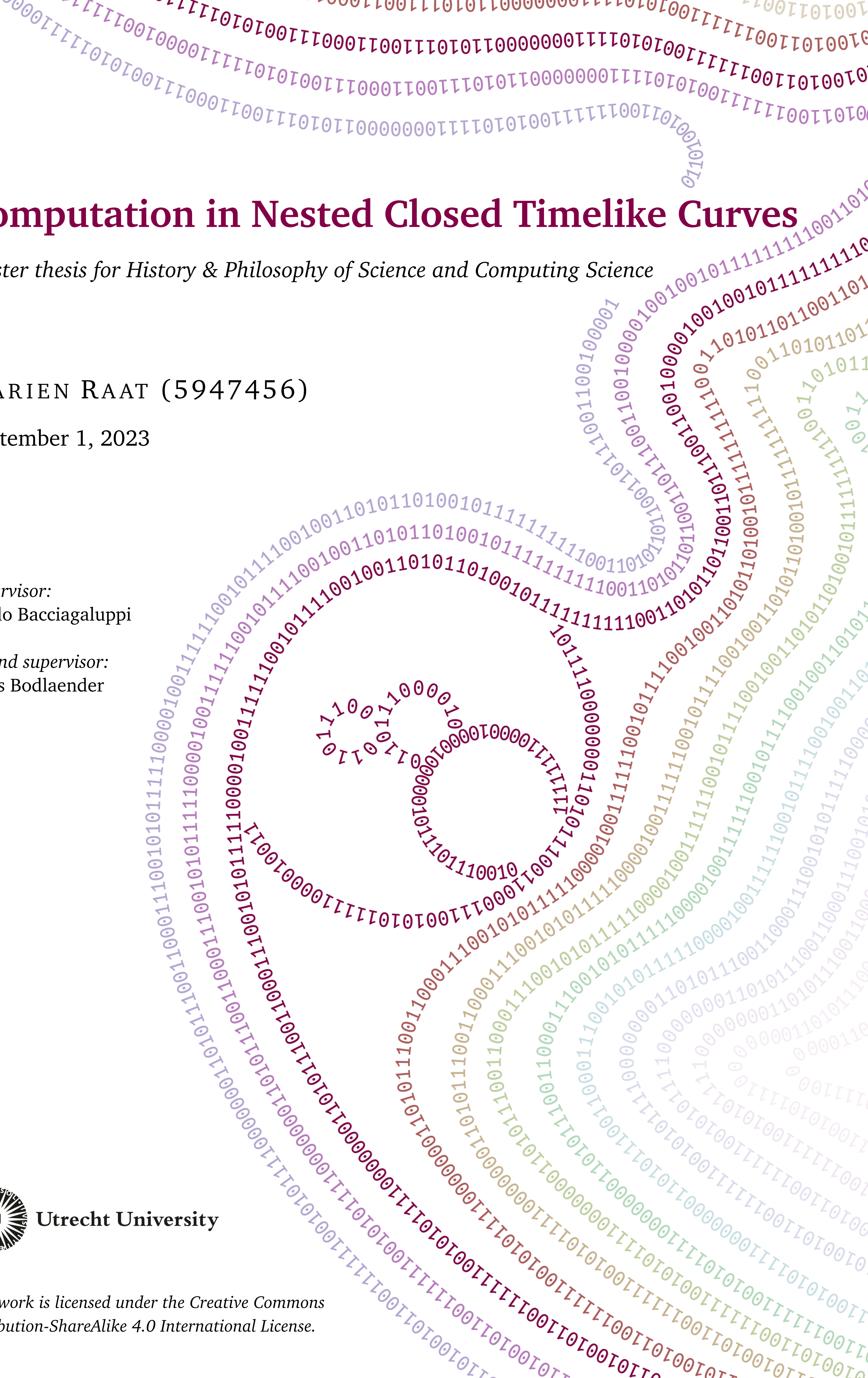


Computation in Nested Closed Timelike Curves

Master thesis for History & Philosophy of Science and Computing Science

MARIEN RAAT (5947456)

September 1, 2023

Supervisor:
Guido Bacciagaluppi

Second supervisor:
Hans Bodlaender

Utrecht University

Lay summary

There are some problems that computers cannot solve. This is true not only for many societal and practical problems, it has in fact been proven for mathematical problems. Turing showed in 1936 that some mathematical problems are so hard that computers can never solve them. In the same paper, he described the Turing machine model for computers and the halting problem, which can not be solved on these machines.

Our current computers are as powerful as the Turing machines and the Turing machine model is still widely used in computer science to describe what computers can and cannot solve. It is often assumed that this means that computers can never solve the halting problem, but this is only the case if no computers can be made that are more powerful than Turing machines.

Figure 1: Alan Turing

Luckily, there is some hope that this might be possible. The Turing machine model is based on classical intuitions of how our universe works. But the new physical theories of the 20th-century—quantum mechanics and general relativity—have shown that universe fundamentally behaves differently from how it seems to work in ordinary situations.

It may be possible to use this behaviour to create computers that are more powerful than Turing machines. Researchers in the field of quantum computation have made great progress in the last three decades in describing computers that could harness quantum effects for computation. These computers have the potential to solve certain problems much more efficiently than current computers. But it turns out that these computers cannot solve any new problems, which means they also cannot solve the halting problem.

However, the effects of general relativity are potentially even more powerful. In this theory, matter can curve spacetime. This might make it possible to travel back in time through a time loop when the spacetime is properly curved. In 2016, Aaronson et al. showed that under certain assumptions about these time loops, it would be possible to create a computer that is stronger than a Turing machine. Such a computer could solve the halting problem, which is impossible on our current computers.

But even these computers still would be unable to solve many problems. Aaronson et al. also showed that these computers cannot solve any problem that is harder than the halting problem. Does this mean that these harder problems are in fact impossible to solve?

In this thesis, I argue that much harder problems can be solved on computers in certain situations. I show that what problems a computer can solve depends on how many time loops are in the universe and how these time loops are connected. With every time loop that is appropriately placed into another time loop, it becomes possible to make an even stronger computer that can solve even more and harder problems. When it is possible to create more time loops during the calculation of a computer, it would be possible to solve a huge amount of harder and harder problems that are all impossible to solve with our current computers.

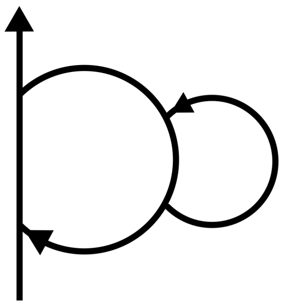


Figure 2: Illustration of two time loops within each other

This thesis gives more information about what can and cannot be calculated in our universe under what conditions. It shows that our current computers might not be the most powerful computers that can exist. It also shows how the power of a computer is related to how many time loops it has access to. And it once again emphasises that physical laws are highly relevant to computational questions. After all, computation is always a physical process as well as a mathematical one.

Preface

Combining the master degrees of History and Philosophy of Science and Computing Science has been an interesting and rewarding mix of very different content as well as very different methods. My Computing Science degree has taught me many highly technical and theoretical results and techniques. I have been especially captured by computability theory and the parsing of formal languages. My History and Philosophy of Science degree on the other hand has taught me to think critically about philosophical, historical and societal questions and helped me discover a passion for the philosophy of physics.

Philosophy of science and computing science are not straightforwardly combined into one thesis. The History and Philosophy of Science degree does not focus much on computing science and the Computing Science degree on the other hand does not focus on historical or philosophical questions. But as I started gravitating towards the philosophy of physics, I saw an important connection to computing. Computers are physical machines and computing is a physical process, so how does the philosophy of physics influence computing? Whereas for quantum mechanics, this question has been considered a lot, for our other fundamental theory, general relativity, this question has not been the focus of much research.

I hope that with this thesis I have contributed a little to answering this question. But I would not have been able to write this thesis without the help of many other people. I want to thank my thesis supervisor Guido Bacciagaluppi and my second reader Hans Bodlaender for their support in the process as well as with technical problems. Scott Aaronson, Sabee Grewal and Ryan O'Donnel for insightful discussions about fixing the proof in the paper by Aaronson et al. My family and friends for their support. All the koele kikkers of the HPS room—Elise, Menno, Elian, Anna, Hilbrand, Nils, Ori and more—for making working on my thesis highly enjoyable. Jochem and Menno for their help with and ideas for the layout of this thesis. And lastly anyone else that supported me, listened to my (temporary) frustrations or endured a, possibly slightly too long, but enthusiastic explanation of my thesis.

Marien Raat
September 1, 2023

Contents

1 | Introduction

“Well, then,” I said, “either I am a lunatic, or something just as awful has happened. Now tell me, honest and true, where am I?”

A Connecticut Yankee in King Arthur’s Court
Mark Twain

Nowadays, our computers seem to be able to do almost anything. A tiny smartphone in your pocket can calculate, communicate over the internet, convert speech to text and even generate pictures based on a textual prompt. However, already before the first digital computer was created, it was known that there are some mathematical problems that computers cannot solve. This was shown by Turing in 1936 and applies to all of our everyday computers.

All our current computers—phones, laptops, servers, supercomputers—are classical computers, based on the architecture described by von Neumann in 1945. But these are not the only type of computers that are possible. In the 1980s the field of quantum computing emerged, and it has gained traction over the last decades. This may result in usable quantum computers in the future, which could solve certain problems exponentially faster than our current computers. But sadly, these quantum computers would not be able to solve any new mathematical problems, that our current computers cannot solve [1].

Let us not give up hope on these problems yet however! Apart from quantum mechanics, another physical theory has revolutionised our ideas about the world. General relativity has taught us that space and time are interconnected and that this spacetime is curved by matter and energy. What implications does this other theory have for computing?

It turns out that in general relativity, there is the possibility of spacetime curving so much that it loops, potentially allowing for time travel to the past. Aaronson et al. have shown that under one model of these loops, computers could solve some of these problems that our current computers are fundamentally unable to solve [2]. At last there is hope that these problems can be solved after all, even if this would require exotic spacetime curvature.

But in this model that Aaronson et al. use, there are again limits to computation. Some of the problems that can be formulated are still unsolvable by these computers, even if they have access to these time loops. In this thesis I show that with the appropriate spacetime configuration, it is possible to solve even more problems. All that is needed for this, is for there to be more time loops in the time loops.

In this chapter I will give a very brief background of the theories needed to understand my result. In Chapter 2 more background on time travel and hypercomputing will be given. Then in Chapter 3 I will introduce a physical model that forms the basis for the rest of my research, on which I will build a computational model in Chapter 4. In Chapter 5 the computational power of this model will be proven. And lastly in Chapter 6 I will give a conditional result on the limits of computation in my models, indicating what problems still cannot be solved.

1.1 Background

My thesis builds on research in computer science as well as in the (philosophy of) physics. So to understand the research, some knowledge is required of both fields. In this section, I will give a short introduction to the concepts that are most important to my thesis.

1.1.1 Computability theory

One of the questions in computer science is the question of what problems can and can not be solved by a computer. For this, we need a model of computation, the Turing machine. Using the Turing machine and related oracle machines, we can establish a hierarchy of computable problems. The introduction to these concepts in this section should be enough to follow the argumentation in this thesis, for a more comprehensive introduction to computability theory, I refer the reader to the book “Turing Computability: Theory and Applications” by Soare [3].

Turing machines

Turing machines were introduced by Turing in 1936 in his famous paper “On computable numbers, with an application to the Entscheidungsproblem” [4]. In the previous years, Church, Kleene and Gödel had been trying to find a formal definition to fit the intuitive concept of intuitively computable. Church and Kleene had focused their efforts on formal recursive functions that could be defined, but it was hard to connect these mathematical models to the process of computing [3, p. 228–35]. The Turing machine however was more similar to a physical machine, inspired by a mechanical typewriter, and Turing gave a strong argument why it could compute everything that was intuitively computable.

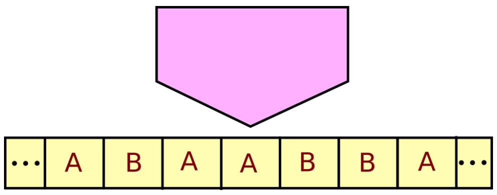


Figure 1.1: Turing machine with finite state machine head, infinite tape and symbols on the tape

A Turing machine has an infinite tape subdivided into cells. On each cell, there can be a symbol from a finite alphabet. We can imagine this as an infinitely long piece of paper, divided horizontally into squares. The machine has a reading head that can move over this tape, one square left or right at each step. The machine also has a finite set of internal states. At each step, it reads the square of the tape under the reading head and based on its internal state and the square it reads, it writes a symbol on the square under the reading head and moves left or right.

A Turing machine also has a specific internal state designated as an initial and one as a halting state. This means that the machine always start in the initial state and is done when it reaches the halting state. It can then be considered to have outputted or returned whatever is on (a specific part of) the infinite tape.

Formally, this means a Turing machine is defined by an alphabet of symbols A, a set of internal states S, including one $s_{\mathtt{initial}} \in S$ and one $s_{\mathtt{halting}} \in S$, and a partial function $t : A \times S \to A \times S \times \{L, R\}$. The machine then in each step reads the symbol under the reading head a and transitions based on that and its current internal state s using $t(a, s)$. It then writes the symbol returned by t, moves to the internal state returned by t and moves left or right based on t. It keeps doing this until it reaches $s_{\mathtt{halting}}$.

Turing argued that this machine can do anything that is intuitively computable. In 1936 there were no digital computers yet, so the main intuition of computing was a person following a defined set of steps to compute a result. Turing argued that such a person would be able to compute the same things as a Turing machine. They would have a finite amount of internal states (or they would become so close they would be indistinguishable), they would only be able to recognise a finite number of different symbols in action, and they would only be able to read and move through their notes at a finite speed [4, p. 74–77]. But Turing also pointed out that any computing machine would have the same limitations [4, p. 79].

Turing proved that there are some problems that a Turing machine cannot compute. For example, the halting problem cannot be computed by a Turing machine. The halting problem is the problem of whether any given Turing machine always eventually reaches a halting state. It is easy to see that this is not computable on a Turing machine by contradiction. Imagine a Turing machine H exists that could tell whether a machine halts. Now we construct a Turing machine C that runs H on itself and stays in a non-halting state forever if it returns true and halts if it returns false. We should be able to construct this machine if H exists, but its output on C would always be wrong. If it returns true, it in fact does not halt and if it returns false, it does. So H cannot exist and the halting problem is not computable on a Turing machine.

