FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Models of Hypercomputation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Master of Science (Computational Logic)(MSc)

im Rahmen des Studiums

Computational Logic (Erasmus Mundus)

eingereicht von

Erman Acar
Matrikelnummer 0727317

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Univ. Prof. Dr.phil. Alexander Leitsch

Wien, 02.04.2012 _____ _____

(Erman Acar) (Alexander Leitsch)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Models of Hypercomputation

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computational Logic)(MSc)

in

Computational Logic (Erasmus Mundus)

by

Erman Acar
Registration Number 0727317

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ. Prof. Dr.phil. Alexander Leitsch

Vienna, 02.04.2012 _____ _____

(Erman Acar) (Alexander Leitsch)

Erklärung zur Verfassung der Arbeit

Erman Acar
Schimekgasse 30/2 1230 - Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Dedication

I dedicate this master thesis to my mom who watches me and my family from the heavens.

Acknowledgements

First of all, I would like to express my deep gratitude to my thesis supervisor, Prof. Alexander Leitsch, for accepting to supervise a thesis in such an interdisciplinary area, despite it was not of his primary research interest. He followed me, corrected me and kept giving support in every direction I moved to, in this interdisciplinary research. Without his scientific virtue and wide-knowledge, I should admit that I would not be able to finish this work. I am also thankful for his great patience and understanding, since it needed revision so many times.

I would like to thank to Prof. Karl Svozil, for his very inspiring talks and also lecture series in Quantum Computation. It helped me so much to discover my own interests. I am indebted to him for revising the parts related to physics, when the time was so critical.

I would like to thank to anybody who participated to the small group of "logic and theoretical physics". In general, I am also indebted to all those people who created all those beautiful ideas that I studied, which gave me a true meaning in life. I count myself lucky for being witnessed to their work and understanding.

Finally, I would like to thank to my wonderful family at most and good friends as well, for supporting me, whatever it took for them. Their never ending faith truly gave life into things.

Abstract

After presenting preliminary knowledge in Computability Theory, some versions and interpretations of the so called Church-Turing thesis will be discussed. A mathematical definition of hypercomputation will be given. Some proposed abstract and physical hypermachine models from the literature will be presented. These will be (mainly) infinite time Turing machines, quantum computers and relativistic machine models. Basic principles behind these models and their computational power will be discussed.

Kurzfassung

Der Darlegung von Grundsätzen der Berechenbarkeitstheorie folgt die Diskussion einiger Versionen und Interpretationen der sogenannten Church-Turing-These. Anschließend wird eine formale Definition der Hyperberechenbarkeit gegeben. Daraufhin werden einige abstrakte und physikalische Hypermaschinenmodelle aus der Literatur präsentiert. Diese sind in erster Linie Turingmaschinen mit unendlicher Zeit, Quantencomputer und relativistische Maschinenmodelle. Grundlegende Prinzipien dieser Modelle und ihre Rechenkraft werden diskutiert.

Contents

CHAPTER 1

Introduction

From the very early times of human history until today, the mankind had always had a big enthusiasm in calculation, as a result of his intelligence. From old star calendars to egyptian pyramids, to the antic greek philosophy, ship engineering, war machines, space technology, *calculation* is all along the human history. But this close interrelation between human mind and its very essential nature had a change forever, when it was shown in the first half of the 20^{th} century by the famous results of Turing and Church that there was a so-called *limit* to the things that we could actually calculate. This limit is uttered by *Church-Turing thesis.* It initiated a new mathematical research field which is what we call as *Computability Theory* today.

Computability theory is a research field in mathematics and particularly a branch of mathematical logic. It emerged due to the famous results of Turing, Church, Kleene, Goedel and others. The results that Computability theory had shown are widely accepted since there was no (empirical) counter example.

However in 1990s a new interdisciplinary field, *hypercomputation* emerged, by the discussions of some group of scientists and philosophers. As some creative scenarios against the limits of computation are put by them, this created skepticism and challenge against the Church-Turing thesis, which says there is a limit to *computation* and this limit is Church-Turing computability. In chapter 2, we give give a brief introduction to Computability theory and the class of computable functions, Turing machines and some basic results are mentioned. Furthermore, we introduce the Oracle Turing machine, a hyper computer from the very early days of computability theory and mention relativization results. It will provide us the basic tools to measure the computational power of the hypercomputers regardless of the intrinsic properties of the computational model. These tools are arithmetical hierarchies and Turing degrees.

Church-Turing thesis is an unprovable statement, roughly stating that *what we can calculate is the class of computable functions* or *We can not calculate more than Turing machines* On chapter 3, we mention the approaches, refinements and discussions made about Church-Turing thesis, in a philosophical manner. On Chapter 4, we try to explain the motivation behind this thesis: *why do we bother with hypercomputation?* We also mention a formal definition.
On chapter 5, we will be mentioning two relevant abstract models of hypercomputation. This includes Zeno machines, and Infinite Time Turing machines. It turns out that Infinite Time Turing machines are in power of deciding *analytical hierarchy* We give the formal definition and the basic principles behind it. We provide an appendix for set theoretical preliminary. On the proceeding chapter, we question the physical models. These models are based on the physical theories. We will mention some approaches, where exotic astrophysical objects, such as black holes, are used to create hypercomputers. We will investigate *Schwarzschild metric* to give the intuition in which basic sense, black holes can be useful to create a model of hypercomputation. Afterwards, we will employ an already a practical model , namely quantum computer, and why it is a strong candidate for *hypercomputer*. We will give the circuit representation of quantum computers and analyze in detail, why it promises a new generation fast computing paradigm. Necessary knowledge to quantum circuits are given in Appendix.

On chapter 7, we talk about the possible ways of improving this work.

CHAPTER 2

Basic Computability Theory

Computable functions are the primary objects of study in *computability theory*, and they embody the formalized analogue of the intuitive notion of algorithm. They are crucial in the sense that, any model of computation which has universal power (e.g. *Turing machines, abacus machines etc.*), comes across to the same class of computable functions. Therefore they are useful to study computability without referring to any concrete model of computation, besides to examine the computational power of any particular model of computation, which is to be examined.

The names *computable* and *recursive*, are synonyms; former is the rather new or, more or less contemporary and latter is the earlier naming in the literature. However they are both still in use in the literature. Thus, we will use those two names interchangeably, as far as the intended meaning stays obvious.

2.1 Computable Functions

In this section, we are going to define the class of primitive recursive (**PR**), total recursive (**R**) and partial recursive functions (**P**).

Definition 1 (Primitive Recursive Functions)**.** *Primitive recursive functions are,*

a) the ***initial functions*** *which are:*

- *zero function:* $\mathcal{O}(n) = 0$ *for* $n \in \mathbb{N}$,
- *successor function:* $\mathcal{S}(n) = n + 1$ *for* $n \in \mathbb{N}$,
- *projection functions:* $\mathcal{I}_i^n(\vec{x}) = x_i$ *where* $1 \leq i \leq n$.

b)

$$h(\vec{x}) = g(f_1(\vec{x}), \ldots, f_n(\vec{x}))),$$

*where g and $f_1 \ldots f_n$ are primitive recursive functions. (**composition**)*

c)

$$h(\vec{x}, 0) = g(\vec{x}),$$

$$h(\vec{x}, n+1) = f(\vec{x}, n, h(\vec{x})),$$

*where f and g are primitive recursive functions. (**primitive recursion**)*

As one may understand from the above definition, primitive recursive functions are closed (by definition) under composition and primitive recursion, thus all functions which are generated from them are also primitive recursive functions.

We will write **PR** to denote the class of primitive recursive functions. Most of the functions which are used in number theory, algebra, or applied mathematics in general are in **PR**. So they can be derived or represented by the scheme introduced in the definition. [1] For instance, basic numerical functions corresponding to usual arithmetical operations such as the addition function *add* which could be defined as

$$add(m, 0) = \mathcal{I}_1^1(m),$$

$$add(m, n+1) = \mathcal{S}(\mathcal{I}_3^3(m, n, add(m, n))).$$

and *multiply* can be defined by using *add* as follows,

$$multiply(m, 0) = \mathcal{O}(m),$$

$$multiply(m, n+1) = add(\mathcal{I}_3^3(m, n, multiply(m, n)), \mathcal{I}_1^3(m, n, multiply(m, n))).$$

Those basic numerical functions are just few of them. One might go on to define new functions just by applying the given rules to new ones, to gather any *computable* function, however there will always be some which are not in **PR**. One such function is,

$$Ack(0, n) = n + 1,$$

$$Ack(m, 0) = Ack(m-1, 1), \text{for } m > 0$$

$$Ack(m, n) = Ack(m-1, Ack(m, n-1)), \text{for } m > 0, n > 0.^2$$

which is called *Ackermann function* (named after the mathematician Wilhelm Ackermann). The reason which makes it not computable by means of

[1] $\vec{x}$ stands for $(x_1, \ldots, x_n)$ for arbitrary n.

[2] Note that $Ack(m, n) = Ack_{m-1}^{(n)}(Ack_m(0))$ where $Ack_k = \lambda x.Ack(k, x)$.

primitive recursive functions, is because of its fast growth rate due to its last line *nested recursive* structure. One might check that,

$$Ack(0,0) = 1$$
$$Ack(1,1) = 3$$
$$Ack(2,2) = 7$$
$$Ack(3,3) = 61$$
$$Ack(4,4) = 2^{2^{2^{65536}}} - 3$$

which is obviously a very big number. Fortunately, we are able to bring such functions into our set of computable functions just by adding one new operation to our scheme, with the cost that they might not be total anymore. [3]

Definition 2 (**minimization**)**.** *Let h be a total function, then the function we get by applying the minimization operator μ on h is defined as*

$$(\mu h)(\vec{x}) = y_0$$

if and only if

$$h(\vec{x}, y_0) = 0 \text{ and } \forall y < y_0, h(\vec{x}, y_0) \neq 0.$$

Note that the minimization operator μ[4] , is simply a search operation for the minimum y (denoted as y_0) to have $h(\vec{x}, y) = 0$. Its domain is $D(\mu h) = \{\vec{x} \mid \exists y.h(\vec{x}, y) = 0\}$. So (μh) is undefined for the case where no such y exists. Hence, it is a partial function.

Definition 3 (Partial recursive functions)**.** *The class of partial recursive functions* **P** *is the smallest set of functions defined via initial functions, composition, minimization and primitive recursion.*

We name the subclass of total functions in **P** as *total recursive functions*, denoted as **R**. Note that f is *recursive*, *computable general recursive*, *total recursive*, *total computable* or $f(x) \downarrow$ (means f is *convergent*) for any possible x" are all synonyms, hence they all mean $f \in \mathbf{R}$. Besides, f is (strictly) *partial recursive* and $f(x) \uparrow$ (*divergent*) for some x which also means f is undefined for some x, are also all synonyms.

We can extend our notion of computability of functions to the sets as follows.[5]

[3] We say a function $f : X \longrightarrow Y$ is total if it is defined for any possible $x \in X$, partial otherwise. And for any partial function, when we say A is the domain of f, we mean $f(a)$ is defined if and only if $a \in A$.

[4] Note that the form that we define μ operator is frequently also called μ_0 operator in the literature.

[5] We omit to give the recursiveness definition for relations, since it is analogous to the sets.

Definition 4. *Let A be any set. A is recursive (primitive recursive) if its characteristic function*

$$\chi_A = \begin{cases} 1 & \text{if } a \in A, \\ 0 & \text{otherwise .} \end{cases}$$

is recursive (primitive recursive).

According to the definition above, a given set A being recursive, we are able to decide on a given a whether $a \in A$ or $a \notin A$. The terms *total recursive* and *decidable* are synonyms.

2.2 Turing Machines

Turing machines are abstract computing machines[6], which are able to devise any algorithm, despite their simplicity in design. It rises as the most popular standard machine model, which impose *computability*, thus one uses *algorithm* or *Turing machine*, implying one another in favour of ones own aim, in computer science and mathematics and any other field.

A Turing machine has a *tape*, divided into unit infinitely many cells, and a *tape head*, either reads or writes a symbol, moving one cell to the left or right. Each cell contains exactly one symbol from the *tape alphabet*, a finite set $\Gamma = \{S_0, S_1, \ldots, S_n\}$ of symbols. $Q = \{q_0, q_1, q_2, \ldots, q_n, q_h\}$ is the set of states where *i-th state* of the machine is denoted by q_i and q_0 is called the *initial state* whereas q_h is called the *halting state*. It is customary but also usual that we choose simply two symbols tape alphabet, $\Gamma = \{0, 1\}$ where "0" can also be called as the *blank* symbol (A tape which is fulfilled with only 0 symbol would be regarded as empty.).

Before giving the formal definition of a Turing machine, let us note that we will use a formalism (inspired from the one which is used to represent the concept of *general machine* in [34]), a more general representation of a *machine*, which will be useful in order to define any sort of machine (i.e. computing machine), since Turing machines will not be the only type that we will be dealing with. We will regard a machine, as 5-tuple (Q, S, δ, I, J) such that

- Q is the set of states.
- S is the storage.
- δ is the transition function, in order to define the interactions between states and manipulations over the storage.
- $I : X \to S$ is the input function where X is the input set.
- $J : S \to Y$ is the output function where Y is the output set.

[6]Named after British mathematician Alan M. Turing.

Note that all of the above are machine-specific, therefore one can arbitrarily define them, in order to express the desired model of machine. Now let us give the formal definition of a Turing machine, in terms of the given framework above.

Definition 5 (Turing machine)**.** *A Turing machine, TM is a machine, $(Q, S, \delta, I^m J_n)$ where,*

- $Q = \{q_i \mid 0 \leq i \leq n, n \in \mathbb{N}\} \cup \{q_h\}$ *is the finite set of states with $q_0 \neq q_h$, where q_0, q_h are the initial state and the halting state, respectively.*
- $S = \{(c, i) \in \left(\Gamma^{\mathbb{Z}}\right) \times \mathbb{Z}\}$ *is the storage where Γ=$\{0,1\}$ is the tape alphabet, i is the position of the head and c is the current tape content. Then, $c(j) \in \Gamma$ denotes the current content of the j^{th} cell of the tape, where $j \in \mathbb{Z}$.*
- $\delta : Q \times S \to Q \times S$ *is the transition function, which is a partial function and $d \in \delta$ is in one of the following forms:*
  - $(q_k, (c, i), q_l, (c, i+1))$ *if the tape head is* moving right*.*
  - $(q_k, (c, i), q_l, (c, i-1))$ *if the tape head is* moving left*.*
  - $(q_k, (c, i), q_l, (c', i))$ *if the tape head is* writing/reading

  where q_k is the current state and q_l is the next state with $q_k \neq q_h$ and c' is the new tape content. If $c \neq c'$ then it means the tape head changes the symbol in $c(i)$ (i.e. $c(i) \neq c'(i)$ and $c(j) = c'(j)$ for all $j \in \mathbb{Z} - \{i\}$).
- $I^m : \mathbb{N}^m \to S$ *is the input function:*
  $I^m(x_1, \ldots, x_m) = (\langle 0^{\mathbb{Z}^-} 1^{x_1} 0 1^{x_2} \ldots 0 1^{x_m} 0^{\mathbb{Z}^+} \rangle, 0)$ *where* $(x_1, \ldots, x_m) \in \mathbb{N}^m$.
- $J_n : S \to \mathbb{N}^n$ *is the output function:*
  $J_n(\langle \Gamma^{\mathbb{Z}^-} 1^{x_1} 0 1^{x_2} \ldots 0 1^{x_n} 0 \Gamma^{\mathbb{Z}^+} \rangle, c_2, i) = (x_1 \ldots x_n)$ *where* $(x_1 \ldots x_n) \in \mathbb{N}^n$.

Let us clarify the input-output convention. Note that the input and output sets are defined to be the set of natural numbers: $\mathbb{N}$, as we are interested in numerical functions. The input function guarantees that at the initial state, the tape head is over the 0^{th} cell, and the natural numbers are expressed in unary convention (e.g. number 4 is 1111), and each input is separated by a "0". Notice that tape content is represented by the notation $\langle 0^{\mathbb{Z}^-} 1^x 0^{\mathbb{Z}^+} \rangle$ where $0^{\mathbb{Z}^-}$ denotes the infinitely many 0 to the left (similar for the case infinitely many to the right), and 1^x denotes x-many ($x \in \mathbb{N}$) consecutive 1 over the tape. In the case of the output, the tape head is over the leftmost cell of the output string, which is denoted by i. Now we will introduce a sort of *labeling*, which will help us to write a command list for a Turing machine. This will in turn give us a chance to identify a Turing machine. As the input/output

function, storage, and arbitrary set of states are standard features of a Turing machine, we need to focus only on the transition function in order to identify a Turing machine.

Definition 6. *Let S_1, S_2 be symbols from the tape alphabet ($S_1, S_2 \in \Gamma$), then a command is a sequence of symbols chosen from the list $Q^*; \Gamma^*; L, R$ (L for left and R for right), which is in one of the following forms:*

1. *$q_k S_0 S_1 q_l$ which is interpreted as $(q_k, (c, i), q_l, (c', i))$,*
2. *$q_k S_0, L q_l$ which is interpreted as $(q_k, (c, i), q_l, (c, i+1))$,*
3. *$q_k S_0 R q_l$ which is interpreted as $(q_k, (c, i), q_l, (c, i+1))$.*

Note that the first command is to write the symbol S_1 where the tape head reads S_0. The second and the third commands are simply to denote moving to the left and moving to the right.

Definition 7. *A set C of commands is said to be* deterministic *if for any $q_i S_j Z q_l$, $q_k S_r Z' q_m \in C$, whenever $q_i S_j = q_k S_r$, we have $Z q_r = Z' q_m$ where $Z, Z' \in \{S_n, L, R\}$, holds.*

Now we can give the definition of the *Turing machine program*.

Definition 8 (Turing machine)**.** *A Turing machine (program) T is a non-empty finite deterministic set of commands.*

We write $T(x)$ to denote Turing machine T run on input $x \in \mathbb{N}$ Note that the Turing machine (program) definition we gave is for the *deterministic Turing machine* since the determinacy condition of the command set enforces having only one applicable command at any state.

