 (1968).
- [11] R. JUNG, J.-H. JOURDAN, R. KREBBERS, AND D. DREYER, *Rustbelt: Securing the foundations of the rust programming language*, Proc. ACM Program. Lang., 2 (2017).
- [12] D. E. KNUTH, *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*, Addison Wesley Longman Publishing Co., Inc., USA, 1997.
- [13] S. MARLOW, *Parallel and Concurrent Programming in Haskell*, O’Reilly Media, Inc., 2013.
- [14] R. C. PRIM, *Shortest Connection Networks And Some Generalizations*, Bell System Technical Journal, 36 (1957), pp. 1389–1401.



- [15] A. SALIGRAMA, A. SHEN, AND J. GJENGSET, *A practical analysis of rust's concurrency story*, 2019.
- [16] S. S. SKIENA, *The Algorithm Design Manual*, Springer, London, 2008.
- [17] A. SPECTOR-ZABUSKY, J. BREITNER, C. RIZKALLAH, AND S. WEIRICH, *Total haskell is reasonable coq*, Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, (2018).
- [18] A. WEINSTOCK AND R. HOLLADAY, *Parallel a\* graph search*.

# VITA

`/*TODO*/` are BS Computer Science student of the Department of Computer Science at the Ateneo de Naga University.