Turing convincingly argued that this model formalised the intuitive concept of computation and the Turing machine is still the most used model to reason about computation. Other models, like the lambda-calculus, have been shown to be able to compute exactly the same problems as the Turing machine. In the context of this thesis, it is important to understand that Turing's demonstration that the Turing machine can compute anything that is computable, relies on the fact that the tape behaves according to Newtonian mechanics. When considering general relativity and quantum mechanics, which have superseded Newtonian mechanics, this no longer holds. This is why I will use a different model of computation in this setting.

Oracle machines

Turing machines are one way to judge the computability of problems. Can the problem be solved in finite steps by a Turing machine? Then we call it computable and otherwise we do not. But in this thesis I will go beyond that and for this we need a hierarchy of computability. To construct this hierarchy, we will use oracle machines.

Oracle machines were originally introduced by Turing [5, p. 173] as an aside in his PhD thesis. But it was Post, who later used these oracle machines to create a hierarchy of computability [3, p. 243–45].

Oracle machines are like Turing machines, with the addition that they have access to an oracle. An oracle can solve a specific problem, for example one might have a *prime oracle*, that can tell for any number whether it is a prime. The oracle machine with a *prime oracle* could then query this oracle at any step to see if a number is a prime.

But a Turing machine can already compute whether a number is a prime, so oracle machines only get interesting when they have an oracle for a non-computable problem. For example, one could have an oracle machine with a *halting oracle*. That machine could solve the halting problem, by simply using its oracle, so that machine would be more powerful than a Turing machine.

Turing degrees

In the theory of Turing degrees, sets of natural numbers represent decision problems. To construct a set of natural numbers for a decision problem, we encode each instance of the problem as a natural number. Then the set representing the problem is all the numbers that represent YES-instances of the problem. The theory of Turing degrees describes what decision problems oracle machines can decide.

Each decision problem has a corresponding Turing degree, which is a set of decision problems that have the same level of computability. This means that all the decision problems are Turing equivalent, they can be decided on the same oracle Turing machine.

Definition 1 (Turing reducibility) Decision problem A is Turing reducible (notated $A \leqslant_T B$) to decision problem B iff A can be decided by an oracle machine with an oracle for problem B.

Definition 2 (Turing equivalence) Decision problems A and B are Turing equivalent (notated $A \equiv_T B$) iff $A \leqslant_T B$ and $B \leqslant_T A$.

Definition 3 (Turing degree) A Turing degree **a** is a set of decision problems, such that for each problem A and B in **a** it holds that $A \equiv_T B$. The Turing degree of a problem C is notated as $[C]$.

Definition 4 (Partial order on Turing degrees) A Turing degree $[A]$ is less than a Turing degree $[B]$ (notated $[A] \leqslant [B]$) iff $A \leqslant_T B$.

This partial order gives us our first Turing degree, which we will call **0**. It is the least element in the set of all Turing degrees and it contains all problems that are decidable on a Turing machine without an oracle. In other words, **0** is the set of computable problems. We can derive higher sets from **0** using Turing jumps.

Definition 5 (Turing jump) The Turing jump from Turing degree $[A]$ (notated $[A]'$) is the set of problems Turing equivalent to the corresponding halting problem of A. Multiple subsequent Turing jumps are notated as $[A]^{(n)}$. For example $[A]'''$ can equivalently be written as $[A]^{(3)}$.

Definition 6 (Corresponding halting problem) The corresponding halting problem of problem A is the problem of deciding whether a given oracle Turing machine with an A oracle will halt.

Turing's logic about the uncomputability of halting problems holds for oracle machines as well. So the halting problem of a problem always has a strictly larger Turing degree. So the Turing degree after a Turing jump is strictly larger. To easily refer to specific halting problems, I will sometimes call the halting problem of a problem in $0^{(n-1)}$ the nth halting problem

1.1.2 General relativity

General relativity is a physical theory published by Einstein in 1915. It generalises special relativity and is our current best theory of gravity. In the theory, spacetime can be curved and this curvature is influenced by the energy and matter that is present at each spacetime location. The spacetime structure is given by solutions of Einstein's field equations.

In 1949, Gödel found a solution of Einstein's equations in which the spacetime curves so much, that it forms a loop or closed timelike curve (CTC from now on). This loop could potentially allow an observer to travel to the past.

This possibility of CTCs showing up in general relativity plays a central role in this thesis. It is these CTCs that could potentially give computers more computational power. However, it is unclear whether the situations in which these situations would show up are possible. One proposal for a situation that could create CTCs however, are Kerr black holes as proposed by Etesi and Nemiti [6], which seems to be a quite physically realistic situation.

1.1.3 Quantum mechanics

General relativity does not describe all of our observations correctly, when systems are very small we observe quantum interference and other quantum effects. These are described by the theory of quantum mechanics. It is highly likely that a CTC, if it exists, will also exhibit these quantum effects. And in the physical model by Deutsch that I use in this thesis the states and the interactions in the CTC are described quantum mechanically.

I will not give a full introduction to quantum mechanics here, for a thorough introduction one can read any introductory physics textbook, for example Griffiths [7]. Or to the reader who wants a shorter, more philosophical introduction, I can recommend the Stanford Encyclopedia entry on quantum mechanics [8].

In quantum mechanics, the state of a system is described by a unit vector in a complex Hilbert space. A Hilbert space is a vector space that is complete in the norm defined by an inner product. The Hilbert space that the states are in depends on the system that we want to describe the state of. It may be a finite-dimensional or infinite-dimensional Hilbert space.

The observables we see in the classical world, like position and momentum, are not always defined completely by the quantum state. For example, a particle might be in a superposition of being in location a and b. If the quantum state of the particle at location a would be $|a\rangle$ and at location b $|b\rangle$ then it could be in a superposition like:

$$\frac{1}{\sqrt{2}}(|a\rangle + |b\rangle) \tag{1.1}$$

In an experiment we would then have a 50% chance of observing the particle in location a and a 50% chance of observing it at location b. How we should interpret the state of the particle before we observe it is a major subject of debate.

An important feature of quantum mechanics that I will use is that all evolution, except collapse in some interpretations of quantum mechanics, is unitary. This means that all evolutions can be described by a unitary operator. A unitary operator is linear, bounded and its adjoint reverses its effects.

1.1.4 Time travel

So in general relativity CTCs can show up that could allow for time travel to the past. But in these situations quantum mechanical effects could show up as well. The problem is that at this time, we do not know how to unify general relativity and quantum mechanics in one theory. This makes it impossible to know for sure how systems behave in which both theories are important. That makes it hard to reason about time travel and computation with time travel.

However, in 1991, Deutsch [9] published a model of how CTCs could be expected to work. He proposed that we should take the CTCs from solutions of general relativity, but describe the interactions and states of the systems quantum mechanically. This is a

1.2 Research questions

In this thesis, I generalise the work that was done by Aaronson et al. [2]. They showed what is computable on a computer that has access to a single CTC. But in this thesis, I investigate whether this is the limit of computation in CTCs. I consider a spacetime model with multiple CTCs nested within each other and investigate how this affects the computational power in those spacetimes. In general, the question I will try to answer is *"Given any $n \in \mathbb{N}$, what is the computational power of a universe with n nested CTCs?"*.

1.3 Relevance

The basis of my research in this thesis is very theoretical. It is unclear whether CTCs are physically possible at all and how exactly they would behave. And if they can exist, it is highly unlikely we will be able to create our own CTCs, let alone harness them for computation. But I think this research is still highly relevant, to computer science, physics, and philosophy.

To physics, because understanding the computational properties of different spacetime models and possible laws of nature will help us better understand how the universe would behave under these circumstances.

This computational exploration of nested CTCs gives a glimpse into how the nesting of these CTCs affects their behaviour. This research also raises the question if there are any physically realistic models in which nested CTCs could show up in general relativity. And people who are committed to a strong version of the Church-Turing thesis might take this research as another reason to consider CTCs to be fundamentally impossible.

To computer science, because it explores computational concepts under a different computational model, that might be more physically realistic than the standard Turing machine model. Furthermore, exploring algorithms in such an unorthodox setting might give us insights into new interesting techniques for algorithms and proofs in classical computational theory. It also connects the theory of Turing degrees to a potentially physically realisable model of computation in an interesting way. And lastly this research should be a welcome reminder that in the end computation is a physical process, so computer scientists should keep an eye on developments in our understanding of physics.

To philosophy, because it explores what kind of problems could be computable. If one subscribes to a materialist view, this has consequences for what can be known in general. In a materialist view, one might conclude based on a strong version of the Church-Turing thesis that problems like Hilbert's ten problems are fundamentally unsolvable. A constructivist might even believe that problems such as that might never have a truth value. But if CTCs are possible this research shows how we can stretch those limits and more problems can become in principle solvable.

2 Time travel and hypercomputation

If my calculations are correct, when this baby hits 88 miles per hour. . . you're gonna see some serious shit!

Dr. Emmett Brown, *Back To The Future (1985)*

In this chapter, I will discuss time travel and how it connects to hypercomputation. I will first discuss previous work on time travel and its logical and philosophical consequences. Then I will introduce the fields of supertasks and hypercomputation, and discuss how these are connected. Lastly, I will discuss different proposals for using time travel to enable supertasks and hypercomputation.

2.1 Time travel

Time travel is a popular science fiction trope and is often dismissed as an impossible fantasy. But a close look at our best fundamental physical theories shows that many forms of time travel might not be impossible. For example, forward time travel, in the sense that an observer could go far into the earth's future in a for them subjectively short time, is possible in the framework of special and general relativity.

The standard example for such forward time travel in special relativity is of a twin, **Alex** and **Brett**. **Alex** remains on earth, while **Brett** takes a trip on a spaceship travelling far away from earth and then back, as illustrated in Figure 2.1. If **Brett** travelled there and back with sufficient speed, they would have aged much less than **Alex**. This is because when an observer travels close to the speed of light, time dilation occurs, causing the time to pass differently for the twins.

This scenario is called the twin paradox because for many people it is initially surprising that Alex ages faster than Brett. It might seem that Brett will also see Alex as travelling very fast relative to them, so Alex must have aged less as well. But in fact, to make the round trip, Brett accelerates, so they are not in an inertial frame. This is what causes the asymmetry

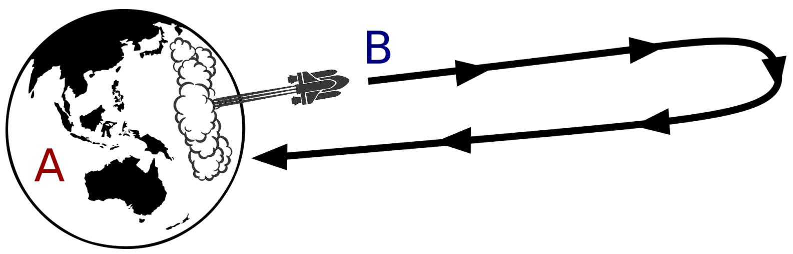


Figure 2.1: The situation in the twin paradox

between Alex and Brett, so the twin paradox is not a paradox in the usual sense [10].

But that is just time travel to the future, and is as discussed, non-controversially explainable in the framework of relativity. Time travel to the past however is very different. With time travel into the past, I mean a scenario in which an observer could keep going into their relative future, experiencing time as normal, but end up back in time. More specifically, they could end back up on the same spacelike hypersurface, in essence experiencing the same time twice.

In science fiction stories this is accomplished by fancy time machines, maybe a DeLorean or a phone booth. In this thesis, I will consider backwards time travel (which I will just call time travel from here on) more abstractly as travelling through a closed timelike curve (CTC). CTCs are loops in spacetime, parts of spacetime through which an observer could travel to their relative future, but end up to the past of some point in time where they started from. If such CTCs are present and an observer can follow this loop, then time travel is possible.

In special relativity, this is not possible, as normal Minkowski spacetime does not contain CTCs. However, in general relativity, matter can curve the spacetime. And already in 1949 Gödel [11] found the first solution of Einstein's general relativity in which CTCs were present.

Whether CTCs are actually physically possible is impossible to say right now. The possibility and stability of CTCs depends on fundamental facts of nature that are not solely described by general relativity but are affected by domains currently described by quantum mechanics. We do not know how to combine general relativity with quantum mechanics, and as they are now the theories are fundamentally incompatible. So to understand how the universe would actually behave in situations that might result in CTCs we need a theory of quantum gravity and this is one of the big open problems in physics. But since we do not have a consensus on quantum gravity yet, we cannot say whether it is physically possible for CTCs to form and whether such CTCs if formed would be stable under small perturbations.

2.1.1 The grandfather paradox

A separate problem that long has made time travel seem impossible is that paradoxes seem to pop up whenever backward time travel is introduced. Usually, we consider causation to only act forward in time, but if observers or information can be transported backward in time, causation seems to be affected as well. This seems to cause paradoxes when we try to imagine how time travel would work.

The most famous of these is the grandfather paradox. In this a time traveller attempts to prevent their own birth. They travel back in time to kill their grandfather, before their grandfather has the chance to conceive one of their parents. For the science fiction enthusiasts, this is a bit like the plot of Back to the Future, where Marty accidentally stops his parents from conceiving him. In the movie, the inherent paradox is ignored and he just slowly fades from family photographs. However, this kind of situation leads to real philosophical issues.

Imagine that the time traveller succeeds in killing their grandfather and thus preventing their conception. Then they could not have travelled back in time to kill their grandfather at all. Which means they would not have prevented their conception, so could have travelled back in time. So there seems to be a contradiction when the time traveller succeeds in killing their grandfather. But how is this possible? What stops the time traveller from succeeding? Is fate, or some other phenomenon stopping the time traveller? We know of no such mechanism.

And these contradictions do not only show up with murderous time travellers, simple objects can cause similar problems when time travelling. Anything that can affect the past could create such a problem. An example is Polchinski's paradox, in which a ball is sent through a CTC such that it knocks itself of the path of the CTC, as illustrated in Figure 2.2.

David Lewis [12, p. 150] points out that the time traveller in the grandfather paradox could fail for some commonplace reason, like just missing their shot or having a heart attack just before their final assault. When we think of the universe as a block universe—where the past, present and future are equally real—it is no more surprising that the time traveller cannot kill their grandfather than it is that I could not score a goal in the soccer match yesterday. For the time traveller because their grandfather was alive later in time and for me because we lost the game without scoring any goals. In both cases, the outcome of the action is determined by the future which is already fixed. The difference which makes the grandfather paradox seem less palatable is that there we usually do not have direct knowledge of the future. Usually, our knowledge about the future stems from our knowledge about the past and present, so it does not put any extra constraints on our expectations about events.