Notice that every Turing machine program identifies a particular Turing machine, so for the sake of simplicity, from now on, we will use T instead of TM. Let $q \in Q$ and $s \in S$ and the superscript α in δ^α denote α-many applications of δ. [7] Then,

- $\delta^0(q, s) = (q, s)$,
- $\delta^{\alpha+1}(q, s) = \delta(\delta^\alpha(q, s))$,
- $\delta^\beta(q, s)$ is undefined for $\beta > \alpha$ where $\delta^\alpha(q, s) = (q_h, s)$. [8]

[7] We omit the superscript when $\alpha = 1$.

[8] Notice that s is the final content of the storage, since q_h is the halting state.

Then, the computing time of a Turing machine T on input x is

$$t_T(x) = min\{\alpha \mid \exists s : \delta^\alpha(q_0, I(x)) = (q_h, s)\}.$$

Recall that s is the final content of the storage since q_h is the halting state. Domain of t_T is given by

$$D(t_T) = \{x \mid x \in N, \exists \alpha, s : \delta^\alpha(q_0, I(x)) = (q_h, s)\}.$$

Therefore every T which halts on input x defines a (T-computable) function.

Definition 9. *Let $T = (Q, S, \delta, I, J)$. Then the input-output function $Func_T$ is*

$$Func_T(x) = J \circ \mathcal{I}_2^2(\delta^{t_T(x)}(q_0, I(x))),$$

and the domain of $Func_T$ is defined as $D(Func_T) = D(t_T)$.

Now we can give the definition of the *Turing-computable functions.*

Definition 10. *A partial function $f : \mathbb{N} \to \mathbb{N}$ is Turing-computable if there is a Turing machine T such that $f = Func_T$.*

The above definition implicitly stipulates that the regarded Turing machine T does not halt for the input $\vec{n}$ whenever $f(\vec{n}) \uparrow$.

$T = \{$

$q_0 1 R q_0,$
$q_0 0 R q_1,$ } moves to the end of the first argument.

$q_1 0 R q_h,$ } halts if there is no second argument.

$q_1 1 R q_2,$
$q_2 1 R q_2,$
$q_2 0 L q_3,$
$q_3 1 0 q_4,$ } moves to the last digit of second argument and erases it.

$q_4 1 L q_4,$
$q_4 0 1 q_5$ } goes to the '0 between two arguments and changes it to '1'.

$q_5 1 L q_5,$
$q_5 0 R q_h,$ } moves to the leftmost 1 and halts

Figure 2.1: Turing machine which computes the function $f_{add} : \mathbb{N}^2 \longrightarrow \mathbb{N}$: the addition of two number.

2.3 Universal Turing Machines

Every Turing machine computes a fixed particular partial computable function. However, we can encode any Turing machine into a string, and this fact enables us to construct a special Turing machine (Universal Turing machine)

which can take the code of any particular Turing machine and its input, as input and simulates that particular Turing machine running on that input. Such machines are called Universal Turing machines. In this section, we are going to introduce such coding technique and give a definition of a universal Turing machine.

Gödel Numberings

We will construct a coding function $\mathcal{G}$ (for Gödel), for coding each Turing machine by a natural number. So that, we will be to able to construct the Turing machine from its given code (by *prime factorization*), just by evaluating $\mathcal{G}$ on that code. Clearly, such function would be one to one. Let us show the coding of commands. For doing that, we will start with a mapping *val* for assigning integers to symbols which might occur in commands

$val(L) = 3$,
$val(R) = 5$,
$val(q_i) = 4i + 7$ for $i \in \mathbb{N}$,
$val(S_i) = 4i + 9$ for $i \in \mathbb{N}$.

Definition 11. *Let C be any command, consisting of symbols $a_0, \ldots, a_3$. Then the Gödel number of C is*

$$\mathcal{G}(C) = 2^{val(a_0)} \times 3^{val(a_1)} \times 5^{val(a_2)} \times 7^{val(a_3)}$$

.

Now we are able to encode commands. Consider the command $C = q_0 0 L q_1$, where 0 is S_0, then Gödel number of C would be $\mathcal{G}(C) = 2^7 \times 3^9 \times 5^3 \times 7^{13}$. The following result which follows from elementary number theory.

Corollary 1. *For any two given commands E and F, if $\mathcal{G}(E) = \mathcal{G}(F)$ then $E = F$.*

In other words, each command has a unique Gödel number, which will induce soon the same property for Turing machines.

Let us extend our coding to Turing machine. However since it is set of commands, we need to order it to get a unique sequence of commands per set. For that purpose, we may use lexicographic order, $\prec_{lex}$, [9] for that purpose.

- L $\prec_{lex}$ R,
- $R \prec_{lex} q_i$ if $i \in \mathbb{N}$,

[9] Given two partially ordered set A and B, the lexicographical order on $A \times B$ is defined as, $(a, b) \preceq_{lex} (a', b')$ if and only if $a \prec_{lex} a'$ or ($a =_{lex} a'$ and $b \prec_{lex} b'$), for $a, a' \in A$ and $b, b' \in B$. Note that, if A and B are totally ordered, the result is a total order.

- $q_i \prec_{lex} S_j$ if $i, j \in \mathbb{N}$,
- $q_i \prec_{lex} q_j$ if $i, j \in \mathbb{N}$ and $i < j$,
- $S_i \prec_{lex} S_j$ if $i, j \in \mathbb{N}$and $i < j$.

Definition 12. *Let T be Turing machine, where its commands form a sequence up to $\prec_{lex}$ as $C_0, C_1, \ldots, C_n$. Then p_i is the i-th prime and the Gödel number of T is,*

$$\mathcal{G}(T) = \prod_{i=0}^{n} (p_i)^{\mathcal{G}(C_i)}.$$

The following corollary needs to be true in order to claim that our coding works well.

Corollary 2. *No positive integer is a Gödel number both of a command and a sequence of commands.*

Computing $\mathcal{G}^{-1}(n)$ for a Gödel number n with the help of prime factorization gives us the corresponding Turing machine description. Note that it is a primitive recursive function[10]. Now we are close to define a *Universal Turing Machine*.

Definition 13. *Let $e = \mathcal{G}(T')$ for some Turing machine T', then e is called the* index *of T', T' is called e^{th} Turing machine, denoted by T_e. If φ_{T_e} is the partial function which is computed by T_e, it is said to be e^{th} partial computable function, and is denoted by φ_e.*

Definition 14. *Let U be a Turing machine and T_e be a Turing machine given with index e and input $x \in \mathbb{N}$. If $U(e, x) = T_e(x)$ for every e, x then U is called a* universal Turing machine *and the function U computes, denoted by φ_U is called* universal function*.*

Now let us state praised Turing's Thesis (Church Turing Thesis the Turing machine adaptation).

Thesis 1 (Turing's Thesis)**.** *Every function which is computable by an effective procedure is Turing computable.*

Without entering philosophical details or discussions regarding Turing's Thesis or *Church-Turing Thesis* which will be of our concern in the next section, let us try to remark a few points about it. Church-Turing thesis states, whatever is computable, is also computable by a Turing machine. This also emphasizes the existence of a so-called equivalence of computational strength in between some standard models of computation e.g. algorithms, Turing machines, *lambda calculus*, *register machines*. This idea is supported by the

[10]For the proof, see [7].

empirical evidence, and in turn tells us that one such computation model can be taken instead of another, if we are talking of in the sense of computational power only. Considering that, one can state that the Turing machine coincides with the concept of *algorithmic procedure* (or a decision procedure, which is induced by finitely many rules), in the sense of computability.

Theorem 1. *There exists a Universal Turing Machine.*

Proof. By Turing Thesis (or Church-Turing thesis).[11] Consider the following algorithm:
Input: (e, x).

1. Construct T_e for the given e by calculating $\mathcal{G}^{-1}(e)$
2. Compute $T_e(x)$ as output.

By Church-Turing thesis, there exists a Turing machine which computes this effectively computable recipe. This is the universal Turing machine that we are looking for. □

Definition 15. *A universal function φ is called a* Goedel numbering *if for all $f \in \boldsymbol{P}$, there exists a $g \in \boldsymbol{R}$ such that $f(e, x) = \varphi_{g(e)}(x)$ for all $e, x \in \mathbb{N}$.*

2.4 Recursive Enumerability

In a loose sense, enumeration stands as a crucial point in understanding the boundaries of computability. In the next section, we will give an example to an *incomputable* mathematical entity, an incomputable set (or so-called *halting set*), which is however *computably enumerable*.

One might consider π, a *computable real* [12] which has an infinitely long representation, with a possibly non-repeating pattern. All one is able to do is just enumerating the digits one by one. Indeed it refers to a never ending computation process, which yields an infinitely long string. However it is considered to be a computable number since we have an effective procedure to *compute* it. However a natural question would be "Are we able to know everything about such number since it is a computable number?". Here, we refer to its representation. In that sense, an object being computable does not imply that we can know everything about it. However this topic is rather philosophical and not be in our main target.

We will go on with the notion of computable enumerability, and its connection to computability in general. Some chosen basic definitions and properties which are thought to be useful or necessary will be given.

[11] Church-Turing thesis is not necessary to prove the theorem. Therefore its existence is independent of the truth of Church Turing thesis. Interested reader can found the formal proof in [40] as well as in books [7] and [20]

[12] This term is first used by Alan Turing in his seminal article.

Definition 16. *A set X is recursively enumerable if there exists a partial recursive function f such that $D(f) = X$.*

Informally, we say that a set X is recursively enumerable if a partial recursive function is defined for that set X which means X is domain of that function. Or equally and more intuitively, a set X is recursively enumerable if and only if we have an effective procedure (a total recursive function) to list or enumerate its elements. Note that we are only able to confirm if the element is in the set. The terms *semi-recursive*, *semi-decidable* or recursively enumerable or *computably enumerable* are all synonyms.

Theorem 2. *A set $A \subseteq \mathbb{N}$ being recursive implies that A is recursively enumerable.*

Proof. Since A is recursive, there exists a function f such that

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \in A, \\ 0 & \text{if } \vec{x} \notin A. \end{cases}$$

As f is total, we may get a partial recursive function f' by modifying f slightly, which is

$$f'(\vec{x}) = \begin{cases} \vec{x} & \text{if } f(\vec{x}) = 1, \\ undefined & \text{if } f(\vec{x}) = 0. \end{cases}$$

which means in turn $D(f'(A)) = A$, hence A is recursively enumerable.

□

Theorem 3. *$A \subseteq \mathbb{N}$ is recursive if A and $\bar{A}$ both are recursively enumerable.*

Proof. Idea: Since any $a \in \mathbb{N}$ is either in A or $\bar{A}$, and we are able to confirm the positive answer for both sets. This yields a recursive task, to decide on A. □

2.5 Incomputability

Eventually, we will introduce *incomputability* which is the most crucial part of our interest. The term *undecidable*, is a synonym to *incomputable* in terms of classical computability theory, originated from the famous *halting problem* which is the question of "Is there a decision procedure to decide whether or not a Turing machine halts on a given input?" where the answer was shown to be negative.[40, 2] By Turing's thesis in turn, what we understand from an *undecidable problem*, is a particular decision task where we are sure of there is no algorithm or decision method to solve it.

We will explain it by giving some examples of undecidable mathematical objects such as some sets or functions, which are in turn going to yield the very idea of there is no calculation or decision procedure to solve them.

Halting problem

Let us restate the halting problem mentioned, in terms of a set:

$$K_0 = \{(i, x) \mid \text{Turing machine } T_i \text{ halts on input } x\} \tag{2.1}$$

namely the halting set of all Turing machines.

Theorem 4 (Undecidability Theorem)**.** *The set K_0 is undecidable.*

Proof. By Contradiction. Assume that K_0 is decidable. Then there exists a function such that

$$h(i, x) = \begin{cases} 1 & \text{if } T_i(x) \text{ halts,} \\ 0 & \text{otherwise.} \end{cases}$$

Consider the following function defined as

$$g(i) = \begin{cases} 0 & \text{if } h(i, i) = 0, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Since g is a partial recursive function, there exists a Turing machine which computes it. Let e be the index of this Turing machine. So then $g = \varphi_e$
One of the two cases must be true:

- if $T_e(e)$ does halt, then $g(e) = 0$ it implies $h(e, e) = 0$, but it means $T_e(e)$ does not halt which is a contradiction.

- if $T_e(e)$ does not halt, that means $h(e, e) = 0$ implies that $g(e) = 0$, but that means $T_e(e)$ halts which is a contradiction again.

□

Here, the crucial subset of K_0, which leads it to be an undecidable set, namely the index set of the Turing machines which halt in their own code, is usually denoted as K and defined as,

$$K = \{x \mid \varphi_x(x) \downarrow\}. \tag{2.2}$$

It is obvious that the halting function h which was used above is incomputable. It might, at first sight, cause a criticism of its not being a very *natural example* as one of those, a mathematician who works in analysis might encounter. In the next section, we are going to give, so to say, a *more natural* example.

The Busy Beaver Function

Recall the Ackermann function, which was not primitive recursive because it was *dominating* [13] every primitive recursive function (**PR**). At this time, we will give a similar example which is a dominating function for the set of computable functions (**P**), hence it is incomputable.

The *busy beaver* or *Radó* [14] function refers to the greatest number which can be generated by a (halting) Turing machine given its number of *operational states*, which starts working on empty input. Therefore the word *busy* refers to maximum operational *busyness* of an eventually halting Turing machine. What we understand by *operational states* of a Turing machine is simply the set of states except the halting state.

Definition 17. *A k-state Turing machine, $k \in \mathbb{N}$, is a halting Turing machine which the number of operational states is k.*

Now we can safely state the Busy Beaver function.

Definition 18. *The Busy Beaver function, $\Sigma : \mathbb{N} \to \mathbb{N}$, is $\Sigma(n) = max\{\varphi_{T(0)} \mid T \in E_n\}$ where T is a Turing machine and E_n is the set of all n-state machines.*

The function is numerically well-defined since for any n there are finitely many n-state Turing machines [15], therefore any such set must have a maximum element. Before proving its incomputability, let us check the table of the known values below to have a little idea of its extremely fast growing rate.

n	$\Sigma(n)$	*by*
1	1	Radó and Lin
2	4	Radó and Lin
3	6	
4	13	Brady
5	$\geq$ 4098	Buntrock and Marxen
6	$\geq$95,524,079	Marxen

Figure 2.2: First few calculated values of $\Sigma(n)$.

As it can be seen that the last two values could not be found exactly yet, it seems that even rather putting an upper bound looks should be impractical.

[13] Dominating refers to exceeding due to its faster growing speed. Formally let $f, g : \mathbb{N} \to \mathbb{N}$, we say f dominates g if for some $k \in \mathbb{N}$, $k < n$ implies $g(n) \leq f(n)$. If S is a set of functions, we say that f dominates S if f dominates $g(n) + 1$ for every $g \in S$. This implies in turn, if f dominates S, then $f \notin S$.

[14] Due to Tibor Radó who originally introduced it in his paper [31].

[15] Assuming that there is a standard for enumerating the states.

The engineering technicalities [17], of computing Σ will not be our concern, however it is obvious that it is a *very hard* combinatorial problem. We will proceed in parallel to script in [14]. Let us prove the following lemmas which will help us to show that Σ is incomputable.

Lemma 1. $\Sigma(1) = 1$.

Proof. Operational state number equals one, does not allow loop, the best we can do is printing with the initial state and then to the halt state □

Lemma 2. Σ *is strictly increasing.*

Proof. We will show that $\Sigma(n+1) > \Sigma(n)$ for all n. Choose any n, and assume we computed $\Sigma(n)$. We have an n-state Turing machine T with $\varphi_{T(0)} = \Sigma(n)$. Construct a new machine T' by adding a new state to T, which simply adds a single 1 to $\varphi_{T(0)}$. Obviously, $\Sigma(n+1)$ is at least great as $\varphi'_T(0) = \Sigma(n) + 1$, hence $\Sigma(n+1) > \Sigma(n)$ holds. □

Lemma 3. *There is a* c *such that* $\Sigma(n+c) \geq 2\Sigma(n)$ *for all* n.

Proof. We will combine two concrete Turing machine model, in order to find a constant c. First consider T_w which writes specified number (n) of 1s. We can construct it simply by, reserving n-many states each for writing a new 1 whenever it encounters a zero and to make a shift by right whenever it encounter a 1, therefore T_w would have n many operational states. Beware that T_w is rather a scheme or class of Turing machines which tells us there is a Turing machine for any specified number of writing 1.

Second, consider T_d, a 11 state Turing machine which duplicates any given input (For its details, see Appendix: Computability Theory).

We combine T_w and T_d by replacing the halting state of T_w by the starting state of T_d.

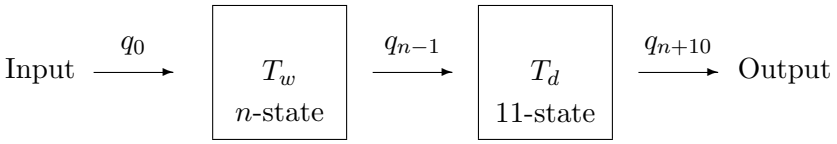


Figure 2.3: The illustration of the combined machine using T_w and T_d.

