CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Survey of Imperative Style Turing Complete proof techniques and an application to prove Proteus Turing Complete

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in
Computer Science

By

Isaiah Martinez

December 2024

The thesis of Isaiah Martinez is approved:

Kyle Dewey, PhD., Chair	Date
John Noga, PhD.	Date
Maryam Jalali	Date

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacinia odio vitae vestibulum vestibulum. Cras venenatis euismod malesuada. Maecenas vehicula felis quis eros auctor, sed efficitur erat suscipit. Curabitur vel lacus velit. Proin a lacus at arcu porttitor vehicula. Mauris non velit vel lectus tincidunt ullamcorper at id risus. Sed convallis sollicitudin purus a scelerisque. Phasellus faucibus purus at magna tempus, sit amet aliquet nulla cursus.

Table of Contents

Signature Page ii

Acknowledgements iii

List of Figures vi

List of Tables vii

List of Illustrations viii

Abstract ix

1 Introduction 1

1.1 Turing Machines 1

1.1.1 Oracles 1

1.1.2 Universal Turing Machines 2

1.1.3 The Church-Turing Thesis 2

1.2 Turing Complete 2

2 Different Proof Techniques for demonstrating Turing Completeness 4

2.1 Turing Complete Proofs Overview 4

2.2 Computer Engineering 4

2.2.1 Gates & Wiring 4

2.3 Computer Science 4

2.3.1 State Machines 4

2.3.1.1 Formal Technical Writing 4
2.3.1.2 Gamified Writing 4
2.3.2 Software Implementation 5
2.3.2.1 Conway's Game of Life 5
2.3.2.2 Rule 110 5
2.3.2.3 Calculator with Store Value 5
2.3.3 Interpreter for TC lang 5
2.3.3.1 Brainfuck 5
2.3.3.2 C 5
2.4 Mathematics 5
2.4.1 Lambda Calculus 5

3 Proteus is Turing Complete 6
3.1 The Proteus Grammar 6
3.2 Initial Thoughts based off of the grammar 6
3.3 Proof Outline 6
3.4 Proof v1 6
3.5 Implementing Rule 110 6

4 Conclusion 7

List of Figures

This list must reference the figure, page it appears, and subject matter.

List of Tables

This list must reference the table, page it appears, and subject matter.

List of Illustrations

This list must reference the illustration, page it appears, and subject matter.

Abstract

TITLE GOES HERE

By

Isaiah Martinez

Master of Science in Computer Science

Abstract which will cover the contents of the entire paper in less than 350 words or so. 1.5 pgs double spaced

state research probem briefly describe methods and procedures used in gathering data or studying the problem give a condensed summary of the findings of the study

Chapter 1

Introduction

1.1 Turing Machines

Alan Turing is generally considered the father of computer science for his numerous contributions including: formalization of computation theory, algorithm design, complexity theory, as well as creating the idea of the Turing Machine. A Turing machine can be described as a machine/automata that is capable of performing operations towards some desired goal given an input. In a sense, it was designed to be capable of performing any single computable task, such as addition, division, concatenating strings, rendering graphics, etc. TMs are at the highest level of computational power, i.e. capable of handling any computation.

INSERT DIAGRAM OF A SIMPLE TURING MACHINE THAT PERFORMS BINARY ADDITION

1.1.1 Oracles

Of course there also exists the Oracle, sometimes called Turing Machines with an oracle, which is capable of solving problems that TMs cannot. It does so by having the ability to respond to any given problem from the TM it is connected to. For example, the oracle would be able to solve the Halting Problem for the associated Turing Machine, but not the Halting Problem in general for all Turing Machines.

INSERT DIAGRAM OF A TM W/ AN ORACLE SOLVING THE HALTING PROBLEM

The reason the oracle is not considered more powerful is because in practice (i.e. reality), is

because there is no such all-knowing source to retrieve information from. As a result, I will look only at TMs for the rest of the thesis.