But how is the universe consistent then? It does not seem that the laws of nature that we know, both from general relativity and quantum mechanics,

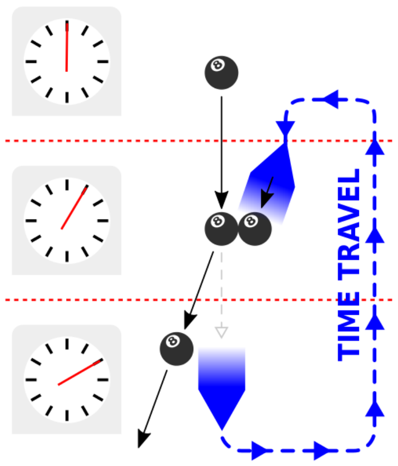


Figure 2.2: Illustration of Polchinski's paradox by BrightRoundCircle used under CC-BY-SA 4.0

are enough to force it to be consistent. It seems relatively easy to set up a situation where the universe could end up inconsistent when following the laws of nature. For example, imagine using a time machine to send back in time a bomb that destroys your lab before you can finish building the time machine. Do the initial conditions just always happen to be such that you are prevented from causing these contradictions for some mundane reason? This would seem to put non-physical constraints on the situations in which CTCs can be constructed, seemingly requiring the initial conditions of the universe to be just so, such that paradoxes cannot occur. And even if we accept this assumption, it might forbid almost all time machines, since (possibly small) inconsistencies would occur in almost any circumstance with time machines.

In 1991, David Deutsch published a paper [9] that gave a different solution to this problem. He proposed a model of CTCs that takes quantum mechanical effects into account. He took a curved spacetime with CTCs as follows from some solutions to general relativity and used quantum mechanics to describe the states and interactions of particles in such a spacetime. On this model, he then postulated a condition of *causal consistency*. This simply requires the quantum state of the CTC to be consistent. He proved that every quantum interaction between the degrees of freedom outside and inside the CTC has such a consistent solution, no matter what the initial conditions are outside the CTC. The following definition is adapted from Aaronson and Watrous.

Definition 7 (Causal consistency) A CTC is causally consistent if the evolution operator in the CTC maps the state of the initial hypersurface back to that same state. [13, p. 2]

The advantage of this consistency condition is that it reduces the amount of constraints on the past of the CTC. Of course, the conditions would still need to be such towards the past of the CTC that a CTC emerges, according to whatever laws of physics, be it general relativity or another theory that supersedes it. But apart from these constraints imposed by physical laws, there would not be any extra constraints needed to prevent paradoxes, since every possible initial quantum state in a given space-time configuration leads to a consistent solution. So if we someday find a procedure to create a CTC and Deutsch's proposal turns out to be correct, then we could use this procedure in any appropriate place without any extra rules being necessary to prevent paradoxes.

2.2 Hypercomputation

This thesis is about the consequences that time travel using Deutschian CTCs has on computation. Specifically, a computer that can use a CTC can compute more things than an ordinary computer. The field of hypercomputation studies all such computational models that are stronger than the Turing machines. In this section, I will introduce the field of hypercomputation and relate it to CTCs.

2.2.1 Church-Turing thesis

Nowadays, the computational model of the Turing machine is the most used model for reasoning about the decidability of computational problems. This model was introduced by Turing in 1936 in his paper On Computable Numbers, with an Application to the Entscheidungsproblem [4]. He introduced the Turing machine as a way of describing what kind of functions can be computed mechanically and used it to show that the Entscheidungsproblem is uncomputable. Not much later, it was shown that Turing machines and the λ-calculus can both compute exactly the same functions, which are the general recursive functions. This led to the thesis that a function can be effectively computed if and only if it can be computed on a Turing machine, or equivalently in the λ-calculus or if it is a general recursive function. This is called the Church-Turing thesis after Alonzo Church and Alan Turing.

This thesis is often used as a definition of computability and a large part of the field of computability is built around the model of Turing machines. But it also challenged scientists to formulate different models of computation that could compute things that Turing machines could not. The field of hypercomputation discusses these models. The ultimate goal would be a stronger model of computation that is still physically realisable. This would make it possible to create a computer that can solve problems that are fundamentally impossible to solve on ordinary computers.

2.2.2 Supertasks

One way of creating a stronger computational model is by giving the Turing machine an infinite amount of steps to execute. This concept borrows from the literature on supertasks. A supertask is a task that has an infinite amount of steps, but of which the result can still be had in finite time in some sense. A classic example of a supertask is the Zeno walk, where the runner needs to first run $\frac{1}{2}$ of the track, then $\frac{1}{4}$, then $\frac{1}{8}$, et ceteris to finish their run. The runner will of course finish, even though there is no final step. The apparent paradox is solved by considering the limit of the steps.

However, it gets trickier when a supertask does not have a limit. In the thought experiment of Thomson's lamp, we consider a lamp that is off. We switch it on after 1 minute. After $\frac{1}{2}$ minutes we turn it off. And then after $\frac{1}{4}$ minute we turn it back on. We keep accelerating this indefinitely. Now the question is, is the lamp on or off after 2 minutes? Neither it being on nor off seems possible because after each time it is off, there will be a time it is on and vice versa. Thomson [14] took this as proof that such a supertask is impossible. But Benacerraf [15] pointed out that Thomson's story only describes the state before the two-minute mark, so it is incomplete for questions about the state after two minutes. So when we use a supertask for computation we need to be careful that all the elements used for the computation are properly defined.

Tasks that speed up indefinitely are problematic to implement physically. It would usually mean that the speed of the elements of the system needs to grow unboundedly, but according to special relativity nothing can exceed the speed of light. However, the curvature of spacetime in general relativity leaves open the possibility for infinite amounts of time to be available before a certain point in spacetime. That would allow us to perform a supertask before a certain point in time, without needing to speed up the process indefinitely. In 1992, Hogarth [16] introduced the idea of using such spacetimes in general relativity to perform supertasks. We call such spacetimes *Malament-Hogarth* spacetimes, named after Mark Hogarth and David Malament, with whom Hogarth corresponded about the idea.

Definition 8 A spacetime is *Malament-Hogarth* iff there is a future-oriented endless timelike worldline λ with a start point p and there exists a point q, with a backwards light cone which contains λ. [17, p. 126]

In such a Malament-Hogarth spacetime (MH-spacetime from here on), one could for example solve the halting problem in finite time. One would start at point p and send a computer that runs the given program into λ, with instructions to send a signal to q if it ever halts. Now the observer can just travel to q in finite time and receive either a signal or no signal, which tells them if the computation halted. The problem that we encounter in the thought experiment of Thomson's lamp does not occur here, since our computation does not depend on some final state of the computer in λ. Hogarth [17] also showed that different MH-spacetimes can have different computational properties. By having nested MH-spacetime-like regions within the infinite part of an MH-spacetime even more problems can be computed.

MH-spacetimes are not as exotic as they may seem, our best understanding of rotating black holes suggests that these create an MH-spacetime. Rotating black holes are represented by a Kerr-Newman spacetime that has the MH property. Etesi and Nemeti [6, p. 354–8] show that an observer falling into the rotating black hole at an appropriate speed and angle can in finite time receive the result of an infinite computation of an orbiting computer.

The halting problem could for example be solved for the observer in finite time. The orbiting computer would run the program until it halts, or forever. If it halts, it sends a signal to the falling observer. If the computer ever halts, the observer receives a signal in finite time, so in finite time the observer will know whether the machine ever halts. The drawback for the observer is that they have to cross the inner horizon of the black hole, from which they never emerge. In a sense, the knowledge from this type of supertask is censored behind an event horizon by the universe.

2.2.3 CTCS

Any spacetime that contains a CTC is an MH-spacetime since we can fit an endless timeline λ in the CTC. But the supertasks proposed for MH-spacetimes by most authors, do not necessarily work for CTCs. Often a supertask is simply done in an MH-spacetime by doing it in λ where there is infinite time. However, although there is an infinite timeline through the CTC, there has to be consistency conditions on this CTC, because there is only one spacetime there. So in a CTC, we cannot simply run a computer in a CTC for infinite time. But CTCs are a very interesting case because they can be finite but still allow non-Turing behaviour.

If Deutsch's consistency condition on CTCs is assumed, Aaronson and Watrous [13] have shown the computational power of CTCs that are limited in storage space to an amount of bits that is polynomial in the input size. They have shown that in such a CTC a computer can exactly solve the problems in PSPACE in polynomial time compared to the input size. These problems can also be solved on normal Turing machines, but in exponential time compared to the input. In 2016, Aaronson et al. showed that there are also new problems that become computable with such CTCs. They showed how the halting problem, and all problems Turing reducible to it, can be decided with access to CTCs [2]. This means that there are strictly more problems that can be solved in a universe with CTCs than in a universe without.

However, Aaronson et al. also showed that in Deutsch's model of one CTC with causal consistency, no problem that is not Turing reducible to the halting problem can be decided. This is because the CTC can be simulated using a Turing machine with a halting oracle. But I will show that with multiple CTCs branching off from each other, even more becomes decidable.

2.2.4 Hypercomputation using time travel

Time travel could also be used to make hypercomputation possible. This could be done in three main ways. The computer can loop through the CTC forever, at a slightly different location in the CTC region each time, so that it can calculate forever in finite time seen from outside the CTC region. Or the time travel can be used to send the result of an arbitrarily long computation back in time. Or lastly, the computer can use information sent back in time in such a way that a consistency condition results in information being produced that cannot normally be (efficiently) computed. This last approach is how I approach hypercomputation in this thesis, but I will explain the other two approaches here.

Andreka et al. [18, p.4–8] give a description of the first approach, to send the computer through the CTC region infinitely often, in section 3 of their paper. This could allow the computer to compute for an arbitrarily long time. The halting problem could then be solved by letting the computer run until it finds that the machine halts. If the machine does halt, a signal is sent to the future of the CTC region. If it does not halt, no signal is ever sent. So at some fixed point in spacetime after the CTC region, an observer will either observe such a signal or not. If they do receive a signal they know the machine halts, if they do not, the machine does not halt.

The CTC region is the region of space through which many worldlines go that form closed timelike curves. If it is then possible to send the computer back in time, close to its older version, it can loop back in time again in a slightly different worldline. There would need to at least be infinite space in this CTC region for hypercomputation to be possible, because to solve these problems the computer needs to potentially be able to compute forever. This setup is illustrated in Figure 2.3 taken from their paper.

However, actually implementing hypercomputation in such a setting is complicated by the need to avoid the creation of black holes. As the hypercomputation is being done by the computer, it might need an arbitrary amount of storage space, which increases the mass of the computer. To make sure that the machine does not collapse into a black hole it needs to grow appropriately large. It also needs to be an appropriate distance from previous and future instances of itself. Because the computer can keep growing in size, the distance between its instances might keep growing and consequently the speed with which the machine moves through the space in the CTC might keep increasing. However, the speed of the computer is bounded by the speed of light, so it might not be possible to keep different versions of the computer far enough from each other. This makes it unclear if it is even possible to use this setup for hypercomputation.

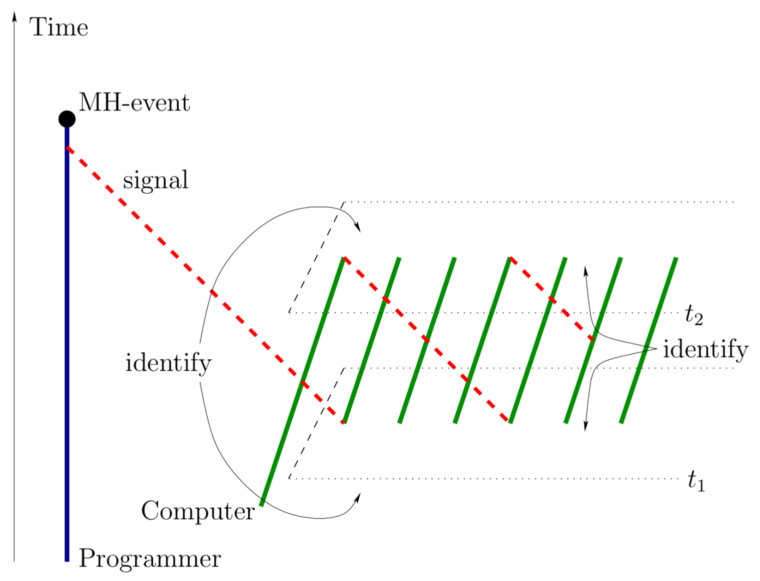


Figure 2.3: Illustration of hypercomputation by sending a computer in the CTC region by Andreka et al. [18]

This is why Andreka et al. propose to use the second way of hypercomputing using time travel: sending the result back in time. They propose that the computer calculates forever in the chronology respecting part of the spacetime, but uses the CTC to send back a signal if it finds a result. Here again, it is crucial that the result of an arbitrarily long computation can be received at a fixed time. For this the CTC does not need to go back arbitrarily far in time however, it is enough for there to be CTCs that go back a fixed amount of time departing from every time that the computer is working. The signal could then be sent through the CTC multiple times, repeated back through the CTC whenever it is received to send it further back in time. Until the signal eventually reaches back to the start of the computation [18, p. 8–11]. This setup is illustrated in Figure 2.4 taken from their paper.

They conjecture that such a CTC could be created by accelerating one mouth of a wormhole. This would have the same relativistic effect as in the twin paradox, such that the wormhole could be used to send a signal back in time. If this wormhole could then exist forever, it could be used for the hypercomputation Andreka et al. propose.

2.2.5 Fast computation using CTCs

The consistency conditions that CTCs impose seem to quite clearly make possible fast computation of hard problems by sending information back in time. For example, NP is a complexity class of problems for which a solution can be verified in polynomial time. Although it has not been proven, it is commonly thought that some problems in NP cannot be solved

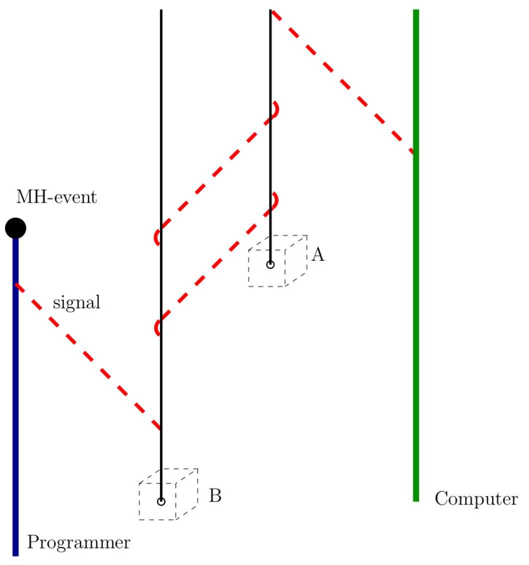


Figure 2.4: Illustration of hypercomputation by sending a result back in time through a CTC by Andreka et al. [18]

in polynomial time on a Turing machine.