The combined machine T_{wd} is a $n + 11$ state machine and for any n, $\varphi_{T_{wd}}$ is at least $2n$. Thus, c can be perfectly 11. □

Theorem 5. Σ *is Turing incomputable.*

Proof. By contradiction. Assume that Σ is computable. Then there is a Turing machine which computes Σ. Let us call it $\mathcal{B}$. Then $\mathcal{B}$ has a number of operational states, call it k. Consider the combination of T_w and two replicas

of $\mathcal{B}$ as in the following form:

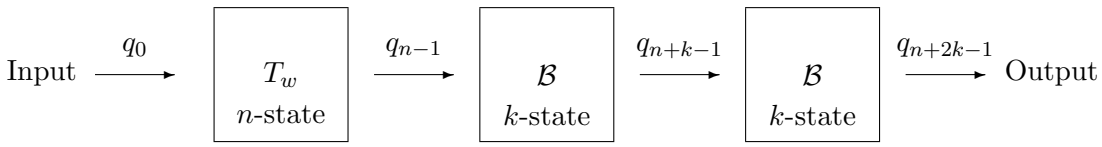


Figure 2.4: The illustration of the combination of a machine, using T_w and $\mathcal{B}$.

which gives us the observation,

$$\Sigma(n+2k) \geq \Sigma(\Sigma(n)) \tag{2.3}$$

for any n. This machine is a $n+2k$ machine. As we had already proved that Σ is strictly increasing, it implies from (1),

$$n+2k \geq \Sigma(n) \tag{2.4}$$

for any n. Let c be as in lemma 3, then we would have,

$$\Sigma(m+c) \geq 2m \tag{2.5}$$

for any m. Applying (2) with $n = m+c$, we get

$$m+c+2k \geq \Sigma(m+c) \tag{2.6}$$

for any m. Using (3) and (4), we have

$$m+c+2k \geq 2m \tag{2.7}$$

for any m. This leads to a contradiction for a value of $m > c+2k$. □

We have shown that our *more natural* example, the busy beaver function is indeed incomputable, however in fact the halting function is in its very core. The reason of looking *more natural* is of any finite string of its values e.g. $\Sigma(1), \Sigma(2) \ldots \Sigma(n)$ for any n is computable. However there is no general algorithm to solve of its all values i.e. n goes to infinity. Recall that we have no general procedure for deciding all non-halting Turing machines, in turn detecting all possible loops.

2.6 Oracle Turing Machines

An Oracle Turing machine (or O-machine) is an abstract machine model which was first proposed by Turing [39] when he was writing his Phd thesis under the supervision of Church, in the late 1930s. He considered the idea of machines

that can perform tasks which any Turing machine cannot. In doing that, the very first hypercomputer model in the history was introduced. However his main aim was to generalize computability theory by creating a *relativized theory of computation.*

In general, Turing introduced a special "Turing" machine which was extended with an extra entity called *oracle* or *black box* in addition to its regular definition. The term *black box* is to express the *inaccessibility* of information that the extra entity has, regarding its any property, working principles etc. Furthermore, an oracle does not have to correspond to any physical entity but just an extra computational resource which we are not interested in its intrinsic property. It must be noted that the question regarding how an oracle works or could be physically implemented is out of concern at the moment. To understand how an O-machine works, the machine might be thought of as having two parts, namely the oracle part and the Turing machine part. The oracle part is the first part, which is for sending a query into and gathering an answer from, which are both carried out by its second part, namely the Turing machine part. That interaction can be formalized, using several extra special states. It must be noted that the question regarding how an oracle works or could be physically implemented is out of concern at the moment.

Definition 19 (O-machine)**.** *An O-machine, T^A is a machine $(Q, S, \delta, I^m J_n)^A$ with oracle A, where,*

- *A is a set of strings (as in tape convention) which corresponds to a subset of $\mathbb{N}$.*
- *$Q = \{q_i \mid 0 \leq i \leq n, n \in \mathbb{N}\} \cup \{q_h, q_?, q_{yes}, q_{no}, q_{copy}\}$ is the finite set of states with $q_0 \neq q_h \neq q_? \neq q_{yes} \neq q_{no} \neq q_{copy} \neq q_0$, where q_0,q_h, $q_?$,q_{yes},q_{no},q_{copy} are the initial state, the halting state, the query state, the answer states: yes and no, and the copy state, respectively.*
- *$S = \{(c, i) \in \left(\Gamma^{\mathbb{Z}}\right)^2 \times \mathbb{Z}\}$ is the storage where Γ=$\{0,1\}$ is the tape alphabet, i is the position of the head and $c = (c_1, c_2)$ is the current storage content. Then, $c_t(j) \in \Gamma$ denotes the current content of the j^{th} cell of the tape number t, where $j \in \mathbb{Z}$ and $t \in \{1, 2\}$ ($t = 1$ is for the* work tape *and $t = 2$ is for the* query tape*).*
- *$\delta : Q \times S \to Q \times S$ is the transition function, which is a partial function and $d \in \delta$ is in one of the following forms:*
  - *$(q_k, (c, i), q_l, (c, i + 1))$ if the tape head is* moving right*.*
  - *$(q_k, (c, i), q_l, (c, i - 1))$ if the tape head is* moving left*.*
  - *$(q_k, (c, i), q_l, (c', i))$ if the tape head is* reading/writing

where $q_k \neq q_h$ *and* c' *is the next tape content.*

- $(q_{copy}, (c, i), q_l, (c', i))$ *if the tape head is* copying *the content of* $c_1(i)$ *to* $c_2(i)$*.*
- $(q_?, (c, i), q_l, (c, i))$ *if the machine is* querying *"*$c_2 \in A$*?" where* $q_l \in \{q_{yes}, q_{no}\}$*. Furthermore,* $q_l = q_{yes}$ *if* $c_2 \in A$*, and* $q_l = q_{no}$ *otherwise.*

- $I^m : \mathbb{N}^m \to S$ *is the input function:*
  $I^m(x_1, \ldots, x_m) = (\langle 0^{\mathbb{Z}^-} 1^{x_1} 01^{x_2} \ldots 01^{x_m} 0^{\mathbb{Z}^+} \rangle, \langle 0^{\mathbb{Z}} \rangle, 0)$ *where* $(x_1, \ldots, x_m) \in \mathbb{N}^m$*.*

- $J_n : S \to N^n$ *is the output function:*
  $J_n(\langle \Gamma^{\mathbb{Z}^-} 1^{x_1} 01^{x_2} \ldots 01^{x_n} 0 \Gamma^{\mathbb{Z}^+} \rangle, c_2, i) = (x_1 \ldots x_n)$ *where* $(x_1 \ldots x_n) \in \mathbb{N}^n$*.*

Notice that the input output convention, is similar to the standard Turing machine. Distinctively in O-machines, we are using one extra tape, namely the *query tape* or the *oracle tape*, which is empty (fulfilled with zeros) initially. In order to send a query to the oracle, we are using the query tape. Especially, by writing the string that we want to query into it. For that purpose, we have the state q_{copy}, and we have two tape heads, having the same position (cell number) on each tape always. [16]The copy state, q_{copy} lets the tape head copy the content of the current cell of the working tape, to the corresponding cell of the query tape. By that way, the machine can copy stepwise the string to be asked, to the oracle tape. And whenever it is required to send a query to the oracle, the query state $q_?$ is handled and it is followed by one of the answer states (q_{yes} or q_{no}) immediately, according to the intended *answer* from the oracle. [17]

Theorem 6. *There exists an oracle Turing machine which can solve the halting problem for any Turing machine.*

Proof. Assume that the machine has the halting set as oracle, such machine would obviously solve any Turing-machine halting problem □

Notice that, if the oracle is an empty set, then the machine is just a usual Turing machine. [18] In general, if an oracle has no extra relevant information, we just have an ordinary Turing machine.

[16]It can also be thought of as one tape head, working on both tape.

[17]Note that we only considered one oracled O-machines, as it serves well enough to our purposes. To handle the case for more than one oracle, one can extend the formalism with further states e.g. $q_{?n}$ for sending query to the n^{th} oracle.

[18]If the oracle tests a recursive problem, again, we get Turing computability (only with a change in *complexity*).

2.7 Turing Degrees and the Arithmetical Hierarchy

In this section, we will introduce two essential concepts, namely *Turing degree* and *arithmetical hierarchy*, which will help us in turn to talk about the computational power of models of hypermachines. Roughly speaking, while the arithmetical hierarchy serves the purpose to measure *definability* of sets of natural numbers, Turing degrees measure the level of uncomputability of sets of natural numbers. We will catch an interlink between those two, by presenting *Post's Theorem*. Then, we will briefly introduce the *second order extension* of the arithmetical hierarchy, so called *analytical hierarchy*, and its relation to the arithmetical hierarchy.

Turing Degrees

If one extends the set of (Turing) computable functions with a characteristic function χ_A for an oracle set A of an oracle Turing machine T^A, one obtains the set of M^A computable functions.

Definition 20. *A function f is said to be,*

i. partial A-computable if f is T^A computable.

ii. A-computable if f is total and T^A computable.

One may define *Turing reducibility* and *Turing completeness* eventually via help of the above definition.

Definition 21. *i. A set B is A-computable (or Turing reducible to A, written as $B \leq_T A$) if χ_B is A-computable.*

ii. A set $A \subseteq \mathbb{N}$ is called Turing hard for a class of sets $\mathcal{X} \subseteq \mathcal{P}(\mathbb{N})$, if $X \leq_T A$ for all $X \subseteq \mathcal{X}$. Furthermore, A is called Turing complete for $\mathcal{X}$ if $A \in \mathcal{X}$ additionally.

Notice that the two close by connected relations $\leq_m$ and $\leq_T$ do not coincide therefore should not be confused. Consider that an expression such, $A \leq_T \bar{A}$ will hold for all $A \subseteq \mathbb{N}$, since asking an oracle will include both the positive and the negative answer, on the other hand we know that $A \leq_m \bar{A}$ does not hold in general (e.g. take $A = K$). However, $\leq_m$ implies $\leq_T$, since for any B and A, if $B \leq_m A$ holds, then one can decide on *if* $n \in B$ (for an $n \in \mathbb{N}$) by simply asking the oracle *Is* $f(n) \in A$*?*.

Corollary 3. *$\leq_T$ is reflexive and transitive.*

Proof. Follows from definitions. □

The following definition is analogous to the relation between *computable* and *computably enumerable* sets.

Definition 22. *A set B is A-computably enumerable if B is the domain of a partial A-computable function.*

This is equal to saying B is A-c.e. (A-computably enumerable), if we can enumerate the members of B via help of an oracle for A. The following, tells us an interesting fact about the *nature* of undecidability.

Corollary 4. *There is no oracle Turing machine which solves its own halting problem.*

Proof. The *diagonalization* method which works for Turing machines in the original undecidability theorem 4 applies. □

No matter which oracle Turing machine one chooses or how much ever one extends the oracle set, we still can obtain a harder unsolvable problem.

This brings us to define the *Turing degree* or the *degree of unsolvability.*

Definition 23. *We say that,*

i. $A \equiv_T B$ *(read as A is Turing equivalent to B) if* $A \leq_T B$ *and* $B \leq_T A$.

ii. the degree of unsolvability or Turing degree for A, is $deg(A) =_{def} \{X \subseteq \mathbb{N} \mid X \equiv_T A\}$

iii. $\leq$ *is an ordering over the collection of degrees, induced by* $\leq_T$ *as follows:* $deg(A) \leq deg(B)$ *if* $A \leq_T B$

iv. $deg(B)$ *is computably enumerable in* $deg(A)$ *if B is computably enumerable in A*

v. A degree is computably enumerable if it contains a computably enumerable set.

We will use sometimes the bold lowercase letter to name a degree, e.g. $\mathbf{a}$ or $\mathbf{b}$ etc. Now we can define the *jump operator* by generalizing the undecidable set K.

Definition 24. *1. Let φ^A denote the Goedel numbering, relativized to a set A. If* $K^A =_{def} \{x \mid \varphi^A_x(x) \downarrow\}$, *then K^A is said to be the jump of A and denoted by A'.*[19]

2. The $i+1^{th}$ *jump of A is defined by,* $A^{i+1} =_{def} (A^i)'$.

3. The jump $\boldsymbol{a}'$ *of* $\boldsymbol{a} = deg(A)$ *is defined by,* $\boldsymbol{a}' =_{def} deg(A')$.

[19]Note that $K^B = K$ for all computable B.

4. *Then,* $n+1^{th}$ *jump* $\boldsymbol{a}^{n+1}$ *of* $\boldsymbol{a}$ *is defined by,* $\boldsymbol{a}^{n+1} =_{def} (\boldsymbol{a}^n)'$.

The following lemma reveals the relation between the jump operation and $\leq_m$.

Lemma 4. *A set* B *is* A*-c.e. if and only if* $B \leq_m A'$.

Proof. $\Rightarrow$ $B = D(\varphi_i^A)$ for some i and $D(\varphi_i^A) \leq_m K^A = A'$ since K^A is m-complete in A-c.e. sets.

$\Leftarrow$ $B \leq_m A' \iff (x \in B \iff f(x) \in A' = K^A) \iff f(x) \in dom(\phi^A) \iff x \in dom(\phi^A \circ f)$ where $\phi^A(x) = x$ if $\varphi_x(x) \downarrow$ and *undefined* otherwise. Thus, B is A-c.e.

□

Lemma 4 implies, $A \leq_m A'$, since A' is A-c.e..

Theorem 7. *For a degree* $\boldsymbol{a}$*, there exists an increasing chain of degrees, in the form of* $\boldsymbol{a} < \boldsymbol{a}' < \boldsymbol{a}'' < \ldots \boldsymbol{a}^i < \boldsymbol{a}^{i+1} \ldots$*, where* $\boldsymbol{a}^{i+1}$ *is c.e. (computably enumerable) in* $\boldsymbol{a}^i$.

Proof. We have to show [i.] $\mathbf{a}^{i+1}$ is c.e. in $\mathbf{a}^i$, [ii.] $\mathbf{a}^i \leq \mathbf{a}^{i+1}$ and [iii.] $\mathbf{a}^{i+1} \not\leq \mathbf{a}^i$

i. Let $\mathbf{a}^i = deg(A)$, then the jump of $\mathbf{a}^i$, $\mathbf{a}^{i+1} = \{y \mid \varphi_y^A(y) \downarrow\}$ by definition 24. Then define $\phi^A(y) = y$ if $\varphi_y^A(y) \downarrow$ and undefined otherwise. As ϕ^A is A-p.c. and $a^{i+1} = D(\phi^A)$, $\mathbf{a}^{i+1}$ is c.e. in $\mathbf{a}^i$.

ii. Follows from definition 23, and also by lemma 4 (Recall that $\leq_m$ implies $\leq_T$).

iii. Let us assume that $\mathbf{a}^{i+1} \leq \mathbf{a}^i$ and try to get a contradiction. If $\mathbf{a}^{i+1} \leq \mathbf{a}^i$, then $K^A \leq_T A$ for $\mathbf{a}^{i+1} = deg(K^A)$ and $\mathbf{a}^i = deg(A)$, by definition 24 and i. But K^A is not A-computable, and by definition 23, we get the contradiction.

□

Notice that a set is c.e. if and only if it is c.e. in some computable set. So a degree $\mathbf{a}$ is c.e. if and only if it is c.e. in $\mathbf{0}$. It is intuitive to denote the degree of an ordinary Turing machine, by $deg(\emptyset)$ since its oracle can be thought of as empty set. Obviously, this is the degree of any computable set. We will denote it by $\mathbf{0}$. Then using the jump operator, one gets $\mathbf{0}'$ which is the degree of K. The chain of degrees we would get, by applying the jump operator, which looks like $\mathbf{0} < \mathbf{0}' < \mathbf{0}'' < \ldots \mathbf{0}^i < \mathbf{0}^{i+1} \ldots$, has a special name which is called *Turing hierarchy*. We will use it to define *arithmetical hierarchy* and then set up a link with *computability* in between.

Arithmetical Hierarchy

One can use the arithmetical hierarchy to classify the degree of unsolvability of decision problems, by linking them to some certain class of sets of natural numbers. Let us give the set theoretic representation of the arithmetical hierarchy.

Definition 25. *The arithmetical hierarchy consists of the classes* Σ_n^0 , Π_n^0 *and* Δ_n^0 *for* $n \geq 0$*, which are defined as follows:*

1. $\Sigma_0^0 = \Pi_0^0 = \Delta_0^0$ *is the class of all computable sets.*

2. Σ_{n+1}^0 *is the class of all such sets* A *that* $A \in \Sigma_{n+1}^0$ *if there exists a* $B \in \Pi_n^0$ *s.t.*
$$A = \{x \mid (\exists y)(x, y) \in B\}.$$

3. Π_{n+1}^0 *is the class of all such sets* A *that* $A \in \Pi_{n+1}^0$ *if there exists a* $B \in \Sigma_n^0$ *s.t.*
$$A = \{x \mid (\forall y)(x, y) \in B\}.$$

4. $\Delta_{n+1}^0 = \Sigma_{n+1}^0 \cap \Pi_{n+1}^0$.

Notice that computably enumerable sets are those in Σ_1^0, e.g. K_0. In general, Σ_{n+1}^0 is the class of sets that are A-computably enumerable for some set $A \in \Sigma_n^0$. The following properties hold for the *arithmetical hierarchy* (of sets of natural numbers):

- Σ_n^0 and Π_n^0 are closed under finite unions and intersections of their respective elements.

- A set is Σ_n^0 iff its complement is Π_n^0.

- $\Delta_n^0 \subset \Sigma_n^0$ and $\Delta_n^0 \subset \Pi_n^0$, for $n \geq 1$.

- $\Sigma_n^0 \subset \Sigma_{n+1}^0$ and $\Pi_n^0 \subset \Pi_{n+1}^0$, for all n and $\Sigma_n^0 \cup \Pi_n^0 \subset \Delta_{n+1}^0$, for $n \geq 1$.

Due to the last two property, the word *hierarchy* is involved, and it does not collapse. Notice that one can also describe a set A as arithmetical if $A \in \bigcup_{n \geq 0} \Sigma_n^0 \cup \Pi_n^0$. Considering the property regarding complements above, Π_1^0 is known as *co-computably enumerable* (or *corecursively enumerable*).

One can also give a representation by assigning the degree of unsolvability to the measurements of *definability* of certain sets of natural numbers. In doing so, sets, to be classified, are defined by logical formulas in the language of *first order arithmetic*. The language of first order arithmetic, consists of predicate symbols $=, <$, the constant symbol $\bar{0}$, a unary function symbol (for

the successor function) $\bar{S}$ and binary function symbols $\bar{+}$ and $\bar{*}$.
[20]

Definition 26. *i. A is Σ^0_n-complete if $A \in \Sigma^0_n$ and $X \leq_m A$ for every $X \in \Sigma^0_n$.*

ii. Π^0_n and Δ^0_n completeness are defined similarly.

We have to define *relativized arithmetical hierarchy* before linking arithmetical hierarchy to Turing degrees.

