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

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	**Second Panel Member**
Panel Member	Panel Member
Date signed: ______________________	Date signed: ______________________
Third Panel Member	**Adrian Leo Pajarillo**
Panel Member	Project Advisor
Date signed: ______________________	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

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 is more than one possible path. There had been many pathfinding algorithms that had been developed throughout the years such as Minimum Spanning Tree (MST), Prim's Algorithm[20], and the A* algorithm, which will be used in this paper.[12]

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 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.[11] 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. [10, 24] In turn, this helps in the advancement of different programming languages that feature functional programming and lambda expressions when it comes to purely-functional algorithms and data structures.

1.1 Project Context

The A* pathfinding algorithm is mostly used as a pathfinding algorithm for video games. While most games are written in an imperative and object-oriented language such as C#, C++, and JavaScript, it is possible to write video games in a functional languages using a reactive functional programming approach.[6] Though, functional programming had been getting more popular recently, imperative programming still dominates the industry, thus more algorithms are designed with imperative programs in mind.[7, 22, 15] Thus, the need for functional approaches to familiar algorithms should be addressed in order for functional programming to be used more widely in video games. However, algorithms designed for imperative programming does not translate as well when implemented in a functional style such as quicksort having a worst-case space complexity of $O(n)$ for the imperative approach while its functional version would have a worst-case space complexity of $O(n^2)$.[1]

The advantages of functional programming, however, lies with its *referential transparency.* In imperative styles, a variable x can change their values over time and may be difficult to track. This is not the case with functional approaches wherein functional expressions always have the same definition throughout the runtime of the program.[14, 11] As such, it is easier to mathematically prove the correctness of programs in the functional style, especially when combined with proof assistants such as Coq or Agda.[4, 23, 9] Splitting functions into smaller functions and reasoning about those smaller components much like lemmas would mean that functions would be modular and composed of proven subfunctions or subroutines.[3, 13]

Due to the nature of *referential transparency* in functional programming languages, functional languages are an excellent candidate for parallel programming. In imperative programming, writing parallel programs would mean that there are non-deterministic separate threads that needs to be tracked to ensure that it is correct. Since there is a shared variable, multiple threads modifying the shared variable could mean bugs. However, since functional languages don't have a mutable states and there are reduced instances of shared variables abstracted from the programmer, the order of execution of pure functions does not matter as the program will still yield the same results.[14]

[1] Recursion in Haskell copies the entire subarray instead of mutating the original array elements, as opposed to the quicksort implementation of Hoare. Thus, one can argue that Haskell's quicksort is not a *true* quicksort.

1.2 Purpose and Description

This research aims to utilize the existing parallel A* pathfinding algorithm [10, 24] and find a way to develop a reasonably-efficient purely-functional implementation of the algorithm using parallel data structures such as STMs or MVars[18].

The A* Pathfinding algorithm is used heavily in video games, telephone traffic, and other graph traversal problems[12]. This research aims to aid in the development of video games and other uses where a parallel A* algorithm would be beneficial using the functional programming paradigm in the future as video game development is dominated by imperative languages.

1.3 Objectives

The main objective of the research is to find an efficient parallel purely-functional implementation of the A* pathfinding algorithm. The research will be done mostly in Haskell with some exceptions. Likewise, concrete comparisons between the number of cores and logical threads will be used to measure the most efficient performance runtime and space complexity of the algorithm.

The researchers aim to complete the following specific tasks:

- Write a *generator* that will generate an arbitrary-sized maze. The maze should be relatively hard to solve without the aid of computers in a short amount of time.[5]
- And a *solver* program that will be written in Haskell, a lazy purely-functional programming language, for translating the output of the generator to a graph.[1]
- The solver program should have a web-based user interface where the researchers can view the maze and how the solver program was able to solve it correctly.
- Performance of the solution shall be measured by using ThreadScope 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 the graph).[12]

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 Trasnactional Memory will not be utilized. The implementation of the graph that the research will use will be Algebraic Graphs.[19]

The concrete implementation and analysis is planned to be tested only on four CPUs such as Intel Core i7-9750H and AMD Ryzen 5 3500x. Other CPU architectures are not planned to be tested on.

Chapter 2

Review of Related Systems and Related Literature

We would write something on this section about the A* pathfinding algorithms and other parallel graph pathfinding algorithms written on both imperative and functional languages.

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] J. Buck, *Mazes for Programmers*, The Pragmatic Programmers, 2015.

[6] M. H. Cheong, *Functional programming and 3d games*, (2006).

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, The MIT Press, 3rd ed., 2009.

[8] L. Dhulipala, G. E. Blelloch, and J. Shun, *Theoretically efficient parallel graph algorithms can be fast and scalable*, in Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA '18, New York, NY, USA, 2018, Association for Computing Machinery, p. 393–404.

[9] 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).

[10] Z. et al., *Parallelizing a* path finding algorithm*, International Journal Of Engineering And Computer Science, 6 (2017), pp. 22469–22476.

[11] 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.

[12] 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).

[13] J. Hughes, *Why Functional Programming Matters*, Computer Journal, 32 (1989), pp. 98–107.

[14] M. Kesseler, *The Implementation of Functional Languages on Parallel Machines with Distributed Memory*, PhD thesis, Radboud University Nijmegen, 1996.

[15] D. E. Knuth, *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*, Addison Wesley Longman Publishing Co., Inc., USA, 1997.

[16] H.-W. Loidl, F. Rubio, N. Scaife, K. Hammond, S. Horiguchi, U. Klusik, R. Loogen, G. J. Michaelson, R. Peña, S. Priebe, Á. J. Rebón, and P. W. Trinder, *Comparing parallel functional languages: Programming and performance*, Higher-Order and Symbolic Computation, 16 (2003), pp. 203–251.

[17] A. Lumsdaine, D. P. Gregor, B. Hendrickson, and J. W. Berry, *Challenges in parallel graph processing*, Parallel Process. Lett., 17 (2007), pp. 5–20.

[18] S. Marlow, *Parallel and Concurrent Programming in Haskell*, O'Reilly Media, Inc., 2013.

[19] A. Mokhov, *Algebraic graphs with class (functional pearl)*, SIGPLAN Not., 52 (2017), p. 2–13.

[20] R. C. Prim, *Shortest Connection Networks And Some Generalizations*, Bell System Technical Journal, 36 (1957), pp. 1389–1401.

[21] F. Rabhi and G. Lapalme, *Algorithms; A Functional Programming Approach*, Addison-Wesley Longman Publishing Co., Inc., USA, 1st ed., 1999.

[22] S. S. Skiena, *The Algorithm Design Manual*, Springer, London, 2008.

[23] 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).

[24] A. Weinstock and R. Holladay, *Hash distributed a* distributed termination detection path reconstruction*, 2016.

VITA

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