However, if a Turing machine can send information back in time, it could take a possible solution that was sent back in time, verify it in polynomial time and cause a contradiction if it was not a solution. It is then concluded that the machine must immediately receive the solution for the universe to be consistent. With an algorithm like Algorithm 2.1 any problem in NP could then be solved in polynomial time on a Turing machine with a CTC.

Algorithm 2.1 Solving a problem in NP in polynomial time with a `sendBack` instruction to send something back in time, and `receive` to receive it.

```
1: possibleSolution ← receive()
2: if isSolution(possibleSolution) then
3:     sendBack(possibleSolution)
4: else
5:     sendBack(nextPossibleSolution(possibleSolution))
6: end if
7: return possibleSolution
```

This assumes that the CTCs would work classically, in the sense that they are not probabilistic, nor quantum. As discussed before, this is a problematic definition of CTCs, since it is hard to avoid paradoxes. For example, what would happen when we run Algorithm 2.1 on a problem that has no solution?

It can be quite complicated to correctly reason about computing using such consistency conditions. Even in established literature mistakes about the workings of such consistency conditions are made. For example, Brun [19] published a paper using this method to solve hard problems. He gives an algorithm for factoring large numbers, which I have adapted to my pseudocode style in Algorithm 2.2.

Algorithm 2.2 Brun's algorithm for factoring large numbers (that does not work correctly).

```
 1: $N \leftarrow \text{input}()$
 2: $p \leftarrow \text{receive}()$
 3: if $p \leqslant 1$ or $N \bmod p \neq 0$ then
 4:     $p \leftarrow 2$
 5:     while $N \bmod p \neq 0$ and $p \leqslant \sqrt{N}$ do
 6:         $p \leftarrow p + 1$
 7:     end while
 8:     if $p > \sqrt{N}$ then
 9:         $p \leftarrow N$
10:     end if
11: end if
12: sendBack(p)
13: return p
```

The idea of the algorithm is that if the received p is not a factor of N, a factor of N is searched for by lines 3–11. But this factor is then sent back. So no matter what happens, the algorithm always sends back a factor of N in time. So it must receive a factor of N as well. And since the p that is received is immediately a factor of N, no computational work has to be done in the loop and the algorithm runs in constant time.

But there is a problem here: nothing forces the algorithm to return the smallest factor of the given number. It can in fact always consistently return N itself. This is indeed always a factor of N, but that can also be accomplished in constant time on a normal computer by the statement `return N`.

The fallacy here is that Brun seems to think about the program as if the counterfactual calculations that enforce the consistent solution actually happen. If p did indeed start as a random number, but then be forced by the loop to be a factor, which is sent back in time after which the loop does not need to run, the algorithm would work correctly. But this is not the right way to think about CTCs, because only one thing happens in the CTC. The counterfactual situation that the inner loop is run does not actually ever happen. In fact, it does not matter what we put between lines 3 and 11, as long as the algorithm sends a different p back from the one it received.

This fallacy is important to keep in mind while reading this thesis, as consistency conditions are a powerful instrument, but need to be correctly applied. And this fallacy is so easy to make, that I found that when Ghosh et al. [20] tried to point out and correct this mistake in Brun's article, they end up making exactly the same mistake again. They greatly expand Brun's algorithm with extra flag registers supposed to keep track of whether the loop has been run, but still overlook the important fact that this counterfactual loop will in fact never run.*

For Deutschian CTCs however, Aaronson and Watrous [13] have thoroughly shown that both classical Turing machines and quantum Turing machines can exactly solve the problems in PSPACE using just a polynomial amount of bits. This includes all problems in NP and gives the interesting result that for complexity theory, quantum computers and classical computers are equivalent when they have access to a CTC. This also turns out to be the case for computability theory as Aaronson et al. [2] showed later.

*Algorithm 3 in the paper by Ghosh et al. can still always return N with an appropriate f flag. For example, f=1 p=4 is consistent for input N=4.

3 | Physical model

The Snake That Eats Its Own Tail, Forever and Ever. I know where I came from — but where did all you zombies come from?

"'—*All You Zombies*—'"
ROBERT A. HEINLEIN

Before I can discuss what the computational properties of nested CTCs are, I first need to explain what physical model I am considering. For this, I will first explain what physical model Deutsch [9] considers in his paper and what kind of different spacetime models it can represent. Then I will point out what kind of spacetimes are excluded by Deutsch's model. Lastly, I will explain the physical model that I will be using, and how it can represent these spacetimes that Deutsch does not consider.

3.1 Deutsch's model of computation in CTCs

CTCs show up in some solutions of the equations of general relativity, but Deutsch points out that these solutions usually have features that require the incorporation of quantum mechanics. How quantum mechanics and general relativity would interact in these situations would require a working theory of quantum gravity, which we as yet lack. But in the meantime, we can consider the quantum mechanical interactions of particles through causality-violating worldlines. So Deutsch chooses to analyse the CTCs using a simple quantum mechanical model.

In this model, the movement of the particles is approximated as classical movement through curved worldlines. This means that some particles can move through the CTCs, while others simply move normally in the spacetime, without looping. The internal degrees of freedom of the particles are treated quantum mechanically. For convenience, we can assume that every particle has one internal degree of freedom, for example spin-$\frac{1}{2}$. This way each particle can represent one qubit.

3.1.1 Interpreting quantum mechanics

Quantum mechanics describes a world that is very different from the classical world we are used to. Particles can be in superpositions of different positions or internal states, something we never experience in the macroscopic world. This discrepancy calls for explanation, which is provided by the different interpretations of quantum mechanics. Each of the different interpretations are empirically equivalent for the experiments we have been able to do so far, since otherwise the non-conforming interpretations would already have been disproven. However, the behaviour of the CTCs as proposed by David Deutsch would differ significantly in the different interpretations.

This is because Deutsch's model tries to ensure the consistency of the CTCs. The CTCs loop back to the same point in spacetime, allowing events in the future of a point to affect the state in that point. But for the model to be consistent, every point in spacetime can only have one state. In Deutsch's model, this state is a quantum state, but the different interpretations disagree on what exactly this quantum state is and how it evolves. To see how this works, we can look at a quantum version of the grandfather paradox, as Deutsch [9, p. 3202–3] does with paradox 1 in his paper.*

Take the scenario of two qubits interacting in a measurement gate that has the following action on the two qubits:

$$|x\rangle\,|y\rangle \Rightarrow |y\rangle\,|x \oplus y\rangle \tag{3.1}$$

Here $|x\rangle$ and $|y\rangle$ are the states of qubits, and $\oplus$ takes the two bits and computes its exclusive or: 0 if x and y are both 1 or 0 and 1 otherwise. The left side of the ket on the right side of the arrow then becomes the input for the $|y\rangle$ on the left side. See Figure 3.1 for an illustration.

Since $|x \oplus y\rangle$ becomes the input $|y\rangle$, this configuration imposes the consistency condition that $|x \oplus y\rangle = |y\rangle$. If $|x\rangle$ and $|y\rangle$ were classical bits, this would cause a problem when $|x\rangle$ is 1, since $(1 \oplus y) \neq y$. However, if we take $|x\rangle$ and $|y\rangle$ to represent qubits, this problem disappears and there are consistent values for any initial $|x\rangle$. For example, if $|x\rangle = |1\rangle$, a consistent value of $|y\rangle$ is $\frac{1}{\sqrt{2}}(|1\rangle + |0\rangle)$.

So far this could all work for the different interpretations of quantum mechanics, but it becomes problematic when real observables are added to the states.

For example, let us now consider the Grandfather paradox quantum mechanically. We examine the state at the time just after when the time traveller plans to kill their grandfather. Let us say the state of the world is

*The following explanation is based on Deutsch's explanation in his paper, but slightly altered to make it easier to understand for people with less background in quantum mechanics

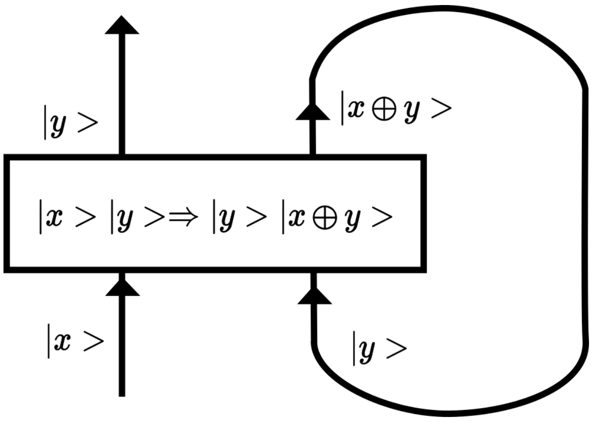


Figure 3.1: Illustration of the flow of qubits in Deutsch's paradox. The square box represents the measurement gate.

$|alive\rangle$ if the time traveller did not travel back in time and kill their grandfather and $|dead\rangle$ if they did travel back in time and kill their grandfather. The evolution around the CTC is then:

$$U(|\phi\rangle) = \begin{cases} |dead\rangle & \text{if } |\phi\rangle = |alive\rangle \\ |alive\rangle & \text{if } |\phi\rangle = |dead\rangle \end{cases} \tag{3.2}$$

Because if the time traveller travels back in time to kill their grandfather, they are never born and cannot travel back in time to kill their grandfather, so the state changes to $|alive\rangle$. On the other hand, if the time traveller does not travel back in time, they are born, so they can travel back in time to kill their grandfather, changing the state to $|dead\rangle$. Now the only possible state that is consistent will be one that remains the same after U. So we solve for:

$$U(|solution\rangle) = |solution\rangle \tag{3.3}$$

Which gives us (renormalised):

$$|solution\rangle = \frac{1}{\sqrt{2}}(|alive\rangle + |dead\rangle) \tag{3.4}$$

So the quantum state is in a superposition of the grandfather being killed by the time traveller and there being no time traveller to kill the grandfather. In the many worlds interpretation, observables can be multivalued, so this quantum state can remain in superposition. This means that in one world the grandfather is killed by the time traveller, corresponding to the part $|dead\rangle$ of the full quantum state. In this world, the time traveller that travelled back in time lives on in their grandfathers' time, but a young version of the time traveller is never born, since their grandfather was killed before their parents were conceived. In the other world, the grandfather is not killed by the time traveller, corresponding to the part $|alive\rangle$ of the full quantum state. In this world, the young time traveller is born to travel back in time later. But crucially, they do not travel back to the past of the same world, but to that different world of $|dead\rangle$. In this sense the time travel that CTCs allow to happen also allow observers to travel between the different worlds of the many worlds interpretation.

However, in all other interpretations, observables like whether the grandfather is killed must be single-valued. And this means that the grandfather needs to be either dead or alive, but cannot be both. So the consistency condition (3.3) cannot be satisfied. This holds for collapse theories—both dynamical collapse and collapse on measurement—, hidden variable theories and statistical interpretation. So in these interpretations, something, statistical flukes or other interventional mechanisms, need to prevent such a situation in which a time traveller kills their grandfather from happening.

And this is not only the case for this paradox, but for all paradoxes with CTCs. Deutsch [9, p. 43–44] shows that when the quantum state develops unitarily and the observables are multivalued, there is always at least one possible quantum state at every point that remains the same throughout a loop through the CTC for any initial condition, as long as the space in the CTC is finite. This means that in the many world interpretations CTCs can behave consistently with any initial conditions, but in the other interpretations this is impossible. These consistency problems would also rule out many other situations that could be paradoxical in these interpretations. So in this physical model, and in the rest of my thesis, we will assume the many worlds interpretation as a given, as Deutsch does as well.

3.1.2 Deutsch's consistency condition

To analyse what happens to the states when causality-violating regions like CTCs are involved, Deutsch stipulates the causal consistency condition, as stated in Definition 7. The consistency condition has to hold on every spacelike hypersurface, there can be only one quantum mechanical state at each point in time. For causality-respecting regions this is obvious and does not cause any surprising effects, all states are simply determined by their neighbouring states. However, in a CTC, a solution needs to be found that respects the dynamics of the region and gives a consistent solution when it loops back. This means that the state needs to be a fixed point solution of the evolution of the CTC and this causes the surprising computational behaviour of the CTCs.

3.1.3 Interacting particles

An important feature of Deutsch's model is that the particles under consideration are split into two groups. One group of particles that travels over the causality-respecting timeline. And another group that travels only through the CTC. So there are no particles travelling from the causality-respecting part to the CTC or vice versa.

However, the particles can interact through quantum gates that take the internal degrees of freedom of the incoming particles and transform their internal degrees of freedom appropriately. This way information can still be exchanged between the CTC and causality-respecting part of the world. Also within the CTC and within the causality-respecting timeline the particles in the same group may interact through quantum gates. In this way the quantum gates allow one to create any quantum computer in Deutsch's model.

3.1.4 Computational universality

This model of CTCs is restricted from a full consideration of CTCs in four main ways. Firstly, the spacetime structure of the CTCs is assumed. This is because there is no functional physical theory that combines quantum mechanics and general relativity. Such a theory is required to give a full explanation of the spacetime structure and the behaviour of a CTC. But I think this restriction is justified because this theory is simply not available. And the model is a reasonable guess of the behaviour of a CTC as relevant for a computation.

Secondly, only the internal degrees of freedom are being considered quantum mechanically. To be more general, all degrees of freedom of the particles should be considered quantum mechanically. However, for the computational properties, I think this restriction is justified as well, since this is also done in the standard gate-based models of quantum computation. Those are also supposed to capture the full computational properties of an ordinary quantum mechanical system, so the conclusion that the same can be done in when CTCs are present is reasonable.

Thirdly, as mentioned in the previous subsection, particles cannot travel between the CTC and causality-respecting timeline. In reality, this should be possible for a CTC, because the CTC is connected to the causality-respecting region in spacetime. However, Deutsch shows that such interchange is not computationally relevant by showing its denotational equivalence. I will explain this argumentation in the next subsection.

Lastly, the model only has a single CTC. Deutsch claims that his model is computationally universal in the sense that a computer in his model can simulate all other computers in all other spacetime models. If this is the case, then any computational properties of Deutsch's model are also true for all other spacetimes. I will show later in this thesis that this is not the case. When CTCs are nested within each other in the spacetime, computers can be constructed that cannot be simulated by computers in Deutsch's model. However, for any spacetime in which the CTCs are not nested his argument is reasonable.