Definition 27. *$\Sigma^A_0 = \Pi^A_0 = \Delta^A_0$ is the class of all A-computable sets. And for $n \geq 0$,*

1. *Σ^A_{n+1} is the class of all such sets X that $X \in \Sigma^A_{n+1}$ if there exists a $Y \in \Pi^A_n$ s.t.*
$$X = \{x \mid (\exists y)(x, y) \in Y\}.$$
2. *Π^A_{n+1} is the class of all such sets X that $X \in \Pi^A_{n+1}$ if there exists a $Y \in \Sigma^A_n$ s.t.*
$$X = \{x \mid (\forall y)(x, y) \in Y\}.$$
3. *$\Delta^A_{n+1} = \Sigma^A_{n+1} \cap \Pi^A_{n+1}$.*

Properties of hierarchy regarding union, complement and subset are analogous. The following theorem reveals a connection between *Turing degrees* and the *arithmetical hierarchy.*

Theorem 8 (Post's theorem)**.** *For any $A \subseteq \mathbb{N}$ and $n \geq 0$,*

i. $A \in \Sigma^0_{n+1} \iff A$ is c.e. in $\boldsymbol{0}^n$.

ii. $\boldsymbol{0}^{n+1}$ is Σ^0_{n+1}-complete.

Proof. i. By induction on n. For induction base $n = 0$.

$\Rightarrow$ $A \in \Sigma^0_1$ if there exists $B \in \Pi^0_0$ (B is computable) s.t. $A = \{x \mid (\exists y)(x, y) \in B\}$. Consider the following function,

$$\phi(x) = \begin{cases} 1 & \text{if } (\exists y)\chi_B(x, y), \\ undefined & \text{otherwise.} \end{cases}$$

[20] The logical formula representation of the arithmetical hierarchy for Σ^0_n can be given as follows (the rest is straightforward):

$$A \in \Sigma^0_n \text{ if there is a formula } \phi[z] : \exists x_1 \forall x_2 \ldots Q x_n \psi \text{ s.t. } K \in A \iff \mathcal{N} \models \phi(\bar{K}),$$

where $Q \in \{\exists, \forall\}$ and $\mathcal{N}$ is the standard model of arithmetic, with the domain $\mathbb{N}$, such that $<$ is interpreted as less than, $\bar{0}$ is interpreted as 0, 1 is expressed by $\bar{S}(\bar{0})$, 2 is by $\bar{S}(\bar{S}(\bar{0}))$,... etc. In addition, for all $n \geq 1$, we define $\bar{n} := S^{(n)}(\bar{0})$ where $S^{(0)}(\bar{0}) := \bar{0}$, $S^{(n+1)}(\bar{0}) := \bar{S}(S^{(n)}(\bar{0}))$.

$\Leftarrow$ A is c.e. $\iff$ $A = ran(f)$ for some computable f $\iff$ $(\exists y)f(y) = x$ $\iff$ $(\exists y)(y,x) \in f$ and $f \in \Pi^0_0$ since f is computable $\iff$ A is Σ^0_1.

For the induction hypothesis, suppose that $n > 0$ and $X \in \Sigma^0_n \iff X$ is c.e. in $\mathbf{0}^{n-1}$ for all n.

$\Leftarrow$ Suppose that $A \in \Sigma^0_{n+1}$, then there is a $B \in \Pi^0_n$ such that $x \in A \iff (\exists y)B(x,y) \iff A \in \Sigma^B_1$ by definition 27. Note that $A \in \Sigma^B_1 \iff A \in \Sigma^{\bar{B}}_1$. It follows that $A \in \Sigma^{\bar{B}}_1$ where $\bar{B} \in \Sigma^0_n$. (Since B was in Π^0_n). So by inductive hypothesis, $\bar{B}$ is c.e. in $\mathbf{0}^{n-1}$, and hence by lemma 4 we get $\bar{B} \leq_m \mathbf{0}^{(n-1)'} = \mathbf{0}^n$. It follows that $A \in \Sigma^{\mathbf{0}^n}_1$, therefore it is c.e. in $\mathbf{0}^n$

$\Rightarrow$ Assume that A is c.e. in $\mathbf{0}^n$, We get by induction hypothesis $\mathbf{0}^n \in \Sigma^0_n$, since $\mathbf{0}^n$ is c.e. in $\mathbf{0}^{n-1}$, and $\mathbf{0}^n \in \Sigma^0_n$. $x \in A \iff x \in D(\varphi^{0^n}_x)$ iff $\exists s \exists y_1 \dots y_k, z_1, \dots, z_l$ such that we can decide $x \in D(\varphi^{0^n}_{i,s})$ using oracle answers: $y_1 \dots y_k \in \mathbf{0}^n \wedge z_1 \dots z_l \in \bar{\mathbf{0}^n}$ where $y_1 \dots y_k \in \mathbf{0}^n \in \Sigma^0_n$ and $z_1 \dots z_l \in \bar{\mathbf{0}^n} \in \Pi^0_n$ Quantifier manipulation leads to $A \in \Sigma^0_{n+1}$. [21] implies that $A \in \Sigma^0_{n+1}$.

ii From i and lemma 4, it follows that $A \in \Sigma^0_{n+1} \iff A$ is $\mathbf{0}^{(n+1)}$c.e. $\iff A \leq_m \mathbf{0}^{(n+1)}$. Together with $\mathbf{0}^{(n+1)} \in \Sigma^0_{n+1}$, it implies that $0^{(n+1)}$ is Σ^0_{n+1}-complete.

□

The *analytical hierarchy* will not be in the scope of our concern. Nevertheless, let us mention few things about it before closing this section.
The analytical hierarchy is a second-order equivalent of the arithmetic one. Therefore, quantifiers range over function variables. The analytical hierarchy is denoted by replacing superscript 0 by 1 in the notation of arithmetical hierarchy, i.e. Σ^1_n. Furthermore, basic properties regarding hierarchy, are straight forward as it was in arithmetical hierarchy (e.g. ϕ is Σ^1_{n+1} iff $\neg\phi$ is Π^1_{n+1}, and $\Sigma^1_n \subset \Sigma^1_{n+1}$).

Δ^1_1 is known as the class of *hyperarithmetic* sets, in the literature. The following fact tells us analytical sets are of higher level of unsolvability, due to definability.

Fact 1. *The arithmetical sets are properly included in* Δ^1_1.

Interested reader can check its proof in [26].

[21] s stands for the boundary of the number of steps of computation (As in Kleene predicate.). The proof is taken from [3]

CHAPTER 3

On the Church-Turing Thesis and the Nature of Computation

All the ideas behind the term hypercomputation, begin with a famous statement of mathematics which is known as the Church-Turing Thesis (will be called as the Thesis in sequel) today, and it is stated as follows.

Thesis 2. *Every effectively calculable function is a computable function.*

The Thesis is indeed based on an empirically robust concept of Theory of Computation today, namely all classes of lambda definable functions, Turing computable functions and partial computable functions are the same, as it is widely accepted today. However, the part in the statement, *effectively calculable* is not purely formal, therefore it is considered intrinsically unprovable since the conceptual *impossibility* of a possible mathematical proof relating an intuitive argument to a mathematically defined one. Thus it caused ambiguity and doubt at some level. As the time went by, not only so many discussions over its meaning have been made so far but also many variants of it took place in the literature by trying to replace it by *more proper* versions given by researchers belonging to different disciplines (mathematics, philosophy, physics). Thus to understand the Thesis and to consider its relevant variants and extensions, is in the center of understanding the field of hypercomputation, and carrying out further discussions based on it, since every particular model of hypercomputation is based on a particular Thesis variant or Thesis itself.

In this part, we will be touching on some rather philosophical aspects of computation and Church-Turing Thesis. It will be opening with a brief historical introduction and afterwards moving on with some concepts like *effectiveness*, Thesis (abbreviation for Church-Turing Thesis) and some of the

its relevant variants. In doing so, we will also mention some quotes of the pioneers sometimes, in order to shed some light from a different angle on the ideas under discussion.

3.1 A Brief History

In 1930s, one of the most important problems was David Hilbert's Entscheidungsproblem, which was asking if there exists a method or algorithm to decide validity of formulas in First Order Logic (FOL). As Church stated Entscheidungsproblem proof-theoretically as follows:

> By the Entscheidungsproblem of a system of symbolic logic is here understood the problem to find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system.

Symbolic logic refers to FOL and the importance of the Entscheidungsproblem arose from the need of being sure of the correctness of mathematical proofs with no doubts. Hence, a formal system were to use for that purpose; to express axioms of mathematics in a formal system and to provide formal proofs for theorems derived from those axioms in that formal system. First order logic was such a formal system available and the validity of formulas in FOL was coinciding with the truth of mathematical statements in that sense.

The first negative answer to that question was given by Alonzo Church in 1936, by introducing the *λ-definable functions*. A very short time after Church, Turing, independently from Church's work, has given a parallel answer to that question by introducing his own assertion, a class of numbers called *computable numbers* which is *effectively computable* with his abstract machine (Turing machine) he introduced.
But what should we understand by a method which is *effective*? As it is stated on a famous article about the Thesis [4]:

Definition 28 (Effective Procedure)**.** *A method, or procedure, M, for achieving some desired result is called* effective *or* mechanical *just in case*

1. *M is set out in terms of a finite number of exact instructions (each instruction being expressed by means of a finite number of symbols),*

2. *M will, if carried out without error, always produce the desired result in a finite number of steps,*

3. *M can (in practice or in principle) be carried out by a human being unaided by any machinery save paper and pencil,*

4. M demands no insight or ingenuity on the part of the human being carrying it out.

As it is obvious from the context, the term mechanical and effective are used as synonym. For an effective method, we can give the example of *satisfiability testing for propositional logic* or a naive computer program which solves a given sudoku problem. Early before the Thesis takes its last form, Turing and Church had expressed it individually in their own ways. It was stated by Turing in an unpublished paper[Turing 1948][1],

> LCMs can do anything that could be described as 'rule of thumb' or 'purely mechanical'.

which is known for Turing's Thesis. Here the abbreviation LCM stands for *logical computing machine* which was denoted particularly as Turing Machine as it is known today. He argued his thesis in his paper with the concept, the computability of a Turing Machine which corresponds to any mechanical method which is used to get the value of a mathematical function, could also be computed by Turing Machine. To him a Turing Machine as its computational power was an equal of a human mind i.e. a clerk which has enough paper (space) and time, with having no insight other than being capable of carrying out the mechanical processes which he has to.
Church had defined it in a different way,

> A function of positive integers is effectively calculable only if recursive.

which is weaker than Turing's, according to its concern on recursive functions of only positive integers. The vague term 'effective calculability' corresponds to the term 'computability' of what Turing showed in his famous paper [40].

> In a recent paper Alonzo Church has introduced an idea of 'effective calculability', which is equivalent to my 'computability', but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem. The proof of equivalence between 'computability' and 'effective calculability' is outlined in an appendix to the present paper.

Following Turings own statement given above, further statements of agreement on equality of Church's and Turing's theses occurred in literature. Nevertheless, we will take the definition of Church-Turing Thesis in Turing's version as it is formulated below:

Thesis 3 (**Church-Turing (revisited)**)**.** *Every function which would be naturally regarded as computable can be computed by a Turing machine.*

[1] It is published in 1968, 14 years after his death.

Here the word 'naturally' can be replaced simply by 'normally' or 'obviously', without disturbing the meaning in a loose sense. Technically speaking, this does not seem to violate what we do understand from the class of functions which can be calculated 'effectively' or 'mechanically', since effectively computable functions are those which can be considered as 'naturally regarded as computable' or 'naturally computable' in short. But conversely, it is not so obvious that all the 'naturally computable' functions are 'mechanical' (more probably, it was more certain when we consider the discussions which going on at the time the Thesis had been asserted).

Nevertheless what Thesis means by 'computable' is still somewhat vague or too general by means of any bound or any type of system, say, if it is calculated by human, or an abstract computing concept, or a machine in a general sense. Thus, it is quite natural to state that the Thesis can also have implicit assertions concerning the current physical theories. The versions of the Thesis which will be mentioned in the following section, vary regarding some extra specifications and restrictions. As there are many in the literature, we will filter some of them which are believed to have less relevance in connection to any type of hypercomputation.

3.2 Some Other Theses

Now we will mention some other theses which are related to the Thesis in their origin and emerged as a result of some modifications made. The explanations to be made will exclude their direct link to the particular model of hypercomputation for the moment.

Thesis M

Gandy, in his article [13], tries to give a more convenient version of the Thesis, which is:

Thesis 4 (Thesis M for 'mechanism')**.** *What can be calculated by a machine is Turing-machine computable.*

In doing so, he defines principles[2] which are assumed to being satisfied by any mechanical device whenever the device will have the computational power of a Turing-machine. In doing so, his aim was to bring more concrete notion to the discussion of arguments of his thesis than the arguments of the original one (Thesis). However, we will not mention them here, because there is a counterexample given for them[3]. Nevertheless, this is no refutation of Thesis

[2]Gandy's four principles, namely, form of description(I),limitation of hierarchy (II), unique reassembly (III) and local causation(IV) (see [13] or [36]).

[3]Fourth Principle is falsified by Newtonian devices, as he said himself:" I am sorry that Principle IV does not apply to machines obeying Newtonian mechanics."

M, nor enough to trivialize its relevance by means of further discussions in which hypercomputation will take place.

Since the word *machine* is used, some amount of physical reality is involved[4] rather than just conceptual ones e.g. Turing-machines, register machines. Thus he had to clarify what he refers to by using it. He states that he will consider only *deterministic discrete mechanical devices*, so then we may re-state it in his intended version which will make stricter sense by means of his argumentation.

Definition 29 (Gandy's Thesis)**.** *What can be calculated by a discrete deterministic mechanical device is Turing-machine computable.*

Even though Thesis M and Gandy's usually coincide in literature (since they both are introduced by Robin Gandy), it is reasonable to make distinction among them. In particular, Gandy's Thesis is a more special version of Thesis M, because the latter encompasses analog machines which will not be the concern of this work.

Weak Church-Turing Thesis

Informally speaking, in *computational complexity* the class of decision problems which are solvable by a Turing machine whose *running time* grows polynomially in the length of the input, is called P. The problems which are in this class are usually referred to as *feasible*, or *tractable*. Straightforwardly, a problem is said to be *intractable*, if it is known to be not tractable e.g. running time requires hyper-polynomial number of steps. The following thesis, also called *Karp-Cook thesis* is a narrower variation the Church-Turing Thesis, emphasizing what is *feasible to computable?*.

Thesis 5 (Weak Church-Turing Thesis)**.** *Tractable problems are those that are in the class P (polynomial).*

In other words it stipulates there can be no intractable problem which can be solved in polynomial time.

The Church-Turing Principle

In his paper [9], Deutsch argues that there is an implicit physical assertion in the Thesis, and he modifies the Thesis in a sense for the aim what is to make this assertion explicit. He proposes an interpretation intended to be more convenient than the Thesis itself. He named it as Church Turing Principle (it is called Church Turing Deutsch principle by others) instead and asserted the new statement as calling it physical or stronger Church Turing

[4]In that sense, it is important to stress that Thesis M does have an account of being a physical interpretation of the Thesis.

thesis in [9].[5] But we will call it Church-Turing Principle. The reader should not be concerned with the details relating to the working principles (inner machinery) of the quantum computer for now. It will be explained in further sections reserved to quantum computer as a model of *weak* hypercomputer.

Definition 30 (Church-Turing Principle)**.** *Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.*

According to Deutsch, Quantum theory (which is discrete) is compatible with the Church Turing Principle (CTP) and since classical physics is continuous and universal Turing machines are discrete, CTP would be falsified. Thus, he tries to convince that the quantum generalization of the class of Turing machines (borrowing the terminology Deutsch is using) namely the universal quantum computer Q, is capable of *perfectly simulating* every *finitely realizable* physical system.

A *finitely realizable physical system* refers to any physical object which *experimentation* is possible. A physical system S is perfectly simulated by a computing machine M if there is a program $\pi(S)$ for M which turns M into a black box which is functionally (input-output relation) indistinguishable from S.

Furthermore, he states in [9] that, a computing machine is said to be operating by finite means if

i) At any time only a finite subsystem is in process in the computation,

ii) the process depends only on the state of a finite subsystem,

iii) the rule that specifies the process performed can be represented by *finite* means.

According to Deutsch, his reformulation of Church Turing thesis is more accurate in the sense of, whilst the original Thesis has implicit *physical* implications, it does not say anything about the *physical* issues regarding the *computable*. CTP, on the other hand, refers exclusively to objective concepts such as *measurement*, *preparation* and *physical system* since it is based on the very idea of any computational process takes place in the physical universe, therefore *computation* has to obey the laws of physics. In that sense, it replaces the terminology like *would naturally be regarded as computable*, which does not fit well into the existing structure of physics, with *in principle be computed by a real physical system.*

[5] Calling the original Church-Turing thesis as a principle is more common among physicist, since it resembles principles of Physics and also physicists are more interested in physical implications of it. Deutsch explains it, by giving concrete examples of similarity i.e. Third Law of Thermodynamics. See also [38].

Although Deutsch has defended his point of view using quantum mechanics only, the statement itself is general by means of physical systems, therefore it has to be *independent* of any particular physical system. Thus one can naturally ask, given any physical theory whether or not the CTP is true in that particular theory? This in turn means, is it possible to construct a universal computing device that can simulate an arbitrary finitely realizable physical system, also within the theory chosen? Although Deutsch concluded that classical physics (Newtonian) would falsify CTP, Fredkin and Toffoli [12] showed that their *billiard ball* model of computation (using Newtonian physics) which is indeed equal to the computational power of Turing machines which is widely regarded as the universal computing power, and furthermore, the work of Pour-El and Richards [30] showed that Turing machines can simulate all of Newtonian-physics. Today we know that Newtonian physics is wrong, nevertheless CTP not just for Computability theory but also has so many possible interesting consequences in physics and mind philosophy[6] which those topics however will not be in the scope of our concern.

[6]Take *human brain* as finitely realizable system, which in turn CTP would tell us, *human mind is Turing computable* which is the corner stone of *Computationalism*.