1.1.2 Universal Turing Machines

A simple abstraction of the standard TM is a Universal Turing Machine. A UTM is capable of solving any computable problem, given the exact process/rules of the TM that will solve it. In essence it is a machine that is not hard-coded with what to perform when given input. The UTM will read the input, and respond based on the rules given. As a result, the UTM is equivalently as powerful as a TM. The only functional difference is the usability of the UTM towards a larger number of problems as opposed to the TM being created for a singular problem.

INSERT DIAGRAM OF A SIMPLE UNIVERSAL TURING MACHINE THAT PERFORMS BINARY ADDITION basically it just takes another input which are the instructions

1.1.3 The Church-Turing Thesis

According to the Church-Turing Thesis, every effectively calculable function can be computed by a Turing Machine. As explained by Robin Gandy, this idea was to show that the limits of computation of reality are bounded by TMs. Thus any automata that violates it, such as an Oracle Machine, would be capable of calculating a non-computable function. In his words, displaying ”Free Will”.

[INSERT REFERENCE TO GANDY 1980 PAPER PG 123-148].

Generally speaking however, the consensus is that the Church-Turing Thesis is widely accepted as fact, and no such oracle machines exist as previously stated.

1.2 Turing Complete

Turing Complete is a closely related term when discussing Turing Machines. For a system to be Turing Complete, it must be capable of performing any computation that a standard TM can perform. An equivalent description would be that for a system to be TC, it must simulate a UTM.

By transitivity, if any system is proven to be TC, then it must be equivalent in power to all other systems that are TC. Thus, all are TMs, which in turn means they are all equivalently the most powerful computation machine.

Some classical/practical examples of TC

Link some funny/interesting examples of TC that have been proven

In this thesis, we will look closely at the several methods to showing that a given system is TC.

Chapter 2

Different Proof Techniques for demonstrating Turing Completeness

2.1 Turing Complete Proofs Overview

We will be exploring the different approaches to demonstrate TC

All proofs are equivalent in goal. The difference lies in their usability for specific domains.

2.2 Computer Engineering

2.2.1 Gates & Wiring

NOR, ADDER, OR, NOT gates

2.3 Computer Science

2.3.1 State Machines

2.3.1.1 Formal Technical Writing

$\lambda\gamma\tau$ etc. being used to describe TC

2.3.1.2 Gamified Writing

$\lambda\gamma\tau$ etc. being used to describe TC, but based on a very verbose and interactive style

2.3.2 Software Implementation

Demonstrate an example of a problem/scenario which shows TC.

2.3.2.1 Conway's Game of Life

Classic example to implement

2.3.2.2 Rule 110

Simpler example to implement

2.3.2.3 Calculator with Store Value

More challenging to implement than the previous two.

2.3.3 Interpreter for TC lang

Implement an interpreter for a known TC language in a given language.

2.3.3.1 Brainfuck

Describe Brainfuck

2.3.3.2 C

Is C TC?

Show how this

2.4 Mathematics

2.4.1 Lambda Calculus

Whatever the fuck this is, is TC

Chapter 3

Proteus is Turing Complete

This section will describe how we will construct 1+ proofs for showing that Proteus is TC.

3.1 The Proteus Grammar

List out the grammar and describe how it can be read, line by line. Probably have a small example of a program that shows most of (maybe all) of the functionality with Proteus

3.2 Initial Thoughts based off of the grammar

List out some ideas that led to the idea that Proteus could be TC

3.3 Proof Outline

Theorems that would be used Steps for proof outline

3.4 Proof v1

Formal proof that follows the outline

3.5 Implementing Rule 110

Implementation of Rule 110 that is a demonstration of TC.

Chapter 4

Conclusion

Succinctly describe what was the several techniques. Compare and constrast them? Perhaps a discussion section?

Bibliography

[1] Baker, N. 1966, in Stellar Evolution, ed. R. F. Stein & A. G. W. Cameron (Plenum, New York) 333