3.1.5 Denotational equivalence

Deutsch attempts to show that his simple model can simulate all other spacetimes without nested CTCs in the following sense. Consider two systems that are both of bounded extension between two spacelike hypersurfaces. They are denotationally equivalent if they have the same function between the states of their starting spacelike hypersurface to their end, given a one-to-one correspondence between states in one system to the other system. We say that Deutsch's model can simulate a spacetime models if and only if for any system in such a spacetime model, one can construct an instantiation of Deutsch's model that is denotationally equivalent to that system.

To show this, Deutsch defines some denotationally trivial transformations of systems. These are transformations of the system into another denotationally equivalent system. For example, a system in which particles have more internal degrees of freedom can be transformed in a denotationally trivial way into a system with multiple particles with one internal degree of freedom representing each original particle. In the same way, positional properties can be represented in the system.

And Deutsch claims that all systems with multiple CTCs can be transformed into a denotationally trivial way to a system with just one CTC. For non nested CTCs this is indeed the case: we make a big single CTC from the start of the last CTC all the way back to the end of the first CTC. Then every particle that goes through the original CTCs can be rerouted through the main CTC, not interacting with any of the other particles during the times that they would not have travelled in the original system.

3.2 Nested CTCs

I propose a physical model that can represent more spacetimes than Deutsch's model. Instead of just one CTC, there can be any amount of CTCs in my model. Only one of the CTCs however is allowed to have a starting point directly in the causality-respecting region. The other CTCs all have a starting point from another CTC, but only one per CTC. This means that there is a straight nesting of CTCs, such that when n CTCs are in the model, they are also nested exactly n levels deep.

In Figure 3.2 I illustrate how the spacetime would be connected in my model. In Figure 3.2a I have sketched the spacetime connections of one CTC. The straight arrow going up represents the causality-respecting part of the spacetime. The illustration should be read with time going forward following the arrows. Then the loop back represents the CTC. The idea is

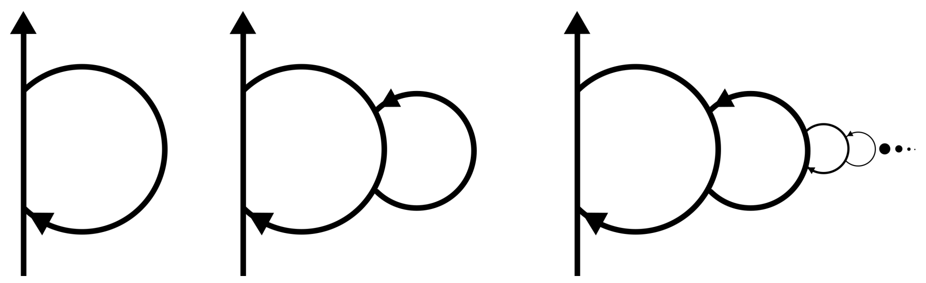

(a) One CTC (b) Two nested CTCs (c) Many nested CTCs

Figure 3.2: Schematic illustration of the spacetimes of my model

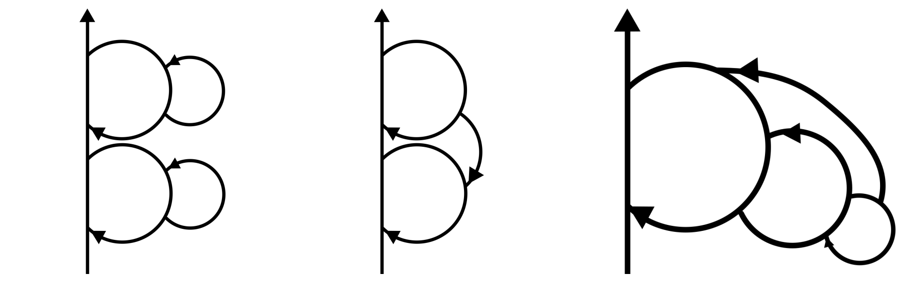

(a) Two stacks of CTCs (b) Bridge (c) Ladder

Figure 3.3: Spacetime models not allowed in my model

that a particle could either follow the straight line into the future, or could loop back by following the CTC.

In Figure 3.2b a nested CTC is added to the spacetime. This CTC has to depart from the original CTC, so that the nesting is exactly 2 levels deep. The second loop is drawn counter-clockwise, because it loops back compared to the local chronology in the first CTC. If it were drawn clockwise, it would only give another equivalent way for a particle to travel into the local future. In Figure 3.2c I show how this extends to adding an arbitrary amount of CTCs. Every extra CTC needs to be added departing from the last CTC.

Like in Deutsch's model, particles are still restricted to one CTC or to the causality-respecting timeline. So the particles cannot travel between the different CTCs. However, information can still be exchanged between particles in different CTCs using quantum gates at the intersection of different CTCs.

This requirement of only direct nesting of CTCs keeps my model simple. As I will show later when discussing the computational model arising from this physical model, the computational properties of a spacetime only depend on the deepest level of CTC nesting present. However, a lot of spacetime models are excluded. In Figure 3.3 I illustrate what kind of spacetimes are excluded. In the subsection on computational universality, I will discuss how these can be reduced to my simple model.

3.2.1 Consistency conditions

In this model of nested CTCs, Deutsch's consistency condition still has to hold everywhere. This means on every spacelike hypersurface there can only be one quantum state. The complexity of this condition does not change as we nest CTCs, but the computational consequences do. This is because for every CTC the universe must find a fixed point of a formula of the following form to fulfil the consistency condition:

$$E(|\rho\rangle) = |\rho\rangle \tag{3.5}$$

Here E is the evolution function of that CTC. However, for all but the deepest CTC, E is not simply a normal quantum mechanical evolution. The evolution of these CTCs depends on the behaviour of the CTC that departs from within it. And this nested CTC once again has a consistency condition and a formula for which it has to find a fixed point. So a consistent solution for a set of nested CTCs is not just one fixed point solution, but an entire chain of fixed point solutions from the deepest CTC to the outer CTC. And all these solutions have to match, such that the deeper solutions are part of the evolution function of the outer solutions.

It is this chaining of the consistency conditions that results in additional computational powers. It might seem like a problem to find such a solution, and of course on ordinary Turing machines it is, since this is a hypercomputational model. However, in the chapter on computational limits, I will show that it is possible to find these solutions with the appropriate (halting) oracle.

3.2.2 Retrospective constraints of nested CTCs

Under Deutsch's consistency condition, nested CTCs, like single CTCs, do not put any constraints on initial conditions as long as the space in the CTC is finite. There is always a solution for the quantum state in the CTCs that satisfies Deutsch's consistency condition. This means there is no way to create paradoxical situations where consistency is impossible like the grandfather paradox, in my model.

To show this, I show that Deutsch's proof for this lack of retrospective constraints for one CTC also applies to many nested CTC. Deutsch proofs that there is always a solution for $\hat{\rho}_2$ for any given initial state $\hat{\rho}_1$ and unitary operator U for the following formula.

$$\mathrm{Tr}_1[U(\hat{\rho}_1 \otimes \hat{\rho}_2)U^\dagger] = \hat{\rho}_2 \tag{3.6}$$

In the model of nested CTCs the state is still described by density operators, and the evolution is still unitary. Even if within a specific CTC there is another nested CTC, the behaviour within that CTC is still unitary. That means that Deutsch's argumentation still holds: there will always be a solution for $\hat{\rho}_2$ in instantiations of (3.6) for nested CTCs for any initial state. So also in the nested CTC model no retrospective constraints are imposed by the CTCs as long as all the CTCs have finite space.

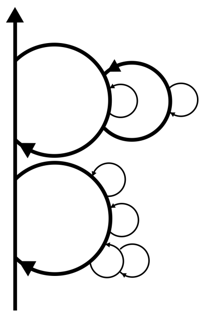
However, when the state can be infinitely large, unitary operations do not always have a consistent fixed point. If for example all states $|n\rangle$ for $n \in \mathbb{N}$ are valid and the mapping $|n\rangle \Rightarrow |n+1\rangle$ happens in the CTC, then there is no consistent solution for that CTC, no matter the initial conditions. This is why Aaronson et al. [2, p. 9] have as an assumption that all CTCs with infinite state space that can exist do in fact have a consistent solution. I assume the same for nested CTC and will make this more explicit in the next chapter.

3.2.3 Computational universality

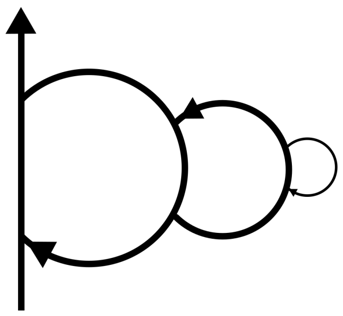
My model includes Deutsch's model of CTCs, so can at least simulate all systems that Deutsch's model can. But there are more denotationally trivial transformations that we can use to transform spacetime models that are excluded to our model. For example, two stacks of nested CTCs of depth n and m can be denotationally trivially transformed into a single stack of $max(n, m)$. The particles that travel through the smallest stack are redirected to the same depth on the larger stack but kept separated so that the two sets of particles do not interact. The time order that exists between the two stacks can be represented by letting the particles of the first stack affect the particles of the second stack in the region between the entrance and exit of the outer CTC of the final stack. For example, the two stacks of Figure 3.3a can be transformed into the singles stack of Figure 3.2b.

When it is not the causality-respecting region (the straight line), that has two stacks of nested CTCs, we can use the exact same transformation. It does not matter for the transformation whether the stacks branch off from a CTC or causality-respecting spacetime. And we can apply this transformation as many times as needed, such that we can transform any set of nested stacks of CTCs to a single stack of CTCs with the same depth as the deepest-nested CTC. For example, we can transform Figure 3.4a into Figure 3.4b in a denotationally trivial way.

The bridges (Figure 3.3b) and the ladders (Figure 3.3c) are more complicated. They cause the solutions of different stacks of nested CTCs to become dependent on each other. I think this can be transformed into my model, but I am not sure what depth of nesting is needed. It is not clear if we need to consider these spacetime models, as I do not know of any way this kind of spacetime models could satisfy Einstein's equations of general relativity. But more research is needed to answer these questions.



(a) A spacetime model that is not allowed



(b) Denotationally equivalent but allowed

Figure 3.4: How a complicated spacetime model can be represented in my model

4 | Computational model

Memory is a strange thing. It does not work like I thought it did. We are so bound by time. By its order... But now I'm not so sure I believe in beginnings and endings.

Dr. Louise Banks, *Arrival (2016)*

In this section, I will introduce a computational model for computation in nested CTCs. This model is based on Aaronson et al.'s [2, p. 9–10] model for computation in CTCs. I call this model TMNC for Turing Machine in Nested CTCs. I will first introduce Aaronson et al.'s model. Then I will define the TMNC and explain how it differs from their model to represent multiple nested CTCs.

4.1 Aaronson et al.'s $\mathrm{TM}_{\mathrm{CTC}}$ model

Aaronson et al. [2, p. 9–10] introduce the $\mathrm{TM}_{\mathrm{CTC}}$ model for a Turing machine with access to a CTC in section 3 of their paper. This model has two tapes: a causality respecting tape R_{CR} and a closed timelike tape R_{CTC}. Both tapes are unrestricted in size, but only a finite part of it can be written to in finite time, so the state of the $\mathrm{TM}_{\mathrm{CTC}}$ can be represented by the ordered tuple (x, y), where x is the content of R_{CR} and y is the content of R_{CTC}. The input is always present on the R_{CR} when the machine starts.

The $\mathrm{TM}_{\mathrm{CTC}}$ can move over these tapes and through its finite set of internal states like an ordinary Turing machine. But it is also probabilistic, in the sense that it can make random steps. It has a special instruction that sets a square on the tape to either 0 or 1 with $\frac{1}{2}$ probability for each option. The instructions of the machine can alter both tapes, and they require that this always results in the machine halting in finite steps with probability 1.

As a result, the $\mathrm{TM}_{\mathrm{CTC}}$ gives a probabilistic mapping over R_{CTC} if we ignore its effects on R_{CR}. They require that this mapping has at least one fixed point for every input on R_{CR}. This requirement is necessary because for an infinite CTC there might not always be a consistent fixed point, as discussed in the previous chapter. To uphold the consistency condition on the CTC, the $\mathrm{TM}_{\mathrm{CTC}}$ will always have such a fixed point on the R_{CTC} tape.

Furthermore, it is required that all the fixed points over R_{CTC} for a given input agree on either rejecting or accepting. A certain halting state of the TM_{CTC} is defined to be either rejecting or accepting if it rejects or accepts respectively with a chance of $\frac{2}{3}$ or higher.

4.2 The TMNC model

For modelling computation in nested CTC, I will now introduce the TMNC model. In many ways, it is similar to the TM_{CTC}, but it is expanded to model computation in nested CTCs. It is instantiated with a specific depth $d > 0$ that corresponds to the degree to which CTCs are nested in the spacetime it uses. So a TMNC of depth 1 models the same spacetime that Deutsch describes and that Aaronson et al. model in the TM_{CTC}. A TMNC of depth 2 uses a spacetime with two CTCs, one originating from the linear spacetime and one originating from within the first CTC. A TMNC of depth 3 will add another CTC originating from the second CTC and so forth.

The reason the TMNC has a parameter for the depth of nesting of the CTCs is that this influences what problems are computable. By specifying the depth as a parameter, I can be more specific about what problems can be solved in what spacetimes. Of course, the model can also be considered as a whole, for any depth parameter. Then it has the power of all possible models.

Like the TM_{CTC}, the TMNC has multiple tapes for multiple spacetime regions. In my physical model there are multiple CTCs, each with its own set of particles travelling through it. In the TMNC this is represented by each CTC and the causality-respecting worldline having its own set of tapes. These tapes represent the data contained in the internal degrees of freedom of the particles in that CTC or worldline.

4.3 TMNC state and actions

In our physical model from the previous chapter, there are particles travelling around worldlines in different CTCs. Every CTC and the causality-respecting region have a distinct set of particles that travel through it. Each of the particles have an internal state and can interact when they pass through the same gate.

The TM_{CTC} represents computation in one CTC and has two tapes for this, one representing data in the causality respecting region of the spacetime and one representing data in the CTC region. The TMNC however can have many nested CTCs and the data in each of these CTCs needs to be treated differently, which is why has a separate set of tapes for each CTC.

The TMNC does not just have one tape for each CTC, but a fixed amount of tapes for each CTC region and one set for the causality respecting region. This makes discussing some of the algorithms in this thesis easier, but does not change the power of the CTC and is only for convenience. It does not change the power, because all the information of a set of tapes can be represented on a single tape by interspersing the data. This does not change the computational power of our TMNC, just as it does not change the computational power of an ordinary Turing machine [21, p. 5–6].