CHAPTER 4

Hypercomputation

In prequel, we had already several times pronounced and roughly described *what is hypercomputation*. It was either hypothetical or a possibly physical machine, or even a physical process which exhibits a behaivour of computing a non-Turing computable function. As you might remember, we had already given a model of hypercomputer, despite the fact that we had not drawn a mathematically rigourous schematics. In this section, we will try to bring a more concrete notion to hypercomputation, by giving formal definition.

Apart from that we will try to convince the reader that *why we bother ourselves by hypercomputation?*, is a question, which has indeed a worthy answer. Let us start from that part.

4.1 Why do we bother with Hypercomputation?

Hypercomputation is a young interdisciplinary field that is focused on the investigation whether computing beyond the Church-Turing limit is possible. This includes various both mathematical and physical methods for the computation of non-Turing-computable functions. Aside from the possibility of physical existence of hypercomputation, this research field can be taken as a hypothetical research field since computability theory which Turing initiated by [40, 39], deals with computable (or relatively computable) problems and analyzes how much computational resources are required to solve them, up to some standard Turing machine models, as we had seen in previous chapters. Therefore the inner machinery of whatever model of computation causing the regarded computing strength remains unrevealed. However, hypercomputation tries to propose unconventional models of computation along with describing their inner machinery (in some detail) which may offer solution strategies for non-computable problems as well as helping to better understand the nature of some problems in mathematics, computer science, [28]

e.g. $\mathcal{P} \neq \mathcal{NP}$ for infinite time Turing machines [5, 8].
The following quote has been stated by Rosen [33], Kreisel [21] and Davis [7].

> "... how can we ever exclude the possibility of our being presented, some day (perhaps by some extraterrestrial visitors), with a (perhaps extremely complex) device or oracle that computes a non-computable function?"

This rather rightful thinking has very basic motivation of science basically. Particularly, it can be a major concern for physics such that whether our universe might inherit (Turing) incomputable physical activity, regardless of we can recognize or not. The close connection between the physical systems and computability was already mentioned in [9]. This brings to our minds, the question of "How can one acknowledge hypercomputation?" This will lead us to bring the formal definition of hypercomputation in an approach, that assuming we do not have any knowledge about any intrinsic property of its inner machinery. Therefore it is called *black box* approach. This section will follow, mainly using the reference [23]

4.2 A Formal Definition

Let B be a subset of $X_1 \times \ldots \times X_m$. The i-th projection of B (for $i = 1, \ldots, m$), written as B_i, is defined by:
$B_i = \{x \mid x \in X_i, (\exists y \in X_1 \times \ldots \times X_{i-1})(\exists z \in X_{i+1} \times \ldots \times X_m)(y, x, z) \in B\}$.
For any $x \in N^m$ we define, $\mid x \mid = \max\{x_i \mid i \in \{1, \ldots, m\}\}$ Now the following definition might be regarded as *a behavioral definition* of hypercomputation.

Definition 31. *Let X, Y be sets and $\mathbb{N}$ be the set of natural numbers. A subset B of $X \times Y \times \mathbb{N}$ is called a* black box *if $B_1 = X$. Furthermore, X is called the input set and Y the output set.*

Notice that we have mapped every possible input-output to a unique natural number, and $B_1 = X$ is total, therefore an output always exists.

Definition 32. *Let B be a black box. We define*
$f_B = \{(x, y) \mid (\exists z)(x, y, z) \in B\}$, $t_B = \{(x, z) \mid (\exists y)(x, y, z) \in B\}$. *$f_B$ is called the input-output relation of B and t_B the computing time of B. If f_B and t_B are functions then B is called deterministic.*

Now we can define the concept of hypercomputer.

Definition 33 (hypercomputer)**.** *A hypercomputer is a black box B where f_B is not Turing-computable.*

Now the next aim is to define a machine, which will only violate the weak Church-Turing thesis (or Karp-Cook) thesis.

Let C be a class of computable monotone functions $\mathbb{N} \longrightarrow \mathbb{N}$ containing the polynomials (over$\mathbb{N}$ with non-negative coefficients). Then C is called a bound class.

Definition 34 (Weak hypercomputer)**.** *A weak hypercomputer is a black box B with the following property: There exists a bound class C such that*

- *$t_M(x) > g(|x|)$ almost everywhere for all $g \in C$ and for all Turing machines M with $f_M = f_b$.*
- *There exists an $h \in C$ such that $t_B(x) \leq h(|x|)$ for all $x \in B_1$.*

Due to its definition, a weak hypercomputer works *faster* than any Turing machine. Consider the following scenario,

- f_B is an **EXPTIME**-complete problem, therefore there exists no polynomial p and no Turing machine M computing f_B with $t_M(x) \leq p(|x|)$ for all $x \in X$, but $t_B(x) \leq p(|x|)$ for all $x \in X$ and for a polynomial p.

CHAPTER 5

Abstract Models of Hypermachines

In this section, we will be introducing several abstract model of hypercomputer. Our notation will be machine specific.

5.1 Zeno Machines

Among the approaches of hypermachines, Zeno Machines (also known as Accelerated Turing Machines) are the one of the most frequently discussed. It is originally abstract model. It has been raised first by Hermann Weyl in 1927, regarding the idea of a Turing Machine model which has the capability of executing an infinite amount (countably) of execution steps in finite amount of time. In doing so, the shared idea is to execute every algorithmic successor step in exponentially lesser amount of time of the predecessor step.

Definition 35 (Zeno Machine)**.** *A Zeno Machine (ZM) is a TM, except that it takes* n^{-i}*units of time to execute its i-th step, where* $n \in \mathbb{N} - \{0, 1\}$ *and* $i \in \mathbb{N}$*.*

For instance, for $n = 2$, it will take for the first step 1 unit of time, $1/2$ for the second step, $1/4$ for the third, $1/8$ etc. Therefore, since for $i \in N$, $\sum_{i\in\mathbb{N}}^{j} 2^{-i} < 2$, the machine will already be performed finitely required number of steps. However to perform infinitely many step will take exactly 2 units of time (i.e. $\sum_{i\in\mathbb{N}} 2^{-i} = 2$).

The reader may realize here that ZM is not a TM, based on its definition above, it has an exceptional property in which it differs from a TM.

In many variational scheme to this model, it has been suggested to solve the halting problem by performing the following *super-task* on it.

```
begin
write 1 to the first cell of the tape (output)
i ← 1
while i > 0 do
  run given TM m for given input n for i steps
  if m halts then
    write 0 to the first cell of the tape
    i ← i + 1
  end if
end while
end
```

Figure 5.1: Algorithm for Zeno Machine

After time unit 2 (the upper bound of the execution time required for the above instructions), ZM will have 0 in the first cell of the tape if the given TM m halts on input n, 1 otherwise. Even though it was an abstract model, there are some works based on it in literature which are also physical. They called *relativistic models*, due to using principles regarding general relativity in physics. We will be introducing them on further sections.

Objections

ZM was controversial. There is some sort of objections making ZM controversial in literature, based on flaws regarding its logical correctness and also physical realizability.

As a logical objection, assume that there is a ZM, which constantly adds independent symbols on every next cell in tape only in one direction, regardless of content of the tape. According to the proposal of ZM, it must be stopped after some amount of time, by completing infinitely (countably) many operations, however such a machine in the scenario above, would not complete its task. Since the concept of time is rare, it is questionable in the end of a certain time point if the machine would stop, or caught in the middle of a loop. Hence the one who is deciding the halting problem is not the machine itself, but the meta-entity, observer etc.

For the physical realization, there are objections to the relativistic realization [24] which is explained in the above section. A popular argument is that since the operation speed of ZM grows exponentially, on the length of given input, it might require that speed which is faster than Speed of Light in a conflict with Special Relativity. On the other hand, even if it would be possible, it would require infinite amount of energy which is an another physical problem.

5.2 Infinite Time Turing Machines

As silicon world introduces us to much faster products of computing devices without any permanent bound of speed every new day, it is quite natural for theoreticians to ask a rather philosophical question: what could we compute if we had an infinitely fast computer? One answer to this question due to mathematics in the last decade, opened a new research topic, which is called *infinite time Turing machines*, and also created an abstract model of hypercomputation.[1] In this section, we will follow the main article [15].

Preliminaries

The idea is simply to extend the finitary behavior of ordinary Turing machines in terms of transfinite ordinal running time, therefore the machine may perform an infinite number (defined by the taken transfinite ordinal) of computation steps to achieve its task, or what we might call as supertask in many cases. We will present the machine which is explained in [15] in terms of our general machine concept.

First, let us explain briefly how the machine works, then we will give the formal definition. The infinite time Turing machine or $ITTM$ has finitely many states just like an ordinary Turing machine has (indicated by Q), however it is extended with a special state called the limit state, q_{lim} which denotes that the $ITTM$ is in the *limit stage*. By the word *stage*, we mean the moment or step that the computation is currently in. For instance 0^{th} stage denotes the time point that the $ITTM$ is initialized but not started to execution yet and 1^{st} stage corresponds to the moment when the first applicable element of the transition function is executed etc. Then by *limit stage* we mean a stage α where α is a transfinite ordinal. We will use the word 'machine' sometimes when it is obvious from the context that we talk about $ITTM$.

Just like an ordinary semi-finite Turing machine, $ITTM$ has a tape head moving back and forth to read and write tape symbols. For the convenience of the results which we will mention, the $ITTM$ we are interested in, will have three dimensional tape in which the names of the dimensions are *input*, *scratch* and *output* respectively, each being like an ordinary Turing machine tape i.e. left ended and infinite to the right. For the sake of simplicity, we will call those three dimensions, as *input tape*, *scratch tape* and *output tape* respectively. Since there is only one head for all three tapes, tape head's position for each tape at any stage of computation will be the same (same natural number denoting its position). We assume that the tape head is able to read or write to all three tapes simultaneously. Let us give the formal definition.

[1] Jeffrey Kidder defined infinite time Turing machines initially in 1989 and worked with Joel David Hamkins, when they both were graduate students.

Definition 36 (Infinite Time Turing Machine)**.** *An Infinite Time Turing machine or ITTM is a machine,* (Q, S, δ, I^m, J_n) *where*

- $Q = \{q_i \mid 0 \leq i \leq n, n \in \mathbb{N}\} \cup \{q_{lim}, q_h\}$ *is the finite set of states with* $q_0 \neq q_{lim} \neq q_h \neq q_0$ *where* q_{lim}, q_h, q_0 *are the limit state, the final state and the starting state, respectively.*

- $S = \{(c, i) \mid (c, i) \in \left(\Gamma^{\mathbb{N}}\right)^3 \times \mathbb{N}\}$ *is the storage where* Γ *is the tape alphabet,* i *is the position of the head and* $c = (c_1, c_2, c_3)$ *is the current content of the storage. Then,* $c(j) \in \Gamma^3$ *is* $(c_1(j), c_2(j), c_3(j))$ *where* $c_t(j)$ *denotes the current content of* j^{th} *cell of the* t^{th} *tape for* $t \in \{1, 2, 3\}$*. The content of the storage at stage* α *for ordinal* α *is denoted by* $c^\alpha = (c_1^\alpha, c_2^\alpha, c_3^\alpha)$*.*

- $\delta : Q \times S \to Q \times S$ *is the transition function is a partial function where any* $d \in \delta$ *is in one of the following forms:*

  - $(q_k, (c, i), q_l, (c, i{+}1))$ *is interpreted as* moving right *where* $q_l \neq q_{lim}$*,*
  - $(q_k, (c, i), q_l, (c, i \dot{-} 1))$ *is interpreted as* moving left *where* $q_k \neq q_{lim}$*,*[2]
  - $(q_k, (c, i), q_l, (c', j))$ *is interpreted as* writing *where the following conditions must hold:*
    - *if* $q_k \neq q_{lim}$ *and* $q_l \neq q_{lim}$ *then* $i{=}j$*, and* $c \neq c'$ *implies* $c_t(i) \neq c'_t(i)$*.*
    - *if* q_k (q_l) *is* q_{lim}*, then* i (j) *is* 0 *and* c (c') *is determined by the composition of each* c_t (c'_t) *where* $t \in \{1, 2, 3\}$*:*
      - $c_t(x) = c_t^\alpha(x)$ *for a successor ordinal* α*, if* $c_t^\alpha(x) = c_t^\beta(x)$ *for all* $\beta > \alpha$*,* $x \in \mathbb{N}$ *(analogous for* $c'_t(x)$*).*
      - $c_t(x) = c_t^\beta(x) = 1 = \sup\{c_t^\alpha(x) \mid \alpha < \beta\}$ *where* β *is a limit ordinal, if there are infinitely many* $\gamma < \beta$ *such that* $c_t^\alpha(x) \neq c_t^{\alpha+\gamma}(x)$ *holds. (similarly for* c'_t*).*

- $I^m : \mathbb{R}^m \to S$ *is the input function:*
  $I^m(x_1 \dots x_m) = (c, 0)$ *where* $\forall_{k=0}^{m-1} \forall_{j=1}^{\infty} c_1(m \cdot j - (m - 1 + k)) = x_{k+1_{(j)}}$ *for* $x_{1 \dots m} \in 2^\omega$ *is the binary infinite string representation of the corresponding real and* $x_{i_{(n)}}$ *is the* n^{th} *digit of binary string* x_i*.*

- $J_n : S \to \mathbb{R}^n$ *is the output function:*
  $J_n\left((c_1, c_2, \langle 0 x_{1_{(1)}} x_{2_{(1)}} \dots x_{n_{(1)}} x_{1_{(2)}} x_{2_{(2)}} \dots x_{n_{(2)}} \dots x_{1_{(j)}} \dots x_{n_{(j)}} \dots \rangle), i)\right)$
  $= (x_1 \dots x_n)$ *where* $x_{n_{(j)}} \in \Gamma$*,* $j \in \mathbb{N}^+$*.*

[2] Note that $0 \dot{-} 1 = 0$.

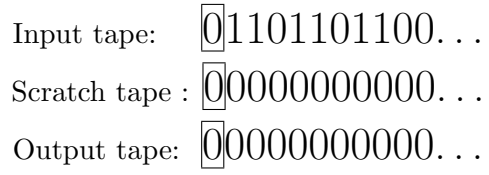


Figure 5.2: Infinite time Turing machine with the tape head on the first cell.

Our tape alphabet Γ will be $\{0, 1\}$.We assume all tapes are filled with zero in the beginning. One should notice from the definitions of input-output functions that the $ITTM$ computable functions are defined on the Cantor space (2^ω) and therefore on 'reals', $\mathbb{R}$. This stands as a direct consequence of machines ability to manipulate infinitely long strings as input, specially when the machine is in special state q_{lim}. One might observe from the definition of input and output functions such that the tape (input/output) convention is based on placing infinitely long binary strings on interleaving cells. Finitely represented numbers, or numbers belonging to a proper subset of $\mathbb{R}$ in general e.g. natural numbers can be represented by padding infinitely many zeros after the number's unary representation, in parallel to a slight modification of the original input output functions. In particular one might restrict himself to a proper subset.[3] It should be noted that we have reserved the very first cell apart from the input/output, in favor of using this cell as a flag in limit *stages*. One can surely increase the number of reserved flag cells, up to the desired purpose.

The machine is initially set to the *start* state which is q_0 where the tape head is set to the *first* cell (the leftmost cell). In the diagram below, it is shown that the tape head of the machine is on the left most cell.

As the definition of transition function says, the limit state can neither be the starting state nor the halting state and in the limit state, the head is plugged off from where it currently is, and placed to the top of the first cell and the content c of the tape t is changed according to the following rules:

- If the content of the cell x converges (eventually stabilizes to 1 or 0), then the value of the cell remains the same in the limit stage,($c_t(x)$ in the definition).

- Else it alternates unboundedly often, then the limit cell value is set to 1.

[3] For natural numbers, one can modify the input/output functions as follows: $I^m : \mathbb{N}^m \to S$ is the input function: $I^m(x_1 \dots x_m) \to (c, 0)$ where $\forall_{k=0}^{m-1} \forall_{j=1}^{x_{k+1}} c_1(m \cdot j - (m - 1 + k)) = 1$. $J_n : S \to \mathbb{N}^n$ is the output function: $J_n\left((c_1, c_2, 01^{x_1}01^{x_2} \dots 01^{x_n}0^\omega), i)\right) \to (x_1 \dots x_n)$ Note that output function is in ordinary Turing machine convention e.g. having a zero between each arguments.

The latter is equivalent to taking the *limit superior* of the values of the cell. By this, as we reach limit ordinal stage β, we can also express the successor steps $\beta + 1$, $\beta + 2$ following β, or even a new limit ordinal stage $\beta + \omega$. We had already defined c^α, the content of the storage at stage α for limit ordinal α. It can also be defined for $q \in Q$ in parallel to c,

- $q^0 = q_0$,
- $q^\alpha = q_i$ for $q_i \in Q$ where α is a successor ordinal,
- $q^\alpha = q_{lim}$ where α is a limit ordinal,
- q^α is undefined if $q^\beta = q_h$ for $\beta < \alpha$.

We may also define it for transition function. Let $q \in Q$ and $s \in S$ and superscript α in $\delta^\alpha(q, s)$ denote α many application of δ or its value at the stage α. We omit the superscript when it is equal to 1. Then

- $\delta^0(q, s) = (q, s)$,
- $\delta^{\alpha+1}(q, s) = \delta(\delta^\alpha(q, s))$,
- $\delta^\beta(q, s) = (q', s')$ if β is a limit ordinal where $q' = q^\beta = q_{lim}$ and $s' = s^\beta = (c', 0)$,
- $\delta^\alpha(q, s)$ is undefined if $\delta^\beta(q, s) = (q_h, s')$ for $\beta < \alpha$.

Above, c' is in turn equals $c^\beta = (c_1^\beta, c_2^\beta, c_3^\beta)$, denoting the content of all three tapes at the stage β which is determined according to the rules mentioned above and also transition function part (writing case) in the machine definition.

We say that a computation process halts on configuration (q, s) if there exists an ordinal α (or $\mathrm{Ord}(\alpha)$) such that $\delta^\alpha(q, s) = (q', s')$ where $q' = q_h$. If at any stage, machine *halts* (i.e. entering the halting state), the output of the computation is the value of the output function on input of what was currently written on the output tape, when the machine is in state q_h. From now on, we restrict ourselves on single argument on both input/output functions, for the sake of *brevity*.

