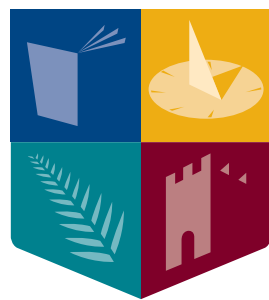

Final Year Project Report

Formalising Alternative Models of Computation in Isabelle



**Maynooth
University**

National University
of Ireland Maynooth

Dara MacConville | 17377693

A thesis submitted in partial fulfilment of the requirements for the
B.Sc. Computational Thinking

5 Credits

Advisor: Dr. Philippe Moser

*Department of Computer Science
Maynooth University, Ireland*

April 13, 2019

ABSTRACT

The goal of this project was to formalise and implement alternative models of computation inside the proof assistant Isabelle, and then to make use of this to assist in providing definitions on and proving various results about these models. In particular it focuses on Cellular Automata, in both one and two-dimensional variants, and with differing topologies.

CONTENTS

1	Introduction	1
1.1	Topic addressed in this project	1
1.2	Motivation	1
1.3	Approach	1
1.4	Metrics	2
2	Technical Background	3
2.1	Topic Material	3
2.2	Technical Material	3
	Bibliography	4

LISTS OF FLOATS

LIST OF TABLES

LIST OF FIGURES

LIST OF LISTINGS

INTRODUCTION

Cellular Automata (CA) are a very simple model of computation. There are many variations and extensions of them, and some like Conway's Game of Life and Rule 110 are known to be Turing Complete. Isabelle is a proof assistant that can be used for anything from mathematical proofs to formal verification of software properties.

1.1 TOPIC ADDRESSED IN THIS PROJECT

This project looks at formalising models of Cellular Automaton in Isabelle to help deal with the complexity that comes with trying to mathematically prove results about them. In addition to providing a formalisation of six different variants of CA, this project provides definitions of important properties they may have, and proves some lemmas and theorems necessary to work with them.


1.2 MOTIVATION

It can be very difficult to work with high level concepts while still being rigorous, and theoretical Computer Science is a very abstract and mathematical discipline that requires exactly that. One of the main concepts used in theoretical CS is the Turing Machine. While it provides an excellent way of approaching computation that allows for conceptually easy high level proofs, being very strict and formal in these can prove difficult.

However the Turing Machine is not the only available model in CS. One of the goals of this project was to look at *alternative* concepts that achieve the same thing, to help better appreciate the benefits and drawbacks. Cellular Automata were chosen as they strike a good balance between simplicity and complexity, as they have a very simple idea, but come with additional geometric aspects to consider.

The motivation to use a proof assistant rather than traditional pen and paper proofs comes from a desire to use the exactness of computers to cope with issues that stem from highly layered and detailed concepts.

1.3 APPROACH

 Isabelle was chosen as the language and framework to implement these models in. This is due to it being very high level, and having advanced tools available to ease the creation process. These include the multitool command **sledgehammer** which invokes several Automatic Theorem Provers and Satisfiability-Modulo-Theories solvers, and **nitpick** which provides counter-examples to statements. These take the burden of large manual proofs off the programmer, while still encouraging simple and understandable definitions, as these are more easy to automatically prove results on.

Isabelle also has a powerful and expressive type system, and the decision was taken to directly make use of this in the CA definitions created. This involved making each kind of CA explicitly into a type, to make use of automatic type checking and other benefits. This contrasts with a more mechanical approach taken in a paper [1] that also works on implementing Turing Machines and other computability concepts in Isabelle. One reason for this difference in approach is their work involves the need to convert between different models, for instance Abacus Machine to Turing Machine, whereas in this project no conversions are done. Additionally the design of such “machines” lend themselves to being built in such a way. However it was not possible to entirely avoid that way of thinking with CA either, in particular in the two dimensional case.

1.4 METRICS

It can be difficult to evaluate exactly how well a definition has been translated from the informal to the formal. It could be possible to have a very precise and elegant program that doesn’t actually capture the intuitive content of the informal idea. As such the definitions written here were evaluated in two main ways:

1. Understanding: Does reading the program lead to the same understanding as reading the original definitions? If so then it is capturing the same spirit as the original, even if the means of execution may differ slightly or significantly.
2. Functionality: As CA are systems that can be run, it is necessary to run them on example programs and see the correct results are achieved. Speed is not a trait that is sought out but accuracy is of course essential.

Also taken into account is the ability to state properties of these systems, and the ease or difficulty of proving results about them.

TECHNICAL BACKGROUND

2.1 TOPIC MATERIAL

2.1.1 FORMALISATIONS IN COMPUTABILITY THEORY

As mentioned before, this is certainly not the first formalisation of computability theory concepts inside of proof assistants. Other works however have focused on more traditional models or with certain specific proofs in mind. These include λ -calculus in Coq [2], partial recursive functions in Lean [3], and the aforementioned Turing and Abacus machines in Isabelle [1]. Other than choices of model and language, the key difference between those works and this project is the end goal. The three mentioned above all set about to formalise computability theory from the perspective of their chosen model or models. This project works with just Cellular Automata and what results can be specifically shown about them.

Additionally the approach in [1] often involves specifying certain exact indices in lists and having to manipulate these numbers. This obviously has its technical benefits and is necessary when working with TMs, but the desire in this work was to take a “wholemeal” functional programming approach to the construction where possible.

The existing research that comes closest to both the goals and execution of this project, is a formalisation of CA in Coq for the purposes of verifying a result about the firing squad problem [4]. Their work bases the CA over the naturals \mathbb{N} , and includes an explicit time parameter. As well as that, their transition function is based more around individual cells, and the properties they define are all based around eventually deriving the proof. This project certainly has similarities, but trades the depth of [4] for breadth, and makes use of definitions that try to be more compact. For instance the time parameter is more external to the cellular automata, and the varieties of different CA are defined with different mechanisms.

2.1.2 CELLULAR AUTOMATA

The concept of CA has been around since the 1940s and were popularised in the 70s with Conway’s Game of Life. However a lot of the names and terminology associated with them comes from Stephen Wolfram [5]. His work is not entirely formal, but others have provided their own formalisations of definitions he put forward [6]. This was especially useful as an aid in translating the some of the classifications of CA into Isabelle, even if their formalisation differed in many other ways.

2.2 TECHNICAL MATERIAL

BIBLIOGRAPHY

- [1] Jian Xu, Xingyuan Zhang, and Christian Urban. “Mechanising Turing Machines and Computability Theory in Isabelle/HOL”. In: *Interactive Theorem Proving*. Ed. by Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 147–162. ISBN: 978-3-642-39634-2.
- [2] Yannick Forster and Gert Smolka. “Weak call-by-value lambda calculus as a model of computation in Coq”. In: *International Conference on Interactive Theorem Proving*. Springer. 2017, pp. 189–206.
- [3] Mario Carneiro. “Formalizing computability theory via partial recursive functions”. In: *arXiv preprint arXiv:1810.08380* (2018).
- [4] Jean Duprat. “Proof of correctness of the Mazoyer’s solution of the firing squad problem in Coq”. In: (2002).
- [5] Stephen Wolfram. *A New Kind of Science*. English. Champaign, IL: Wolfram Media, 2002. ISBN: 9781579550080;1579550088;
- [6] Karel Culik and Sheng Yu. “Undecidability of CA Classification Schemes”. In: *Complex Systems* 2 (1988).
- [7] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Jan. 2002. doi: [10.1007/3-540-45949-9](https://doi.org/10.1007/3-540-45949-9).