Furthermore, just like the TM_{CTC}, the TMNC is a probabilistic machine. Probabilistic Turing machines are not any stronger than normal Turing machines, since there is a variant of a non-deterministic Turing machine, which also decides the same problems as a normal Turing machine [22, p. 248].

It is important to note that our TMNC needs to be instantiated with a specific depth to result in an actual computational model, this is because each depth of CTCs has a distinctly different computational behaviour. I will first describe the tapes and actions that a TMNC has access to, then I will introduce the consistency condition that results in the extra computational power.

A TMNC has a depth d, larger than 0, depending on what spacetime the TMNC it is representing computation in. The depth corresponds to the number to which the CTCs in the spacetime are nested. This parameter d affects the computational power, but the TMNC also has a second parameter k that does not affect the computational power. k is an integer larger than 0, that determines the amount of tapes that the TMNC has for each depth.

So a TMNC has $k \cdot (d + 1)$ tapes, k tapes for every CTC and another k tapes for the causality-respecting region. All the tapes share the same finite alphabet, usually $\{0, 1, \#\}$.

The state of a TMNC has one extra element, an integer c between 1 and d. This represents in which part of the spacetime the current computations are happening. It is important to keep track of this, because the changes of the c state will be used to determine the consistency conditions that have to hold over the tapes as I will explain later.

The value of c starts at 1 and can be incremented or decremented in a special machine step. However, once c has been decremented, it cannot be incremented anymore and the machine can only halt when $c = 1$. These restrictions are necessary to make sure that the TMNC can be implemented in a spacetime with only d nested CTCs.

When writing algorithms for the TMNC we refer to the different tapes using a number and a letter, for example 3b. The number refers to the depth of the CTC that the tape represents. The letter distinguishes different tapes at the same depth. The number we use to refer to the depth however, will depend on the current depth of the machine.

I have chosen to keep the number of the tape not static, but changing with the current c of the machine. This will make it much easier to formulate sub algorithms that can then be run with any current value of c. To be precise, the number is $\mathit{tapeDepth} - c + 1$. The letter is simply an alphabetical enumeration of the different tapes with the same depth. So if $k = 3$, then for a TMNC of depth 2 at current depth 1, we would have the tapes $\{0a, 0b, 0c, 1a, 1b, 1c, 2a, 2b, 2c\}$. Tape $0b$ is the second tape for depth 0 and $2a$ is the first tape of depth 2. In some situations, some tapes might have a negative number, but we usually do not need to refer to these tapes, although it is of course possible.

4.4 Consistency condition

The tapes of any depth higher than 0 represent data in a CTC. Incrementing c to that number represents the start of that CTC and decrementing to below that number represents exiting it. So, since the quantum state needs to be the same at the start and the end of the CTC according to Deutsch's causal consistency, the probability distribution of the tape determined by the quantum state at the start and the end needs to be the same as well. So we postulate a consistency condition on all the tapes corresponding to a depth higher than 0.

Definition 9 (Tape consistency) A tape is *tape consistent* iff the probability distribution over its contents when the current depth of the machine first is equal to the corresponding depth is the same as the probability distribution over its contents when the machine last has c set to the corresponding depth.

This consistency condition is similar to the consistency condition that Aaronson et al. postulate on the TM_{CTC}, but at different times in the computation for the different CTCs. This is what gives the TMNC its hypercomputational powers. The higher the depth of the TMNC, the more consistency conditions apply and the more computational power, as I will show. This consistency condition is satisfied by nature as a matter of necessity, since without it the corresponding spacetime would be inconsistent. So all tapes that correspond to data in a CTC region will always be tape-consistent.

What the probability distributions that satisfy these conditions are depends on the input that the TMNC gets on the tapes corresponding to the causality-respecting worldline. The tape consistency condition has to hold for each possible input.

If there are multiple possible probability distributions over the tape, we cannot assume that nature will pick one particular one, only that it will pick one. When designing algorithms for the TMNC we need to make sure that all the possible solutions give us the expected result.

4.5 Halting and deciding

Because the TMNC is a probabilistic model, halting and deciding need to defined slightly differently from a Turing machine. I define it the same way Aaronson et al. do for their $\mathrm{TM}_{\mathrm{CTC}}$.

Definition 10 (Halting) A TMNC halts iff for every $\epsilon > 0$ the TMNC will have halted with a probability $1 - \epsilon$ at some time t.

The accepting and rejecting definitions work with an error margin of $\frac{1}{3}$, as Aaronson et al. do as well for their model. Many algorithms will not need any error margin and can give the result without any errors.

Definition 11 (Accepting) A TMNC accepts a certain input iff for every possible situation satisfying *tape consistency* the chance of the machine accepting is larger than $\frac{2}{3}$.

Definition 12 (Rejecting) A TMNC rejects a certain input iff for every possible situation satisfying *tape consistency* the chance of the machine rejecting is larger than $\frac{2}{3}$.

Definition 13 (Deciding) A TMNC decides a language L iff it accepts every $x \in L$ and rejects every $x \notin L$.

4.6 Restrictions on the model

Aaronson et al. put three restrictions on their $\mathrm{TM}_{\mathrm{CTC}}$. That it always halts, that there is always a fixed point for the R_{CTC} and that the fixed points for a given input all agree on accepting and rejecting. The first restriction also applies directly to the TMNC.

The second restriction is expanded to cover all the tapes representing data in a CTC region. That is, a TMNC has to be defined such that every tape representing data in a CTC region has at least one tape-consistent solution for each input. The last restriction only needs to be rephrased for the TMNC: all different tape-consistent solutions of all the tapes need to agree on either rejecting or accepting for each input.

4.7 Connection to physical model

Of course this computational model is meant to represent computation in the physical model from the previous chapter. The relation between a TMNC and a world with nested CTCs should be like the relation between a Turing machine and the (classical) macroscopic world as we know it. Just like one can imagine a little machine writing on and shifting a long paper tape, you should also be able to imagine some form of computer that implements the TMNC. Different tapes of the TMNC would follow different depths of CTCs, corresponding to the depth of the tape. This physical travelling of the tape back in time allows enforcing the tape consistency from Deutsch's consistency condition. From this I think it is clear that the TMNC could be implemented in my physical model.

4.8 Infiniteness of tapes

As mentioned before, the TMNC has infinitely long tapes it can read or write on. However, for an actual computer to be able to save an infinite amount of data, like on the infinite tapes, it would need to have infinite mass. To not collapse into a black hole, this would also need infinite space. It is hard to imagine an actual computer that is infinitely large, this would require huge resources and infinite space to create. In the case of the TMNC, if it needs information from the entire infinite tape, the CTCs would also need to have infinite space to house the CTC tapes. This seems like a problematic requirement.

On the other hand, Turing machines have infinite tapes as well. It is common practice in computability theory to not limit the tape size. This way the computability of a problem does not depend on how big or complicated the problem instance is, but only on whether the problem is fundamentally solvable.

There is an important difference between Turing machines and TMNCs regarding this though. Turing machines can only affect a finite part of the infinite tape in finite time. This is because each step can only write on one cell and only finite steps can be executed in finite time. For TMNCs however, this is not the case.

A TMNC creates probability distributions over its tapes, and those distributions might have weight on changes on the entirety of the infinite tapes. Take for example a TMNC of depth 1, that in its loop walks the tape to the right until it reaches something that is not a '1' and then adds another '1' with 50% probability. This has a consistent probability distribution on the CTC tape, but that distribution extends all the way to infinity, even though the machine has run for a finite time.

If we run this machine in a single world however, it will give us back a finite tape with a certain amount of '1's on it. Probably the amount of '1's will be very small, as the probability distribution falls of exponentially for longer strings of '1'. So how should we interpret this? Is there a problem with infinity here or not?

In a many worlds interpretation of this situation each of the parts of the probability distribution is a world. In each of these worlds only a finite part of the tape is affected. So no infinitely large computer is needed. However, what is needed for the computer to behave correctly is for the machine to be able to grow arbitrarily large. There can be no restriction in space, otherwise the machine would not behave as we expect it to and the final probability distribution will be affected. And this change in probability distribution can also change the result of the TMNC.

In classical computability theory we have such problems as well. The problem of whether a number is a prime is only solvable for any number on a normal computer if that computer can be arbitrarily large. If there is a maximum size of the computer then it would not be able to find the solution for a really large number, of which the representation does not fit into the computer memory.

The situation in this case is slightly different though, because the machine might need to be able to grow infinitely large for a finite problem instance. I will show in Chapter 5 that to give a negative answer to a halting problem instance, we need a probability distribution over the entire infinite tapes in the CTCs. This means that unrestricted growth of the machine is needed for any of the problem instances, so the connection between the theory and the practical implementations might be lost. In our classical computability theory it is known that the Turing machine has an infinite tape, but that any finite problem instance of a solvable problem only needs finite space. In the TMNC this is no longer the case, even for simple problem instances, the machine needs to be able to grow unrestrictedly large.

But once again the expected size of the machine does not grow large at all. It remains finite and very small. It is just required that nothing restricts it from growing larger and larger in worlds with smaller and smaller magnitudes in the universal wave function. Furthermore, this is a general problem that any proposal for hypercomputation has. All proposals I know of for hypercomputation in general relativity require the unrestricted growth of the machine, simply because to solve these uncomputable problems infinite space is required. For example, Andreka et al. [18] propose to let a computer calculate infinitely into the future and send the result back in time. This requires that the machine can grow arbitrarily large in the future, they do not consider this a fundamental problem. It even requires that the machine survives forever, while the TMNC only uses finite time.

When considering the physical realisability of this computational model, this arbitrary growth is something to take into account. But the model does not make any more exotic assumptions than other hypercomputation models. In fact, because the machine always remains finitely sized and halts in finite time, it uses a lot less resources than proposals in which the Turing machine is simply made to calculate forever.

5 | Computational properties

And what if you could go back in time and take all those hours of pain and darkness and replace them with something better?

Gretchen, *Donny Darko (2001)*

In this chapter, I will first introduce the theory of Turing degrees to reason about computability. Then, I will discuss the computational properties of the TMNC model that was introduced in the previous chapter. Specifically, I will prove Theorem 1. To show this, I will show that an algorithm for solving multiple dth halting problems on a TMNC of depth d exists. Later in the chapter, I will use this algorithm to show that all problems that all problems with Turing degree $0^{(d)}$ are decidable by a TMNC of depth d. I will use the theory of Turing degrees I introduced in Section 1.1.

Theorem 1 (Computational power of TMNCs) A TMNC of depth d can decide all problems with Turing degree $0^{(d)}$.

5.1 Solving multiple dth halting problems

Definition 14 The problem of solving multiple dth halting problems is as follows:
Given: a list of n descriptions of oracle Turing machines with a $(d-1)$th halting problem oracle, where $n > 0$
Output: for each of these oracle Turing machines whether they would halt on an empty output.

In this section, I will proof Lemma 1. I will use a k of 3, so the TMNC will have three tapes for each depth.

Lemma 1 For any $d > 0$ there exists an algorithm on a TMNC of depth d that can solve multiple dth halting problems.

This algorithm differs for each of the possible depths a TMNC can have, because the problem differs for each possible depth. So to give an algorithm for each of the possible depths a TMNC can have, I will use induction. The base case is that an algorithm exists for a TMNC with depth $d = 1$ to solve multiple 1th halting problems, which are ordinary halting problems. In the induction hypothesis I proof that an algorithm exists for solving multiple $(d+1)$th halting problems on a TMNC of depth $(d+1)$, given an algorithm for a TMNC of depth d which solves multiple dth halting problems.

5.1.1 The base case

I will now proof the base case of Lemma 1, as given in Proposition 1. To do this, I will give such an algorithm here. The algorithm is based largely on the algorithm given by Aaronson et al. [2, p. 12–14] for solving problems reducible to the halting problem in one CTC, but adapted to the TMNC model I have defined. This is because this claim is the same as the result of Aaronson et al., just for a more general model.

Proposition 1 (Base case) There exists an algorithm for a TMNC of depth 1 that can solve multiple 1th halting problems.

Let us assume that the n descriptions of the Turing machines we are to determine the halting for are given as a list on tape $0b$. For each of the Turing machines, we can simulate their machine steps on our TMNC using the behaviour that it shares with a Turing machine. As such, given a transcript of a Turing machine, we can determine if it correctly describes how the Turing machine would run by simulating the steps. We can also check if the Turing machine halts at the end of the transcript, again by simulating.

We use this fact and a transcript on tape $1a$, which has a consistency condition, to create a probability distribution over transcripts on tape $1a$ that is distinctly different for each Turing machine that halts, than for those that do not halt. The algorithm is given in pseudocode in Algorithm 5.1.

This algorithm works because it forces each part of tape $1a$ into a different fixed point whether each Turing machine given halts or not. And the parts of tape $0b$ are then correspondingly forced to have the correct result. For each machine, if it does not halt, the fixed point of its part of tape $1a$ is an infinite distribution of valid execution histories of the machine of length l. But each history with length l has a probability of 2^{-l}, so the length of the history on part j of tape $1a$ is almost always very short. The result on part j of tape $0b$ however will then always be "LOOP". If the machine does halt however, then the corresponding fixed point is simply the halting history with probability 1 and on tape $0b$ it will always record "HALT".

Since Algorithm 5.1 solves multiple 1th halting problems on a TMNC of depth 1, we have proven Proposition 1. ■

Algorithm 5.1 Solving multiple halting problems with depth 1

```
Require: There are $n$ descriptions of Turing machines on tape $0b$
 1: for each Turing machine description $T$ on part $j$ of tape $0b$ do
 2:     $t \leftarrow$ transcript of $T$ on part $j$ of tape $1a$
 3:     if $t$ is a valid transcript of $T$ then
 4:         if $t$ is a halting transcript then
 5:             write “HALT” to part $j$ of tape $1c$
 6:         else
 7:             if $\mathrm{random}() < 0.5$ then
 8:                 write only the first step of $t$ to part $j$ of tape $1a$
 9:             else
10:                 add one more step of $T$ to $t$
11:                 write $t$ to part $j$ of tape $1a$
12:             end if
13:             write “LOOP” to part $j$ of tape $0c$
14:         end if
15:     else
16:         write the first step of $T$ to part $j$ of tape $1a$
17:         write “LOOP” to part $j$ of tape $0c$
18:     end if
19: end for
Ensure: Tape $0c$ has for each of the $n$ descriptions whether they halt on
    empty input
```

5.1.2 The induction case

Proposition 2 (Induction case) There exists an algorithm for a TMNC of depth $(d + 1)$ that can solve multiple $(d + 1)$th halting problems.