Definition 37. *The computing time of a machine $M = (Q, S, \delta, I, J)$ on input x is*

$$t_M(x) = min\{\alpha \mid Ord(\alpha) \wedge \exists s.\delta^\alpha(q_0, I(x)) = (q_h, s)\}$$

where s is the final content of the storage, since q_h is the halting state.

The domain of (t_M) is defined as follows

$$D(t_M) = \{x \mid x \in \mathbb{R}, \exists \alpha, s : (Ord(\alpha) \wedge \delta^\alpha(q_0, I(x)) = (q_h, s)\}$$

We take q^β and s^β as undefined whenever $t_M(x) < \beta$. Therefore, every $ITTM$ which halts on input x defines a function $ITTM$-computable function.

Definition 38. *Let $M = (Q, S, \delta, I, J)$, the input/output function $Func_M$ of M is*

$$Func_M(x) = J \circ \mathcal{I}_2^2 \delta^{t_M(x)}(q_0, I(x)).$$

The domain of $Func_M$ is defined as,

$$D(Func_M) = D(t_M).$$

It is noteworthy to state that we are assuming every $ITTM$ is represented by a unique natural number, moreover every natural number represents an $ITTM$.[4] Let us give some general definitions.

Definition 39. *Let M be an $ITTM$. We say that;*

- *A partial function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is $ITTM$-computable if there is an M such that $f = Func_M$.*
- *A set of reals A is infinite time decidable if its characteristic function χ_A is $ITTM$-computable.*
- *The set A is infinite time semi-decidable if $D(Func_M) = A$ for an M.*
- *A set is A is α-decidable if its characteristic function $\chi_A = Func_M^\alpha$, where $Func_M^\alpha(x) = J \circ \mathcal{I}_2^2 \delta^{t_M(x)}(q_0, I(x))$ if $t_M(x) \leq \alpha$, undefined otherwise.*

The first three definitions are in parallel to the ordinary Turing machine case. However the last one help us to classify computable sets regarding the length of the computations required, in terms of computation steps. For instance, restricting ourselves to finite input and time, the class of functions $f : 2^{<\omega} \longrightarrow 2^{<\omega}$ would be ω-computable whenever it is computable by an ordinary Turing machine.

Now we will show that focusing ourselves in only countable infinite time computations will be enough.

Definition 40 (Snapshot)**.** *Let $M = (Q, S, \delta, I, J)$ be an infinite time Turing machine, α be an ordinal. Than a* snapshot *of M at stage α is the tuple $\langle \delta, x, \delta^\alpha(q_0, I(x)) \rangle$.*

Note that snapshot contains all the information regarding, transition function (program)[5], input, and tape contents and head position (the value of $\delta^\alpha(q_0, I(x))$) at stage α for that input and transition function.

Theorem 9. *Every halting infinite time computation is countable.*

[4] Giving a code to each machine is the same as in the case of ordinary Turing machines.

[5] Note that what we mean by a $ITTM$ program is just an ordered transition function.

Proof. Assume that an $ITTM$ with transition function δ on input x runs for uncountably many steps without halting. We will show that computation will not halt at all. Take the snapshot of a limit ordinal ω_1 stage of its computation, $\langle \delta, x, \delta^{\omega_1}(q_0, I(x))\rangle$.This information can be coded into a real. Let $y \in \mathbb{R}$ be the real code of this snapshot. We will argue that this snapshot actually must have been repeated earlier at a countable stage, and the computation will continue to repeat itself forever.

As $\delta^{\omega_1}(q_0, I(x)) = (q^{\omega_1}, (c, i))$ where $i = 0$ follows from $q^{\omega_1} = q_{lim}$. If the j^{th} cell of c_k'th tape, $c_k^{\omega_1}(j) = 0$ where $k \in \{1, 2, 3\}$, then there exists a countable stage $\alpha < \omega_1$ in which for all γ such that $\alpha < \gamma < \omega_1$, $c_k^{\gamma} = 0$ holds (by the definition of δ). However if $c_k^{\omega_1}(j) = 1$ then either there exists a countable stage $\beta < \omega_1$ which is similar to the above case for zero or $\sup\{c_k^{\beta}(j) \mid \beta < \omega_1\} = 1$ where $c_k(j)^{<\omega_1}$ alternates unboundedly often. Since the length of the tape is denumerable and $\mathbb{N}$ is *cofinal*[6] in $\mathbb{R}$, we can find a countable stage α_0 by taking a countable supremum such that by the stage α_0, every eventually stabilizing cell is just stabilized and the only cells which keep changing unboundedly often are those which change cofinally often. So there must be sequence of countable ordinals $\alpha_0 < \alpha_1 < \alpha_2 \ldots$ such that all cells which change after α_n, changes at least once by the stage α_{n+1} [7]. Let $\xi = \sup_n \alpha_n$. ξ is a limit stage, therefore $\delta^{\xi}(q_0, I(x)) = (q^{\xi}, (c', i'))$ and $q^{\xi} = q_{lim}$, hence $i' = 0$. Let y' be the real which codes this snapshot. The cells which have stabilized before ω_1, have stabilized before ξ and the cells alternated unboundedly often before ω_1 have alternated unboundedly often. Therefore the snapshots $\langle \delta, x, \delta^{\omega_1}(q_0, I(x))\rangle$ and $\langle \delta, x, \delta^{\xi}(q_0, I(x))\rangle$ are the same, so are y and y'. Thus, the computation has just repeated itself. Limit stages where the snapshots repeats unboundedly often will yield the very same snapshot, thus such computation processes are infinite loops indeed.[8]

□

Power of the Infinite Time Turing Machines

Theorem 10. *Infinite time Turing machines are hypermachines.*

Proof. Consider a well known undecidable problem, namely deciding truth of any first order arithmetic statement. We will argue that it is infinite time

[6]Let A be a set and let $\leq$ be a binary relation on A. Then a subset B of A is said to be cofinal if for every $a \in A$, there exists some $b \in B$ such that $a \in b$. The cofinality $\mathrm{cf}(A)$ of a partially ordered set A is the least of the cardinalities of the cofinal subsets of A. Similarly, the cofinality of an ordinal α is the smallest ordinal β which is the order type of a cofinal subset of α.

[7]Note that α_{n+1} is not necessarily the successor ordinal of α_n

[8]Note that it is possible that a snapshot of a computation might repeat itself twice even though it is not in an infinite loop, and it might halt after ω many steps. However, if the computation is in an infinite loop, snapshot of the two consecutive limit ordinal stage will be exactly the same. In addition, between these two stages none of the cell which were 0 at the first limit will ever switch back to 1 while diverged ones keeps altering back and forth.

decidable. A given arithmetical statement $\forall n\varphi(n,x)$, the machine is able to check the truth of $\varphi(n,x)$[9] for all possible values of $n \in \mathbb{N}$. □

The argument used in the proof of the last theorem states that arithmetical statements are indeed infinite time decidable. However, the power of the Infinite Time Turing machines even exceeds the arithmetical hierarchy.

Theorem 11. *Any Π^1_1 (Σ^1_1) set is $ITTM$-decidable.*

In order to show this result, we will need to introduce some preliminary work with the following well known result.

The relation $\lhd \subseteq A \times A$ where Ord(A) $= \omega$, can be coded by a real $x \in \mathbb{R}$ such that

$$(\langle n,k\rangle)^{th} \text{ bit of } x = \begin{cases} 1 & \text{if } n \lhd k \text{ holds,} \\ 0 & \text{otherwise.} \end{cases}$$

where $\langle .,. \rangle$ is a bijective pairing function, $n, k \in A$, and WO be the set of reals such that real $x \in WO$ if x corresponds to a well-ordering $\lhd$. We need the following fact which is a well-known result.

Fact 2. *WO is Π^1_1-complete.*

By this fact[10], since any Π^1_1 set is Turing reducible to WO, and any reducing computable function f should also be computable by $ITTM$, all we have to show is that WO is $ITTM$-decidable.

Theorem 12 (Counting-Through theorem)**.** *WO is $ITTM$-decidable.*

Proof. We will describe informally the idea of (hyper) algorithm [15] which for a given real x , decides whether $x \in WO$ or $x \notin WO$. Assume that we are given x. We use the first cell of all three tapes as flag.

1. Check whether x codes a linear ordering (reflexive, transitive, and antisymmetry), e.g. for reflexivity, check for every element a of A whether x says $a \lhd a$ holds. Transitivity and antisymmetry is analogous. Return NO if it fails. [11]

2. Guess/Search for the next least element of the ordering and write it to the scratch tape and replace it, every time a new one is found. Turn the flag cell of the scratch tape to 1 and then 0 whenever we find a new one. Return NO if the flag cell retains 1 (the machine finds a smaller one infinitely many time, therefore there is no least element and x does not code a well order).

[9] That can be coded by a real, using the Goedelization.
[10] See [37] or p. 136 of [18].
[11] NO can be taken as 0 to the output tape and 1 as YES.

3. When the least element is found (which is the one witnessed in the scratch tape), delete every of its occurrence from the input tape. In doing so, turn the flag cell of the input tape to 1 and then 0 every time a limit stage is reached.

4. Check weather the input tape is empty, Return YES, otherwise repeat 2 and 3.

As searching the least element and deleting its occurrences from the input tape will yield a new x which is a subset of original x, this will make us checking its every subset to be well ordered and the input tape becomes empty gradually, which is the intersection of all subset is what we wanted. □

Note that we distinguish limit stages which are limits of limits (compound limits) as both the input and the scratch tape flag cells retains being 1.

That Π^1_1 is infinite-time decidable follows by Fact 2, and the decidability of Σ^1_1 , being the complement set, therefore f also decides Σ^1_1. This proves 12.

Obviously, this is the naive procedure, just to expose the computability result we needed. However, one can get an optimized version. If we just erase one element each round, and search for the next while erasing the previous, then for an order type α, the algorithm would take approximately $\omega + (\omega \times \alpha) + \omega$ many steps. The first ω is for checking linear order, and can also be combined with the first search for minimal element (and thus can be eliminated). Searching and erasing would take $\omega \times \alpha$, which takes ω steps for each individual element of the order. The final ω many steps is what it takes to recognize that we are done. So with this procedure, it would take $(\omega \times \alpha) + \omega$ many steps.

Before proceeding let us define the recursive ordinals as it was defined in [32].

Definition 41. *An ordinal α is a recursive ordinal if there exists a relation R such that:*

1. *R is a well-ordering (of some set of integers);*

2. *R is recursive;*

3. *and the well-ordering given by R is order isomorphic to α.*

By the help of recursive ordinals, we can extend the set of decidable sets further up the analytical hierarchy.

Definition 42. *Let β be a recursive ordinal coded by some recursive relation E on ω. Then A is said to be $\beta - \Pi^1_1$ whenever for each $\alpha \leq \beta$ there is a set A_α with $A_\beta = \emptyset$ such that,*

$$\{(k, x) \mid x \in A_{|k|}\} \in \Pi^1_1 \tag{5.1}$$

$|k|$ *denotes the order type of* k *w.r.t.* E *and*

$$A = \left\{ x \mid \exists \alpha < \beta \left(odd(\alpha) \wedge (x \in \bigcap_{\gamma<\alpha} A_\gamma \setminus A_\alpha) \right) \right\} \tag{5.2}$$

where $odd(\alpha)$ *if and only if* α *is of the form* $\alpha+2n+1$ *and* $even(\alpha)$ *if* $\neg odd(\alpha)$*, for* α *is a limit ordinal or zero and* $n \in \mathbb{N}$*.*

That class of sets extends beyond Π^1_1 obviously, however one can still extend it by allowing more complicated relations of E.

Definition 43. *We say that* $x \in \mathbb{R}$ *is writable if there exists an ITTM such that* $\delta^\alpha(q_0, I(0)) = (q_h, s)$ *where* α *is the length of the computation, and* s *is of the form* $((c_1, c_2, x), i)$*.*

Informally speaking, we expect a real x to be writable if there exists an infinite time Turing machine with input 0, halts with x is written on the output tape.

Definition 44. *An ordinal* α *is writable if there is a writable real which codes it.*

Theorem 13. *if* β *is a writable ordinal, then every* $\beta - \Pi^1_1$ *set is decidable.*

Proof. Assume that A is $\beta - \Pi^1_1$ where β is writable ordinal. Consider an algorithm such that, first fills up some part of the scratch tape with E encoding β, which is used in turn as an input for an Π^1_1 algorithm (recall theorem 3 which says Π^1_1 is decidable) in order to constructs the set $\{(n, x) \mid x \in A_{|n|}\}$ (by making a list of the numbers n with the property that the input x is in $A_{|n|}$. Finally, by counting through (using the same fashion in Theorem 4) the relation coding β, the algorithm searches for an odd ordinal $\alpha < \beta$ such that $x \in \bigcap_{\gamma<\alpha} A_\gamma \setminus A_\alpha$.

□

In the sequel, we will try to put a limit on the complexity of $ITTM$-decidable sets. Recall that a snapshot of an infinite time Turing machine is complete description (transition function, input , all current tape contents and the head of the position) of its computation, on the corresponding stage, which is determined by machines transition function. We will call that a transfinite snapshot sequence *accords* with the transition function δ if every successive snapshot is obtained by the computation according to the previous snapshot, and the limit snapshots are obtained from the earlier ones. We say that a snapshot according the transition function is *settled* if the last snapshot belongs to a halting state or repeats and earlier snapshot.

Theorem 14. *G be the graph of a ITTM-computable function, then* $G \in \Delta^1_2$*. Therefore, every decidable set and semi-decidable set is* Δ^1_2*.*

Proof. Suppose $f(x) = y$ be a function where $f = Func_M$ for some M. Let $\vdash$ be a well-ordering relation defined over snapshots where the following (possibly transfinite) sequent

$$\langle \delta, x, \delta^{\alpha}(q_0, I(x)) \rangle \vdash \langle \delta, x, \delta^{\alpha+1}(q_0, I(x)) \rangle \ldots \vdash \langle \delta, x, \delta^{\beta}(q_0, I(x)) \rangle, \tag{5.3}$$

corresponds to the computation process of $Func_M(x)$ then there exists a sequence of corresponding reals

$$r_{\alpha} < r_{\alpha+1} \ldots < r_{\beta}, \tag{5.4}$$

coding the snapshots, $(\mathbb{S}, \vdash)$ being order-isomorphic to $(R, <)$ where $R \subseteq \mathbb{R}$ and $\mathbb{S}$ is the set of snapshots.(Recall that every halting computation is countable.) Now observe that $Func_M(x) = y$ iff there is a real z coding the well-ordered real sequence according to M on input x with output y. Therefore by $\exists z \forall x \exists y Func_M(x) = y$, graph of f is Σ^1_2. Recall that every supertask computation either halts or repeats itself in countably many steps (which is $r_{\alpha} = r_{\beta}$ for some α). Therefore we only need to consider settled sequences such that $Func_M(x) = y$ iff every real z' coding a well-ordered real sequence according to M on input x shows that output as y, which is $\forall z' \exists y \forall x Func_M(x) = y$ in turn means graph of f is also Π^1_2, implies that Δ^1_2. □

CHAPTER 6

Physical Hypermachines

In this section, we will deal with two (possible) physical models of hypercomputation. The first model will be based on *relativity theory*. We will investigate few scenarios, which will help us to understand "how the *gravitational force* can be used to manipulate time to let us to compute more than normally we can". And the second model, namely the quantum circuits, will be taken as a candidate for weak hypercomputation, as it might (possibly) help us to solve an intractable problem in polynomial time.

6.1 Relativistic Models

Introduction

Recall that the theory of Infinite Time Turing machines that we mentioned in previous section, was of no concern regarding the practical usability. In this section, we will mention some approaches which can be regarded as *physical* counterpart of Infinite Time Turing machines.

Notice that the understanding of the concept of time which classical computability theory or complexity theory has, is nothing counterintuitive to our daily understanding of *time*. Furthermore the understanding of *simultaneous* and *infinite* (amount of time) in those fields is also intuitive. What we are interested in such fields is not the unit of time which we may choose and eventually may or may not correspond to the real time measurement units (i.e. *minute* or *year*) but rather an understanding regarding *time* dealing only with *what comes first*, *what comes next* and *what takes longer* (by means of the implicit time unit). Therefore Church-Turing thesis implies this sort of understanding of *time*, however *time* (as well as *space*) is *relative* and can be manipulated by *gravity* according to the theory of *General Relativity* of Einstein, a widely accepted mathematical theory of physics which is

also confirmed by various experimental evidence. Therefore the question of *Is hypercomputation physically possible?* is naturally regarded as a question of physics, hence the answer should consider the related accepted physical theories as granted.

The approaches which we will mention in this section, are *thought experiments* [1] sharing at least one common point; all of them use *time dilation* implied by *relativity theories (special and general)*, in order to dilate the *time* big enough at a particular point of space, relative to another point, so to say to *infinite* such that carrying out a *supertask* becomes possible at one point, according to the other point. Thus we call them *relativistic models*. We refer the reader to [16] for basic knowledge on the theory of General Relativity.

Performing Supertasks in Spacetimes

Itamar Pitowsky was the first who sets up thought experiments (in [29]) to show how we can perform supertasks in spacetime. Let us introduce his scenario.

Let us assume that we have no restriction on space, and that our only concern is of time. Suppose that a mathematician, *Alice* wants to give an answer to *Whether or not, π's decimal expansion has seven consecutive 7s?*, so-called Wittgenstein's problem. Obviously answering such a mathematical problem might require a supertask to perform. [2] While Alice leaves her students to calculate the decimal expansion of π on earth, she gets in a technologically highly advanced satellite orbiting around earth, such that it has the capacity of staying in a fixed orbit while boosting its tangential velocity $V(t) = c(1 - e^{-2t})^{1/2}$ where t is the Earth's time scale, c is the speed of light. This is a set-up regarding special relativity. We calculate the *time dilation* in special relativity by the formula,

$$\Delta t' = \gamma \Delta t \tag{6.1}$$

where Δ' (proper time) is the time interval between two events happening at the same place, measured by the inertially moving observer, Δt is the measured time interval of the other observer and $\gamma = 1/\sqrt{1 - \nu/c^2}$ is called the *Lorentz factor* where ν is the relative velocity of the moving observer to the other, c is the speed of light. If we substitute ν by $V(t)$ in the Lorentz factor and name it as τ, then τ is the satellite's local time scale, and then the infinitesimal time interval $d\tau$ would be $e^{-t}dt$. From this,

$$d\tau = e^{-t}dt \tag{6.2}$$

[1] Since we are not technologically advanced enough to check the given scenarios experimentally.