Definition 15 (Induction hypothesis) There exists an algorithm for a TMNC of depth d that can solve multiple dth halting problems.

For the induction step, we now want to prove that Proposition 2 holds given induction hypothesis (Definition 15). The solution is given in Algorithm 5.2.

Just like in the base case, we try to simulate the oracle Turing machines from our input. For the normal machine steps, we can simply use our TMNC as before. However, when the machines consult their dth halting oracles, it becomes more complicated, since our TMNC does not have access to such an oracle. However, our induction hypothesis tells us that there is an algorithm that solves multiple dth halting problems on a TMNC of one lower depth. So, we simply save up these dth halting problems and solve them all collectively.

Apart from that, the algorithm works similarly to Algorithm 5.1. It again induces a probability distribution on tape $1a$ that gives short non-halting histories for non-halting oracle Turing machines and halting histories for halting oracle Turing machines. The algorithm is formulated in pseudocode in Algorithm 5.2.

Algorithm 5.2 Solving multiple $(d+1)$th halting problems with TMNC of depth $d+1$

```
Require: There are $n$ descriptions of Turing machines with a $d$th halting
    oracle on tape 0b
 1: for each machine description $T$ on part $j$ of tape 0b do
 2:   loop
 3:     $t \leftarrow$ transcript of $T$ on part $j$ of tape 1a
 4:     {For $\mathit{validTranscript}$, see Algorithm 5.3}
 5:     if $\mathit{validTranscript}(t, T, j)$ then
 6:       if $t$ is a halting transcript then
 7:         write “HALT” to part $j$ of tape 0c
 8:       else
 9:         if $\mathit{random}() < 0.5$ then
10:           write only the first step of $t$ to part $j$ of tape 1a
11:         else
12:           add one more step of $T$ to $t$
13:           write $t$ to part $j$ of tape 1a
14:         end if
15:         write “LOOP” to part $j$ of tape 0c
16:       end if
17:     else
18:       write the first step of $T$ to part $j$ of tape 1a
19:       write “LOOP” to part $j$ of tape 0c
20:     end if
21:   end loop
22: end for
23: increment the current depth
24: run the algorithm from the induction hypothesis
25: decrement the current depth
Ensure: Tape 0c has for each of the $n$ descriptions whether they halt on
    empty input
```

The $\mathit{validTranscript}$ function defined in Algorithm 5.3 determines whether the transcript of the Turing machine on the tape is valid. For this, it needs to simulate the oracle calls as well. It does this by putting the requests on tape 1b and assume that tape 1c contains the results. Running the algorithm from the induction hypothesis makes sure that the correct results are on tape 1c. The consistency condition on tape 1c makes sure that this result is already present when $\mathit{validTranscript}$ is run.

Algorithm 5.3 validTranscript function

```
Require: A transcript $t$ for an oracle Turing machine $T$, numbered $j$
1:  $state \leftarrow$ initial state
2:  for each step in $t$ do
3:      $s, state \leftarrow$ simulate next step from $state$
4:      if $s$ is oracle query then
5:          put the query on part $j$ of tape 1b
6:          assume the result of the query is what is on part $j$ of tape 1c
7:      end if
8:      if $s \neq$ step then
9:          return false
10:     end if
11: end for
12: return true
```

5.1.3 Induction result

By giving a base case and the induction step, we have shown in this section that on every TMNC of depth n, where n is a natural number larger or equal to 1, there exists an algorithm that given multiple oracle Turing machines with a $(n-1)$th halting oracle, it can correctly decide for each of these machines whether they halt. So we have proven Proposition 2. ■

Given the proof of the base case and the induction case, we can conclude that Lemma 1 holds. □

5.2 Solving the halting problems

Now we can show that any TMNC of depth d can solve the dth halting problem. To do this, we simply call the corresponding algorithm given in the previous section with a singleton list of just the oracle machine we want to determine the halting of. The result is then extracted from the singleton list that is in the output.

5.3 Solving problems with Turing degree $0^{(d)}$ or lower

But we can extend our result beyond just solving the halting problem. A TMNC of depth d can solve any problem that is Turing reducible to a dth halting problem. That means, that we can solve all the problems in $0^{(d)}$ and in any Turing degree lower than $0^{(d)}$. To show this, we again make use of the algorithm for solving multiple dth halting problems. The algorithm for solving a problem P given an oracle Turing machine that solves P using a dth halting oracle is given in Algorithm 5.4.

Algorithm 5.4 Solving problem P, where P is Turing reducible to dth halting on a TMNC of depth d

```
 1: $j \leftarrow 0$
 2: $\mathtt{state} \leftarrow$ initial state of T
 3: while $\mathtt{state}$ is not halting do
 4:     $\mathtt{step} \leftarrow$ next step of T from $\mathtt{state}$
 5:     if $\mathtt{step}$ consults the oracle then
 6:         write the queried machine to part j of tape 0b.
 7:         $\mathtt{state} \leftarrow \mathtt{step}$ applied to $\mathtt{state}$ with oracle returning what is on
            part j of tape 0c
 8:         $j \leftarrow j + 1$
 9:     else
10:         $\mathtt{state} \leftarrow$ next state after simulating $\mathtt{step}$
11:     end if
12: end while
13: run solving multiple halting problems {Algorithm 5.1 and Algorithm 5.2}
14: return $\mathtt{isAccepting}(\mathtt{state})$
```

Because our TMNC model of depth d is realisable in a space-time with d CTCs nested within each other, this result implies that in such a space-time we can solve any problem with a Turing degree of $0^{(d)}$ or lower.

6 | Computational limits

Do you now understand why I say the future and the past are the same? We cannot change either, but we can know both more fully.

The Merchant and the Alchemist's Gate
TED CHIANG

In the previous chapter, I have shown what is computable in spacetimes with different nestings of CTCs. I have given an algorithm that can solve problems Turing reducible to different levels of halting problems, depending on the depth of the nesting of the CTCs. But I have not shown that this is all that can be computed in such spacetimes. In principle, there might be an algorithm that can solve even more problems that can be run in these spacetimes. In this chapter, I will try to show that this is in fact all that can be computed in these spacetimes.

I will prove this based on the proof that Aaronson et al. [2, p. 16–21] give for the computational limits of a single CTC. I use this as the base case of my proof. However, when working on my proof, I realised that there is a problem with their proof. I will explain this problem here first, but then assume that it can be solved for the rest of my proof. So my proof only holds if Aaronson et al.'s proof is fixed in a satisfactory way.

6.1 Problem in Aaronson et al.'s proof

First I will give a short introduction to how the proof in section 5 of their paper works. They want to show that in a spacetime with one CTC nothing more can be computed than the problems that are Turing reducible to the halting problem. So they attempt to simulate a computer with access to a CTC with just an oracle Turing machine with access to a halting oracle.

If this is indeed possible, then all problems that can be solved in a CTC are Turing reducible to the halting problem. The Turing reduction would simply be to run the simulation of the CTC on the oracle machine and solve the problem that way. So if an algorithm to simulate a computer in a CTC on an oracle machine with just a halting oracle exists, then nothing more indeed can be computed in these spacetimes.

The tricky part of simulating the CTC is finding an approximation of the fixed point of the evolution of the dynamics in the CTC. We know that a fixed point exists, and that to fulfil Deutsch's consistency condition the CTC must be in that fixed point state. To successfully predict if a computer in a CTC accepts or rejects, we need to approximate the real fixed point closely in terms of trace distance. The trace distance is the largest measurable difference between two quantum states, so if we are close enough in this sense, the measured result will be close enough as well.

To find this approximation, they dovetail over all possible quantum states in finite-dimensional truncations of the real Hilbert space. For each of these states, they then check using the halting oracle if it is an almost fixed point of the quantum evolution in the CTC.

Definition 16 (Almost fixed point) A value σ is an almost fixed point of the function \$ with an error of ϵ iff for all $k > 0$ it holds that $|\$^k(\sigma) - \sigma| < \epsilon$. So repeated application of the function never takes the value further than ϵ away from the original value.

They prove that if a quantum state is an almost fixed point of error ϵ in terms of Euclidean distance, then it is also within ϵ from the real fixed point ρ in terms of Euclidean distance. But of course, they want to be close in terms of trace distance, so they need to convert the Euclidean distance to trace distance. They do this in Proposition 9 using the "Almost As Good As New Lemma". Here however, the trace distance turns out to be up to a factor $2^{k/4}$ larger than the Euclidean distance, where k is the amount of dimensions of the Hilbert space of the approximation, which will be problematic for the proof.

To compensate for this difference between the Euclidean and trace distance, in the proof they are forced to search for a quantum state that is an almost fixed point with an error that decreases exponentially with k. But for the algorithm to work, such an almost fixed point needs to exist, which means that there also needs to be a quantum state on a finite-dimensional truncation of the Hilbert space that is as close as that error to the real fixed point. So they assume that we can always find a k and $\sigma \in M_k$ such that:

$$\|\sigma - \rho\|_{tr} \leqslant \frac{1}{2^{k+12}} \tag{6.1}$$

However, this is not true. It is given that a fixed point ρ exists, but it exists in the infinite-dimensional Hilbert space M with basis $\{0,1\}*$. The σ's we consider are all in a Hilbert space M_k with basis $\{0,1\}^k$. And ρ might have its support divided over the different vectors in its basis, such that no σ is close enough in trace distance, if the support on the vectors representing tapes of length decreases exponentially based on the length it represents. This way it might be further than $\frac{1}{2^{k+12}}$ away, no matter how big we make k.

An example of a machine in a CTC where ρ is like this is the following. It enters the CTC and then it checks if at the start of the tape is a 1. If there is it moves 5 spaces to the right and repeats, until it finds an unwritten part of the tape. Then with $\frac{1}{2}$ probability it writes a 1. The fixed point of this machine is a probability distribution with the chance of a written part of the tape of length $5n$ to be $\frac{1}{2^n}$. So no approximation only considering tapes of length up to k will ever be closer than $\frac{1}{2^{k+12}}$.

I have tried to fix this problem, but it is not easy to solve since for vectors in infinite dimensional Hilbert spaces the bound between the trace and Euclidean distances has an exponential factor. So to get rid of this exponential factor, the proof needs to stay away from Euclidean distances. It is then however not clear how to prove that any almost fixed point is close to a real fixed point.

I have discussed this problem with the authors, but we have not found a solution yet. Scott Aaronson's PhD student Sabee Grewal has attempted to fix the proof, but not succeeded yet. Just recently in the last few weeks of my thesis, Scott Aaronson has proposed a different way of proofing the limits of computation of CTCs for classical computers. This proof does not require switching between L_1 and L_2 norms and the hope is that this can be adapted to the quantum case with appeals only to the trace distance.

Because this new proof idea comes late in my thesis planning, I sadly do not have the time to develop this idea into a new version of my proof. But I am hopeful that this will indeed turn out to give us a proof for the original statement in Aaronson et al.'s paper. If so, my extension of the proof here should work to extend that fixed proof, so that it is clear that my result is valid as well.

So I will now proceed as if the original proof of Aaronson et al. works, in the hope that it can be in fact be fixed.

6.2 Proof for nested CTCs

I want to prove that a computer in a set of n nested CTCs can exactly decide the problems that are decidable on an oracle machine with a halting-n oracle. I have already shown that such a computer can at least decide the problems decidable on an oracle machine with a halting-n oracle, so in this part I just want to show that such a computer can solve no more problems than those that can be solved using an oracle machine with a halting-n oracle. I will do this, by showing that an oracle machine with a halting-n oracle can always simulate a set of n nestedCTCs.

Definition 17 (Halting-1 oracle) An oracle for the normal halting problem.

Definition 18 (Halting-n oracle) An oracle for the halting problem of machines with a halting-$(n-1)$ oracle.

Because I want my argument to hold for any number of nestedCTCs, I will use induction. The induction hypothesis will be that:

Definition 19 (Induction hypothesis) The fixed point state of $n-1$ nestedCTCs can always be approximated to any required precision $\epsilon > 0$ in finite steps on an oracle machine with an halting-$(n-1)$ oracle. Such that:

$$\|\sigma - \rho\|_{tr} < \epsilon \tag{6.2}$$

Where σ is the approximation returned by the oracle machine and ρ is a true fixed point.

6.3 Base case

The base case is very similar to what Aaronson et al. [2] prove in section 5, but a bit more generalised. What we want to prove is the following:

Theorem 2 (Base case) The fixed point of a singleCTC can be approximated to any required precision $\epsilon > 0$ in trace distance in finite steps on an oracle machine with a halting-1 oracle.

The difference from Aaronson et al.ś proof is that we want to be able to approximate to any given accuracy, not just a fixed accuracy that is enough for answering yes-no questions. Whether this follows directly depends on how the proof by Aaronson et al. is fixed, but if the new proof direction that Aaronson suggested works, this will be true.

When we simulate a CTC, we are given a superoperator \$ that implements the unitary evolution of the CTC. This superoperator can be simulated on a normal Turing machine, because it does not contain any nested CTCs. This can be done by dovetailing over closer and closer approximations to all possible mixed states. For this, I will use the M_k definition of Aaronson et al. [2, p. 20].

$\{0,1\}^*$ is taken as the basis of the Hilbert space where the real fixed point ρ is in. Then one can approximate that with truncated Hilbert spaces M_k for $k > 0$, which are k-dimensional truncations with $\{0,1\}^k$ as basis. I use the following algorithm, based on the algorithm from Aaronson et al.:

Now to prove our base case, we have to prove two things:

1. That Algorithm 6.1 always finds such a σ and halts.
2. That any such σ that Algorithm 6.1 returns conforms to (6.2).

For the first point, that the algorithm always halts, we can use the same proof as Aaronson et al. (assuming it can be fixed). □

To fix Aaronson et al.'s proof, they will need to get rid of the factor k in the conversion from the error of the almost fixed point to how far away it

Algorithm 6.1 Finding an approximation of the fixed point of one CTC

```
for each $k > 0$ and $\sigma \in M_k$ do
    Use the halting-1 oracle to check whether there exists a $t > 0$ such
    that:
        $\|\sigma - \$^t(\sigma)\|_{tr} > \epsilon$        (6.3)
    if no such $t$ exists then
        return $\sigma$
    end if
end for
```

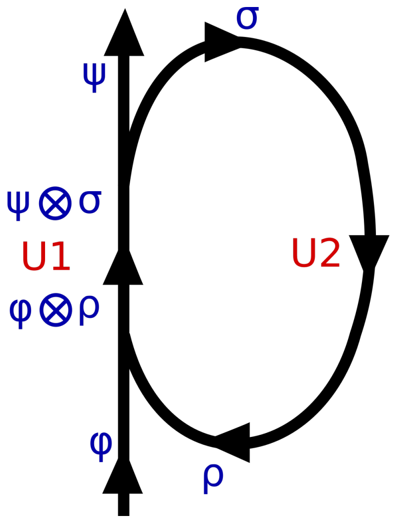


Figure 6.1: A schematic illustration of the quantum interactions of a CTC

is from the fixed point. If this is done, that proof can also be used directly here to show that this slightly altered algorithm gives the right approximate fixed point.

6.4 Induction step

Now it is time to go beyond what Aaronson et al. have done and see what happens when a nested CTC is added.

Again we want to find the fixed point of a CTC. But this time, our CTC has nested CTCs in it. If we only look at the deepest CTC and consider the CTC it is nested in as locally straight spacetime, our system looks like Figure 6.1. In this figure, U_1 and U_2 are superoperators representing some unitary evolution. All the labels in blue represent some quantum state at that point. From this figure it is clear that the nested CTC puts the following constraint on the system:

$$U_2(P_R(U_1(\phi \otimes \rho))) = \rho \tag{6.4}$$

Where P_R discards the left side of the tensor product. We are guaranteed that for any initial state ϕ of our system, this has a solution for ρ. We will call this solution ρ_ϕ. The true superoperator that describes the unitary evolution of our system and maps ϕ to ψ is then:

$$\$(\phi) = P_L(U_1(\phi \otimes \rho_\phi)) \tag{6.5}$$

If we could run this function on our oracle machine with a Halting-$(n-1)$ oracle, we could just use the same algorithm and we would be done. However, the induction hypothesis does not give us ρ_ϕ, instead it promises us a σ_ϕ that is arbitrarily close to ρ_ϕ. However, we can simulate U_1 to any level of precision, and superoperators cannot increase trace distance. So with σ_ϕ we can create an approximation function A(state, max error) such that for any ϕ:

$$\|A(\phi, c) - \$(\phi)\|_{tr} \leqslant c \tag{6.6}$$

And this A can be decided on the oracle machine with a Halting-$(n-1)$ oracle, because the σ_ϕ is given by the induction hypothesis and U_1 is a normal unitary evolution that can be simulated on any Turing machine. So on our oracle machine with a Halting-n oracle, we can use the following algorithm for any required accuracy $c > 0$:

Algorithm 6.2 Induction step

```
for each $k > 0$ and $\sigma \in M_k$ do
    Use the halting-$n$ oracle to check whether there exists a $t > 0$ such that:
        $$\|\sigma - A^t(\sigma, \frac{c}{3t})\|_{tr} > c \tag{6.7}$$
    if no such $t$ exists then
        return $\sigma$
    end if
end for
```

Now to prove our induction step, we have to prove two things:

1. That Algorithm 6.2 always find such a σ and halts.
2. That any such σ that Algorithm 6.2 returns conforms to (6.2).

6.4.1 Algorithm always halts

We are promised that \$ always has a fixed point ρ. So we can always choose a high enough $k > 0$, such that there exists a $\sigma \in M_k$ for which it holds that:

$$\|\sigma - \rho\|_{tr} \leqslant \frac{c}{3} \tag{6.8}$$

From triangle inequalities it follows that for any $t > 0$:

$$\|A^t(\rho, \frac{c}{3t}) - \rho\|_{tr} \leqslant \frac{c}{3} \tag{6.9}$$

Now it is clear that:

$$\begin{aligned}
&\|A^t(\sigma, \frac{c}{3t}) - \rho\|_{tr} \\
&\leqslant \|A^t(\sigma, \frac{c}{3t}) - A^t(\rho, \frac{c}{3t})\|_{tr} + \|A^t(\rho, \frac{c}{3t}) - \rho\|_{tr} && (6.10)\\
&\leqslant \|\sigma - \rho\|_{tr} + \frac{c}{3} && (6.11)\\
&\leqslant \frac{2c}{3} && (6.12)
\end{aligned}$$

Here (6.10) holds because of the triangle inequality. (6.11) holds because superoperators cannot increase trace distance and because of (6.9).

Using another triangle inequality, we can see that for all $t > 0$:

$$\begin{aligned}
&\|\sigma - A^t(\sigma, \frac{c}{3t})\|_{tr} \\
&\leqslant \|\sigma - \rho\|_{tr} + \|\rho - A^t(\sigma, \frac{c}{3t})\|_{tr} \\
&\leqslant \frac{c}{3} + \frac{2c}{3} = c && (6.13)
\end{aligned}$$

So clearly, Algorithm 6.2 will halt when it encounters this σ, if not sooner.
□

6.4.2 Algorithm is correct

Now we need to show that if for some state σ it holds for all $t > 0$ that $\|\sigma - A^t(\sigma, \frac{c}{3t})\|_{tr} \leqslant c$, it also holds for all $t > 0$ and some constant $d > 0$ that:

$$\|\$^t(\sigma) - \sigma\|_{tr} \leqslant d \cdot c \tag{6.14}$$

So that σ is an almost fixed point of \$. By the same reasoning as in the base case, it would then imply that $\|\sigma - \rho\|_{tr} \leqslant \epsilon$ for any arbitrarily low $\epsilon > 0$ by decreasing c.

First I will proof the following proposition about A.

Proposition 3 For any $t > 0$, $m > 0$ it holds that:

$$\|A^t(\sigma, m) - \$^t(\sigma)\|_{tr} \leqslant m \cdot t \tag{6.15}$$

Proof. I will proceed by induction over t. If $t = 1$, then by definition:

$$\|A(\sigma, m) - \$(\sigma)\|_{tr} \leqslant m \tag{6.16}$$

If $t = n$, then:

$$\begin{aligned}
&\|A(A^{(n-1)}(\sigma, \frac{m}{n}), m) - \$(\$^{(n-1)}(\sigma))\|_{tr} \\
\leqslant\ &\|A(A^{(n-1)}(\sigma, m), m) - \$(A^{(n-1)}(\sigma, m))\|_{tr} \\
&+\|\$(A^{(n-1)}(\sigma, m)) - \$(\$^{(n-1)}(\sigma))\|_{tr} \qquad (6.17)\\
\leqslant\ &m + m \cdot (n-1) = m \cdot n \qquad (6.18)
\end{aligned}$$

Where (6.17) follows from the triangle inequality and (6.18) follows from the induction hypothesis. ■

Now we can proceed with the correctness proof with a triangle inequality:

$$\begin{aligned}
&\|\$^t(\sigma) - \sigma\|_{tr} \\
\leqslant\ &\|\$^t(\sigma) - A^t(\sigma, \frac{c}{3t})\|_{tr} + \|A^t(\sigma, \frac{c}{3t}) - \sigma\|_{tr} \\
\leqslant\ &\frac{c}{3} + c \leqslant 2c \qquad (6.19)
\end{aligned}$$

So (6.14) holds for $d = 2$, and we can use the method from the base case (that is not done yet) to prove our induction step. □.

6.5 Completing the induction

If the proof of the base case and the induction step are correct, we have proven our claim. Then we know that the computational strength of nested CTCs exactly as our upper bound is equal to our lower bound. However, this is of course reliant on Aaronson et al.'s proof being fixed.

7 | Conclusion

Hij stond in het laboratorium van dr. Simiak. De vrouw met de mooie grijze ogen was zijn moeder. De stank in zijn neus was de geur van een half gesmolten materietransmitter. En de man die hem zacht op een stoel neerdrukte, was zijn bloedeigen vader. Het mes viel uit zijn krachteloze hand en bleef trillend in de vloer steken.
Rudolf van Amstelveen was teruggekeerd in zijn eigen eeuw.

Kruistocht in Spijkerbroek
THEA BECKMAN

In this thesis, I have researched what the computational powers are of a system with nested CTCs that behave according to Deutsch's model. I have proposed how the nesting of CTCs would work in his model. Then I have proposed a computational model in this physical model to describe how computation would work in this setting.

I have investigated the computational powers of this model, the TMNC. I have concluded that a TMNC in a spacetime with n nested CTCs can solve all problems of Turing degree 0^n. I have shown this by giving an algorithm that can solve the halting problem corresponding to that Turing degree, for any depth of nesting. If the TMNC is not restricted in the number of nested CTCs, it follows that it can solve the problems of any Turing degree reachable through Turing jumps from the computable problems.

I have also given a conditional result on what cannot be computed in this setting. To do this, I have attempted to find the corresponding consistent states of the CTCs using oracle machines, as Aaronson et al. have done for the case of one CTC. However, I found a problem in the proof that Aaronson et al. give, and neither I nor them have been able to solve this yet. My proof for the limits only works if their proof can be fixed.

If this is the case, however, I show that the a TMNC in n nested CTCs cannot compute anything more than the problems of Turing degree 0^n. This would mean that my proof is tight, in the sense that I show exactly what can and cannot be computed by computers in CTCs.

7.1 Future research

Research in the computational powers of computers in CTCs still needs a lot of work. Firstly, it should be investigated if Aaronson et al.'s proof can be fixed. If it cannot be fixed, then future research should try to find a different bound on the computational power, or attempt to show that one can actually compute more in CTCs.

It also would be interesting to investigate what the computational powers of more exotic spacetime structures would be. For example, how would the bridges and ladders from Chapter 3 affect the computational powers?

The computational complexity of nested CTCs is also still an open question. I believe that as long as the nested CTCs all have polynomial size relative to the input, more than PSPACE can still not be computed, but this remains to be proven.

More research on the physical side of computation is needed as well. Are CTCs in fact physically possible? And is it possible for CTCs to nest within each other? Under what conditions would this be possible?

Bibliography

[1] Antony Galton. "The Church–Turing Thesis: Still Valid After All These Years?" In: *Applied Mathematics and Computation*. Special Issue on Hypercomputation 178.1 (July 1, 2006), pp. 93–102. ISSN: 0096-3003. DOI: 10.1016/j.amc.2005.09.086.

[2] Scott Aaronson, Mohammad Bavarian, and Giulio Gueltrini. "Computability Theory of Closed Timelike Curves". In: *arXiv [quant-ph]* (Sept. 18, 2016). arXiv: 1609.05507.

[3] Robert I. Soare. *Turing Computability: Theory and Applications*. Vol. 300. Springer, 2016. DOI: 10.1007/978-3-642-31933-4.

[4] Alan Mathison Turing. "On Computable Numbers, With an Application to the Entscheidungsproblem". In: *Journal of Math* 58.345 (1936), p. 5.

[5] Alan Mathison Turing. "Systems of Logic Based on Ordinals". In: *Proceedings of the London Mathematical Society, Series 2* 45 (1939), pp. 161–228.

[6] Gábor Etesi and István Németi. "Non-Turing Computations Via Malament–Hogarth Space-Times". In: *International Journal of Theoretical Physics* 41.2 (Feb. 1, 2002), pp. 341–370. ISSN: 1572-9575. DOI: 10.1023/A:1014019225365.

[7] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Higher Education from Cambridge University Press. Aug. 16, 2018. DOI: 10.1017/9781316995433.

[8] Jenann Ismael. "Quantum Mechanics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021. URL: https://plato.stanford.edu/archives/fall2021/entries/qm/ (visited on 07/13/2023).

[9] David Deutsch. "Quantum Mechanics Near Closed Timelike Lines". In: *Physical Review D* 44.10 (Nov. 15, 1991), pp. 3197–3217. DOI: 10.1103/PhysRevD.44.3197.

[10] Peter Pesic. "Einstein and the Twin Paradox". In: *European Journal of Physics* 24.6 (Sept. 2003), p. 585. DOI: 10.1088/0143-0807/24/6/004.

[11] Kurt Gödel. “An Example of a New Type of Cosmological Solutions of Einstein’s Field Equations of Gravitation”. In: *Reviews of modern physics* 21.3 (1949), p. 447.

[12] David Lewis. “The Paradoxes of Time Travel”. In: *American Philosophical Quarterly* 13.2 (1976), pp. 145–152. ISSN: 0003-0481. URL: https://www.jstor.org/stable/20009616.

[13] Scott Aaronson and John Watrous. “Closed Timelike Curves Make Quantum and Classical Computing Equivalent”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2102 (Feb. 8, 2009), pp. 631–647. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.2008.0350.

[14] J. F. Thomson. “Tasks and Super-Tasks”. In: *Analysis* 15.1 (1954), pp. 1–13. ISSN: 0003-2638. DOI: 10.2307/3326643.

[15] Paul Benacerraf. “Tasks, Super-Tasks, and the Modern Eleatics:” in: *Journal of Philosophy* 59.24 (1962), pp. 765–784. ISSN: 0022362X. DOI: 10.2307/2023500.

[16] Mark Hogarth. “Does General Relativity Allow an Observer to View an Eternity in a Finite Time?” In: *Foundations of Physics Letters* 5.2 (Apr. 1, 1992), pp. 173–181. ISSN: 1572-9524. DOI: 10.1007/BF00682813.

[17] Mark Hogarth. “Non-Turing Computers and Non-Turing Computability”. In: *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*. Vol. 1994. Issue: 1. 1994, pp. 126–138. DOI: 10.1086/psaprocbienmeetp.1994.1.193018.

[18] Hajnal Andréka, István Németi, and Gergely Székely. “Closed Timelike Curves in Relativistic Computation”. In: *Parallel Processing Letters* 22.3 (Sept. 2012), p. 1240010. ISSN: 0129-6264, 1793-642X. DOI: 10.1142/S0129626412400105. arXiv: 1105.0047.

[19] Todd A. Brun. “Computers with Closed Timelike Curves Can Solve Hard Problems Efficiently”. In: *Foundations of Physics Letters* 16.3 (2003), pp. 245–253. ISSN: 08949875. DOI: 10.1023/A:1025967225931.

[20] Soumik Ghosh, Arnab Adhikary, and Goutam Paul. “Revisiting Integer Factorization Using Closed Timelike Curves”. In: *Quantum Information Processing* 18.1 (Jan. 2019), p. 30. ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-018-2130-4.

[21] Yaroslav D. Sergeyev and Alfredo Garro. *Single-Tape and Multi-Tape Turing Machines Through the Lens of the Grossone Methodology*. July 15, 2013. arXiv: 1307.3976[cs].

[22] Walter J. Savitch. “Deterministic Simulation of Non-Deterministic Turing Machines”. In: *Proceedings of the first annual ACM symposium on Theory of computing - STOC ’69*. 1969, pp. 247–248. DOI: 10.1145/800169.805439.