[2] If somehow we were lucky that we observe such a string of seven consecutive 7 without performing a supertask, we can simply replace it by a question regarding observing a string which includes *seventyseven* consecutive 7 which might convince one requiring a supertask.

we can take integration on both side, we get

$$\tau = \int_0^\infty e^{-t} dt = 1 \tag{6.3}$$

which means that one second passing in satellites time scale does correspond to eternal time in earth's time scale. Therefore Alice has enough time to learn the answer.[3] As her students remained on earth to calculate π and send her the answer by signaling (photon), if they encounter seven consecutive 7. [4] Therefore, Alice have to wait just a second once the satellite reaches the required speed, and if she gets no signal she will stop the satellite and detach herself knowing that answer is no, or she will get a signal in one second. Note that Pitowsky sets up two things nicely in the scenario such that first $V(t) < c$ which is appropriate that nothing is faster than the speed of light up to general relativity and second the integration of the formula gives you a simple result.

Pitowsky thought that his story was impossible because such an operation might need an infinite amount of space. Even though *Einstein's Field Equations* do not imply that the material universe must be finite, most physicists believe that it must be finite. [5] However that is not true for the supertask we have presented, since it is known that it is possible to compute just the nth digit of many transcendentals in almost linear time and logarithmic space (see [6]). [6]

We might have some objections to Pitowsky's argument from *real world*, as one of them might be a natural astrophysical argument such that earth last not forever as even stars wouldn't. Not this one, but two objections against Pitowsky's argument has been raised by John Earman in [10]. First, since the acceleration of the satellite is enormous (simply by taking the derivation of the $V(t)$ up to dt, which is $e^t/\sqrt{1-e^{-2t}}$) such that any human being who would be exposed to an enormous gravitational force would be crushed. Therefore it is possible that Alice might never know the answer. And before mentioning the second, let us give the formal definition Earman gave in [11] for what we call *Pitowsky Spacetime*.

Definition 45. *A pair (M, g), where M is a connected four-dimensional Hausdorff C^∞ manifold and g is a Lorentz metric, is a Pitowsky spacetime if there are future-directed timelike half-curves γ_1, $\gamma_2 \in M$ such that $\int_{\gamma_1} d\tau = \infty, \int_{\gamma_2} d\tau < \infty$ and $\gamma_1 \subset I^-(\gamma_2)$.*

[3] It is one second since c is given in km/s.

[4] We may imply simply by her students, her students and their students and their students etc. through the eternity.

[5] Since even the most dense objects which is observed so far; black holes, have finite masses (A super-massive black hole has 10^5 to 10^9 times mass of Sun).

[6] Pitowsky was not wrong in thinking so because his paper dates back to 1990, the argument he had used as supertask was testing of *Fermats Last Theorem* since truth was not known until 1995, a revealing proof was given by Andrew Wiles.

$I^-(\gamma_2)$ is called *chronological past* which is the collection of all past events of γ_2. Half-curve means that there is a certain point of past event where there is no event before that event, along the curve. γ_2 corresponds to Alice and γ_1 corresponds to her students.

The second objection is based on the "fact" that Alice eventually will know the answer. However if the answer is negative, it means she will not get any signal. There is no definite moment point of when she will get the message (according to the definition above, since $\int_{\gamma_2} d\tau < \infty$ does not determine a certain point over the curve), hence the absence of a signal does not necessarily mean for her to believe that the message will not be sent. Therefore she might think that it has not arrived yet. To avoid such objections, David Malament and Mark Hogarth defined an alternative space time structure which differs slightly.

Definition 46. *A pair (M, g), where M is a connected four-dimensional Hausdorff C^∞ manifold and g is a Lorentz metric, is a* Malament-Hogarth spacetime *if there are future-directed timelike half-curve $\gamma_1 \in M$ and a point $p \in M$ such that $\int_{\gamma_1} d\tau = \infty$ and $\gamma_1 \subset I^-(p)$.*

As one may notice that, there is no reference to γ_2 in that definition, therefore no direct reference to our mathematician Alice. However there is a future directed timelike curve γ_2 from a point $q \in I^-(p)$ to p such that $\int_{\gamma_2(q,p)} dt < \infty$, where q can be interpreted as the event which the signal is sent (therefore q lies in the casual future of the past endpoint of γ_1), and p is the point where the receiver must get the signal. Hence fictional Alice would know that she has to have the signal at p, if it has been sent already, therefore she would know that no signal means the answer (of the supertask problem) is negative.

In [11], Earman notes that , this setting is satisfactory to *effectively decide* on membership of a recursively enumerable but non-recursive set of integers. The reason why Malament-Hogarth spacetime allows supertask lies under the following technical result. We refer reader to [16]. As we will not define *global hyperbolicity* formally, we will not enter the full technical details of the result, thus we take it as a granted fact. It might be useful however to give an intuition such that one might keep in mind in a simple sense as a spacetime being *globally hyperbolic* means each event is timelike related. In a *not globally hyperbolic* spacetime, this feature does not necessarily hold, therefore events may not related to each other by cause and effect.

Fact 3. *A Malament-Hogarth spacetime is not globally hyperbolic.*

As we do not give the original proof, we refer reader to [11]. However, one simple argument is that if a spacetime (M, g) is globally hyperbolic and p, q are timelike related, then there exists a longest timelike curve connecting them. However by definition, there is a p such that $\gamma_1 \subset I^-(p)$ and $\int_{\gamma_1} d\tau = \infty$,

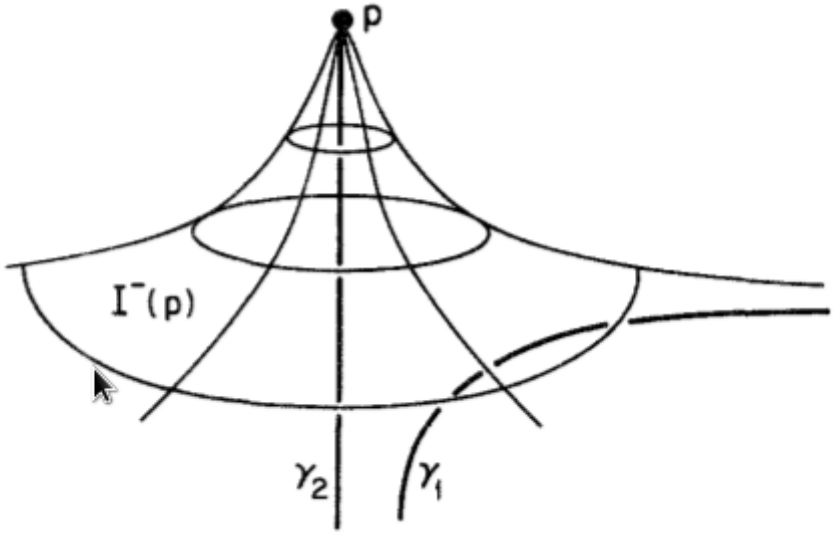


Figure 6.1: An example of demonstration for Malament-Hogarth spacetime.

$\int_{\gamma_2(q,p)} dt < \infty$ can be longer than any finite candidate chosen between q to p, which would yield a contradiction.

Now let us give a *hypercomputer* construction, that Shagrir and Pitowsky presented in [27]. Let the machine H be a pair T_a and T_b of two communicating computers via Morse-like alphabet signaled by photons. T_a is allowed to move only over the curve γ_1 and T_b' is placed initially on the future directed timelike curve which goes from the point q over γ_1 to p. It takes, T_a to go along all γ_1 infinite time, while T_b goes from q to p in a finite time, say, 1 day. H works as follows. Computer T_b is a universal computing machine such that computes the Turing computable function $f(n)$ is defined as what nth Turing machine outputs as it halts on input n, undefined otherwise. Now $h(n)$ be the halting function defined as 0 if $f(n)$ is undefined, 1 otherwise. Obviously, H computes h, as T_b'sall infinite path travel along γ_1 while computing $f(n)$ will be under T_a 's past once it reaches p (i.e. $\gamma_1 \subset I^-(p)$).

Naturally there are objections against the feasibility of the system. First T_b requires infinite memory, which in turn makes H physically impossible. One response is there are space-times (e.g. Reissner-Nordstroem) which allows spatial infinity [16], which is an instance of Malament-Hogarth spacetime. A second response is Turing machines themselves are using infinite tape. Therefore, excluding infinite storage is a misleading argument against computing halting function by means of classical computability theory. Another rather philosophical objection was on the nature of computation such that a T_b component of H as never halting, can not be classified as a machine *computing* computational task, which also can be applied to Infinite-Time Turing ma-

chines. However that is not serious since there is a system component T_a) which allows to decide the halting function, (or limit state in Infinite-Time Turing machines). A rather meaningful objection is against H's being a computing device in the sense that, in such a case one can not stop T_b working on $f(n)$ being undefined and thus the system or H is not re -usable. This behaviour might exhibit a point of view that a particular machine H might not be regarded as in strength of computing h, however it still computes something that an ordinary Turing machine can not. Which is our original definition of *hypermachine* is based on.

Black hole computation

Black holes as being one of the most mysterious yet the most popular object of popular science books, perhaps it is not a big surprise that we encounter those exotic objects in the field of hypercomputation. In this section we will give a brief understanding of the nature of black holes as it will become easy for one to imagine thought experiments allowing hypercomputation eventually. Then we will discuss some alternatives on using black holes gravitational power on computing incomputable functions. As generally well known, black holes are the most dense objects with huge masses which curves the spacetime around them in extreme, such that even *light* the fastest traveling object in universe (according to Einstein's relativity) can not escape from their enormous gravitational power. Therefore we do not see them and name them as Black holes. A natural question might be *how much velocity an object might need to totally escape from another objects gravitational force?*. The answer is the formula which determines *escape velocity*[7] :

$$\nu_e = \sqrt{\frac{2GM}{r}} \tag{6.4}$$

where G is the universal gravitational constant, M is the mass of the body which is the center of the gravity and r is the distance from the center of the gravity. Black holes are of no exception, by the formula we wrote above we can derive[8] a formula which tells us what is the radius (independent of mass) that a body needs to that even *light* cannot escape:

$$r = \frac{2Gm}{c^2} \tag{6.5}$$

This formula tells us what should be the radius of a body of given mass, such that technically it turns out to be a black hole. One may find it by the formula such that one has to squeeze it to a radius of $3km$ for a body which is equal

[7] This applies to spherically symmetric objects and derives by the idea of the kinetic energy should be equal to gravitational force (Newtonian) in order to escape: $\frac{1}{2}m(\nu_e)^2 = \frac{GMm}{r}$.

[8] By pulling r.

to Sun's mass and $9 \times 10^{-3}m$ for the Earth to get a black hole. [9] This is the intuition behind how a matter should be dense in order to become a black hole and bend the spacetime in a way that it exhibits absurd nature to our understanding.

The gravitational force that black holes exhibit are nothing special but just a result of the general law(s) that Einstein's theories imply. Roughly stating the basic idea: mass tells spacetime how to curve and spacetime tells mass (matter) how to move.[10] The mathematical toy that allow us to make accurate conclusions and meaningful assertions about the nature and geometry of the space and time (in big scale), is nothing but a particular solution of Einstein's Field Equations [16], which is called metric (due to the convention and naming is an element of the fields *topology* and *differential geometry*). Here we give a solution which is called the *Schwarzschild metric*; a general solution for outer part[11] of any symmetrically distributed spherical mass, admitting the simplest (non-rotating, non-charged) black hole model, namely the *Schwarzschild black hole*. The Schwarzschild metric is given as follows,

$$d\tau^2 = \left(1 - \frac{r_s}{r}\right)\frac{dt^2}{c^2} - \left(1 - \frac{r_s}{r}\right)^{-1}\frac{dr^2}{c^2} - \frac{r^2(d\theta^2 + \sin^2\theta d\varphi^2)}{c^2} \tag{6.6}$$

where τ is the proper time (the time measured by the moving observer) $r_s = \frac{2GM}{c^2}$ is called the *Schwarzschild radius* with M is the mass of (any) black hole (that we calculated previously as r), where, c is the speed of light in meters per second, t is the time coordinate in seconds according to our reference of frame (assumed to be stationary at infinity), r is the radial coordinate in meters (the distance from the black hole's mass for positive values), θ is the *colatitude* in radians, and φ is the *longitude* in radians.

We will only make a simple case analysis over the Schwarzschild metric, which in turn will tell us more about the behavior of a black hole or namely a *Schwarzschild black hole*. First, the term $(d\theta^2 + \sin^2\theta d\varphi^2)$ is the metric for unit sphere, which is used to deal with the angular position of the observer. It will not be of our concern. For the sake of simplicity, let us name it Ω and furthermore take $c = 1$. The whole metric will look like as follows.

$$d\tau^2 = \left(1 - \frac{r_s}{r}\right)dt^2 - \left(1 - \frac{r_s}{r}\right)^{-1}dr^2 - r^2(d^2\Omega) \tag{6.7}$$

The term $\left(\frac{1-r_s}{r}\right)dt^2$ stands for *time* (since it stands for the change in t; consists of dt^2) where as $\left(1 - \frac{r_s}{r}\right)^{-1}dr^2$ is its *space* analog. Notice that they have opposite signatures.[12] So whenever $d\tau^2$ becomes *positive*, we say that what our

[9] Values taken are for Sun's mass= $1.9891 \times 10^{30}kg$ and Earth's mass= $5.9722 \times 10^{24}kg$ according to Nasa's website: http://nssdc.gsfc.nasa.gov/planetary/factsheet/.

[10] This quote is due to the American physicist John Archibald Wheele.

[11] By outer part, we mean the region which is left beyond the surface.

[12] Note that this is due to the *spacetime signature*. For details see [16] and [19].

metric measures (spacetime interval[13]) is *timelike* and if *negative*, we say that it is *spacelike*. Importance of this concept lies behind knowing that one can communicate only in timelike intervals (due to the speed of light limitation), which means change in distance in any time unit should be smaller than the distance measured by the light travel per the same time unit (e.g. a *light year* which is the distance a light photon takes in one year).
Let us make case analysis for critical cases. One of them is $r > r_s$ in turn means $r_s = 2GM \to r$ (r_s/r goes to 0). Two cases is possible:

- either $M \to 0$ which means our black hole has no mass (i.e. completely *evaporated* or disappeared) therefore no gravitational effect[14]

- or $r \to \infty$ which means that the object is infinitely far (whatever that means) from the mass of the black hole, such that again no gravitational force applies.

The other critical case is when $r < r_s$ where object is inside the event horizon which causes $(1 - r_s/r) < 0$, therefore dr term becomes positive, and dt term becomes negative, which means the interval is totally spacelike, and hence one cannot communicate or travel, this brings us to basic principle of black holes which is one crosses the event horizon cannot get out. An object which falls through the event horizon ends up its journey in *singularity* where theoretically speaking, density becomes infinite (end of time). Another and the final critical case is that $r \to r_s$ (from up), which means that the object gets close to the event horizon such that $r_s/r \to 1$ has two consequences:

- dr term gets big,

- dt term gets small, which is interpreted as time dilation.

Therefore we say that all intervals become gradually spacelike, and ends up with as there was no timelike interval anymore such that one cannot communicate or travel (as in previous case). Just as a matter of interpretation, it is said that space and time replaces one another. This is called *space and time reversal*.

[13]Spacetime interval is the measure of distance between two events, in terms of time and space.
[14]The metric becomes the one for flat like.

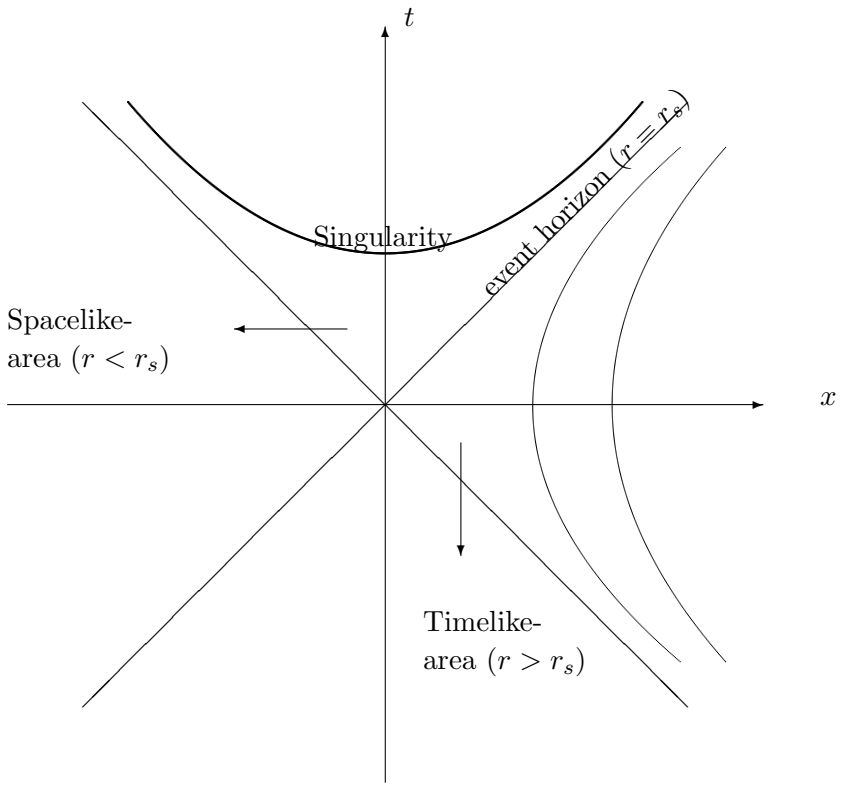


Figure 6.2: The observer and the computer on Schwarzschild blackhole, according to Kruskal coordinate system.

The figure above is an abstraction of *Kruskal coordinate system* for the Schwarzschild black hole, where so-called hyperbolae represents time-like curves, so an object which stands still in space moves along the hyperbola in upward direction (which is the direction of time *future-directed*). x-axis is r-like however it does not correspond to r itself. This is due to such Kruskal coordinate system has its own metric, however one can prove that the Schwarzschild metric can be reduced to the Kruskal metric, where the Schwarzschild metric is represented in terms of the Kruskal metric (see [16]). Therefore one can say that the geometric behavior is similar. The event horizon is denoted by the inclined straight line which has the angle 45° degree with the $x-$axis. This corresponds *symbolically* to the speed of light (according to the origin; the intersection of x and t). One might notice that with its lower counterpart (45° in reverse direction from the x-axis) covers the region which is called the *known past*.

The analysis of the Schwarzschild metric regarding the (Schwarzschild) black hole, might already have created some intuition, in reader, of how to build a physical process or set up which allows *hypercomputation*, (at least) in terms of a thought experiment. Consider the following scenario.

Assume that we have two communicating parties, one which makes the ordinary computation, say, a *computer* initialized to perform a supertask and sends a signal (in forms of photon(s)) to the other party if itself finds an answer[15], and the other, an *observer* (e.g. a human or a photon detector device) which will observe the computer, and finally will come up with the answer *yes* or *no* (in the absence of a signal) to the question regarding supertask. At that point, it must be clear that we are only interested in one-way communication, as *observation* is a matter of photon detection. We assume that the computer, is located in a constant far distance from the black hole [16], thus, the gravitational effect which is exerted over the computer, will be negligible. The observer have to be located somewhere near to (or over) the event horizon in order to exhibit time dilation, necessary enough for the computer to perform the supertask, namely *infinite*. In another words, as the distance between the observer and the event horizon decreases, the clock (hypothetical) of the observer will run slower (compared to the one, which is of the computer), hence we look for somewhere, which makes the clock of the observer totally *frozen* relative to the clock of the computer. And there is exactly the point $r = 2GM$ which is over the event horizon (recall dt term becomes zero and hence the time is relatively frozen). Following is the illustration of the described scenario.

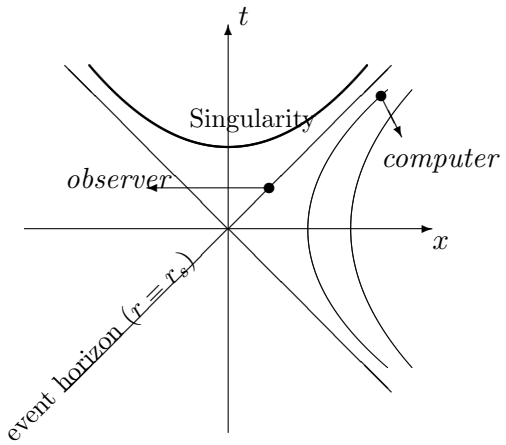


Figure 6.3: The scenario which the observer is over the event horizon.

[15] This regards to the answer which is classically computable part.

[16] As solar system orbits around the super-massive black hole which is the center of *Milky Way Galaxy*.

There are two problems of feasibility one has to deal with this approach of Schwarzschild black hole. First the enormous tidal forces emerge as one reaches the event horizon, would crush any being or device we could build today i.e. spaceship which the observer would planted in. Orbiting over event horizon is equally impossible even if we presume that we have such an advanced technology to build such spaceships with enough throttle power to hover over the event horizon, since it would require to move faster than light (check [16]) which is forbidden as the *theory of relativity* stipulates. Tidal forces of the event horizon can be avoided as one chooses a sufficiently big black hole, however, the biggest black hole observed so far, is even not big enough to avoid tidal forces as it is needed. One possibility to overcome this issue, is to use a small scale detection mechanism (e.g. a quantum circuit). The second, is the singularity. As the *observer* crosses the event horizon (presumably endured against the tidal forces of event horizon), has no other way but gets dragged into the singularity (the center of the black hole) with the speed greater than or equal to the speed of light. Hence, it is not guaranteed that the *observer* will ever get the message which the *computer* sends, as there is no chance to survive (by means of not to lose its *identity*) once it reaches to the singularity.

To achieve a thought experiment which allows hypercomputation without such consequences, as it is employed in another black hole model in [24], what is called the *Kerr black hole*.[17] that any accelerating observer cannot keep its position static no matter how much acceleration it has, such that it moves along the rotating axis. As it should be clear from the context that, this black hole model comes as a result of another spacetime which is defined; *Kerr spacetime*, which in turn means another *exact* solution of Einstein's field equations; the *Kerr solution*.

We will explain why this model works, by introducing the different character of Kerr black hole. However, we will not mention the *Kerr metric* formally, since the very idea is just a variation of Schwarzschild metric along with the depth of the extra technical details are irrelevant to our main focus which is hypercomputation. We refer the interested reader to [24].

The Kerr black hole is a rotating black hole. Considering most of the observed celestial bodies rotate, astrophysicists believe that the Kerr solution reflects the reality more likely, compared to the Schwarzschild solution. As a consequence of its rotation, the Kerr black hole has two event horizon instead of one (as it was in the Schwarzschild black hole), which are the *outer event horizon* and the *inner event horizon*. The outer event horizon just behaves as the event horizon of the Schwarzschild black hole, i.e. it flips the time to the space. Besides, due to the black holes rotation, it also denotes the boundary that any accelerating observer cannot keep its position *static* no matter how much acceleration it has, such that it moves along the rotating axis. However,

[17]Named after New Zealander mathematician and astronomer Roy Kerr.

the inner event horizon flips space to time back, such that one becomes able to move again in a timelike curve, through the singularity. The singularity is also different in the Kerr black hole, which forms no more of a point shape, but rather a ring shape. This is a subtle point in choosing the Kerr black hole for realizing a supertask, such that observer might be able to avoid the singularity by determining its route and angle of its fall (since it will be in a time like spacetime geometry inside the inner event horizon).[18] This creates a counter argument against the *inevitable final crush of the falling observer by the singularity in the Schwarzschild black hole* This also implies that the observer to get the signal sent from the computer before it hits singularity.

The region remains between the inner and the outer event horizons, is called the *ergosphere*. Quite expectedly, one cannot leave ergosphere by crossing the outer event horizon back. The circumference of the outer event horizon and inner event horizon decreases as the angular momentum of rotation increases. That is the reason that a black hole has been chosen to be massive and slowly rotating in [24], so to speak, to save the falling observer from enormous tidal forces. The following figure is taken by [24], is a schematic representation of the falling of the observer in the Kerr black hole.

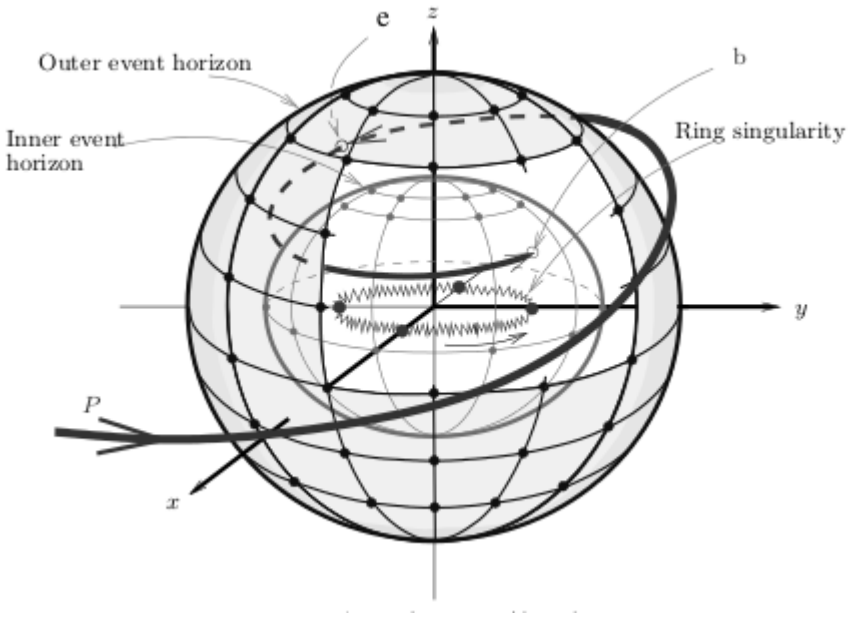


Figure 6.4: The route of an observer falling into a Kerr black hole.

In the three dimensional diagram above, z is the axis of rotation, P is the free falling observer, e and b are the points where observer enters the outer

[18]Technical details are in [24].

and the inner event horizons of the black hole, respectively.

6.2 A possible physical case for Weak Hypercomputation

In this section, we will push a well known computational model, so called (classical) quantum computation, as a serious challenge to Weak Church Turing thesis.

Shor's Order-Finding Algorithm

In this section, we will consider a problem which is yet believed to be intractable *classically*; however a tractable solution exists in quantum computation literature. The order-finding problem may be defined as the following:

Definition 47 (order-finding problem)**.** *Suppose that a and N are positive integers, with $a < N$ and having no common prime factors. The least positive integer r satisfying $a^r = 1(modN)$, is said to be the 'order' of 'a modulo N', and the order-finding problem is to find the order for a given a and N.*

It is believed that there is no algorithm solving the order-finding problem which is polynomial in the size of problem specification $O(n)$, where n is the number of bits needed to encode N completely.

It is well-known that quantum computers are indeed quantum circuits and it is also known that for any chosen Turing machine, there exists a quantum computer which it corresponds [9].

To show that todays quantum computers are indeed relevant candidates for weak hypercompuation, we will show that the order-finding problem (a likely intractable problem) can be solved (polynomially) using *phase estimation* (See Appendix: Quantum Computing) combined with some extra arrangements. Therefore, the circuit is a weak-hypercomputer if the order-finding problem is classically intractable. The key idea is to apply *phase estimation* to the following unitary operation.

$$U_{a,N} |x\rangle = |ax\rangle \, (modN) \tag{6.8}$$

where $x \in \{0,1\}^n$.[19]

What might be the eigenstate of $U_{a,N} |x\rangle$? [20] For sure there can be in many forms; however, considering the following one might give us a little insight on finding the order r using *phase-estimation*:

[19]Note that without considering the case $N \leq x < 2^n$ our definition of the operation would be inconvenient, so in that case, for simplicity $U_{a,N} |x\rangle$ will be taken as x.

[20]Recall that $|\phi\rangle$ is an eigenstate for a unitary operator U if $U |\phi\rangle = v |\phi\rangle$.

$$|\psi_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi ijk/r} \left|a^k mod N\right\rangle \tag{6.9}$$

We can show that $|\psi_j\rangle$ where $0 \leq j < r$ are eigenstates of the operator $U_{a,N}$ as follows [21],

$$\begin{aligned}
U_{a,N}|\psi_j\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi ijk/r} \left|a^k a \ mod N\right\rangle && (6.10)\\
&= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi ijk/r} \left|a^{k+1} mod N\right\rangle && (6.11)\\
&= \frac{1}{\sqrt{r}} \left(e^0 \left|a^1 mod N\right\rangle + \ldots + e^{-2\pi ij(r-1)/r} \underbrace{|a^r mod N\rangle}_{1} \right) && (6.12)\\
&= \frac{e^{2\pi ij/r}}{\sqrt{r}} \left(e^{-2\pi ij/r} \left|a^1 mod N\right\rangle + \ldots + e^{-2\pi ijr/r} |1\rangle \right) && (6.13)\\
&= e^{2\pi ij/r} |\psi_j\rangle && (6.14)
\end{aligned}$$

In order to be able to apply phase-estimation, we need both to find a way to prepare an eigenstate in form of $|\psi_j\rangle$ which will correspond to an eigenstate in the phase estimation circuit, and to implement $U_{a,N}$ which will correspond to the controlled gate. For the former, note that such an eigenstate will be enough, since it doesn't change even if we had to apply the phase estimation procedure several times. Recall that our main aim was to find the order r, and it looks like we need to know r first to prepare the state $|\psi_j\rangle$, unfortunately. Here is a very tricky observation which allows us to proceed,

$$\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |\psi_j\rangle = |1\rangle \,. \tag{6.15}$$

Why is that so?(Exercise 5.13 of [25]) It actually follows from the basic inner fact which is

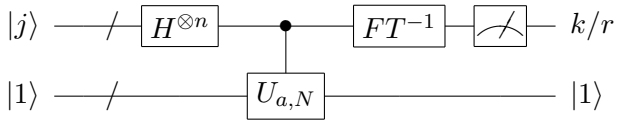
$$\sum_{j=0}^{r-1} e^{-2\pi ijk/r} = \begin{cases} 0, & \text{if } k \neq 0 \\ r, & \text{otherwise} \end{cases} \tag{6.16}$$

Therefore it becomes,

[21] Following derivation is just a complete extraction of the shortcut statement 5.38-39 in [25].

$$\begin{aligned}
\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}|\psi_j\rangle &= \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{-2\pi ijk/r}\left|a^k mod N\right\rangle \\
&= \frac{1}{r}\sum_{j=0}^{r-1}\sum_{k=0}^{r-1}e^{-2\pi ijk/r}\left|a^k\ mod N\right\rangle \qquad (6.17)\\
&= \frac{1}{r}r\left|a^0\ mod N\right\rangle = |1\rangle
\end{aligned}$$

Hence, this - running phase estimation on $|1\rangle$- gives us the ability to run the phase estimation (assuming we have $U_{a,n}$) as if we had used any randomly selected $|\psi_j\rangle$ from $\{|\psi_0\rangle \ldots |\psi_{r-1}\rangle\}$ in it. So that with the measurement, the output we will get in the first register of our circuit will be k/r which corresponds to the estimated phase.



(6.18)

Before coming up with the answer regarding the question of *How can we pull out r from k/r?*, let us deal with another problem which has higher priority which is: to implement $U_{a,N}$.
We would like to have such a transform as below:

$$\begin{aligned}
|z\rangle|x\rangle &\rightarrow |z\rangle|a^z x(mod N)\rangle \\
&= |z\rangle\left|a^{z_1 2^0}\times\ldots\times a^{z_n 2^{n-1}}x(mod N)\right\rangle \\
&= |z\rangle U_{a_1,N}^{2^0}\ldots U_{a_n,N}^{2^{n-1}}|x\rangle \qquad (6.19)\\
&= |z\rangle U_{a,N}|x\rangle
\end{aligned}$$

(6.20)

where $|z\rangle$ is the state for the first-register bits (n-many), and controlled $U_{a_j,N}^{2^{j-1}}$ is the gate corresponds to j-th bit of $|z\rangle$. As it can be seen, $U_{a,N}$ might be formulated as taking the *modular exponentiation* of the content of the second register. This operation can be done by a quantum circuit with $O(n^3)$ in the size of resource n. Please see the appendix (quantum computation) for further details.
Let us turn back to the problem of getting r from k/r. We know that it is an approximation and if we were able to compute the nearest rational number, by reducing it to a irreducible extent, we might get r. Hopefully, there is an algorithm which does the job, so-called the *continued fraction algorithm.*

The Continued Fraction Algorithm

We will explain the algorithm informally. In order to write $x \in \mathbb{R}$ in continued fraction form we first split it in the form of

$$x = a_0 + y_0 \tag{6.21}$$

where $a_0 \in \mathbb{Z}^+$ and $y_0 \in [0, 1)$ and then inverting y_0 part as a fraction (if $y_0 \neq 0$), which in turn becomes $x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{y_1}}$ where $a_1 + y_1 = \cfrac{1}{y_0}$. So we repeat the same process for every new non-zero[22] y we get (which is the sequence $y_1 \dots y_n$ for n many operations) and get x which is in the form of:

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{\vdots}{a_{n-1} + \cfrac{1}{y_n}}}} \tag{6.22}$$

where $a_{n-1} \in \mathbb{Z}^+$, $y_n \in (0, 1]$. If this is the supremum, then it means y_{n+1} is zero, so we stop and y_n becomes our a_n. In brief, what we need to get the supremum of the continued fraction for $x \in \mathbb{R}$ is the following two rules:

i) $x = a_0 + y_0$,

ii) $y_i = \cfrac{1}{a_{i+1} + y_{i+1}}$ where $a_0 \dots a_{i+1} \in \mathbb{Z}^+$ and $y_0 \dots y_{i+1} \in [0, 1)$.

The finite sequence $[a_0, \dots, a_n]$ [23] obtained by using the procedure explained above is called n^{th} *convergent* of the continued fraction. Associating the real number x to the n^{th} convergent (for desired possible n), is called *continued fraction algorithm*. About the complexity of the process; real $x = k/r$ as a fraction of k and r, each being n-bits integer, it would take $O(n)$ steps to *split and invert*, each requiring of $O(n^2)$ many gates to carry out the arithmetic operations, would make $O(n^3)$ in total.
Regarding the

Quantum Order Finding Procedure

1. $|0\rangle\,|1\rangle$

[22] We stop as our new y becomes 0, not surprisingly.

[23] It is trivial to show it since (numerator in y_i) > (numerator in y_{i+1}) which is strictly decreasing.

2. $\Longrightarrow \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle |1\rangle$ (creating superposition: O(n))

3. $\Longrightarrow \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \left|a^j mod N\right\rangle \approx \frac{1}{\sqrt{r2^n}} \sum_{j=0}^{r-1} \sum_{k=0}^{2^n-1} e^{-2\pi ijk/r} |j\rangle |\psi_k\rangle$ (applying $U_{a,N} : O(n^3)$)

4. $\Longrightarrow \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \widetilde{|k/r\rangle} |\psi_k\rangle$ (applying inverse transform to the first register: $O(n^2)$)

5. $\Longrightarrow \widetilde{k/r}$ (measuring first register $O(n)$)

6. $\Longrightarrow r$ (applying continued fraction algorithm : $O(n^3)$)

Thus the running complexity of the whole algorithm is polynomial in the size of $O(n)$.

CHAPTER 7

Possible Improvements

In this thesis, we tried to investigate few models of hypercomputation to some degree. While some of the models were completely from the mathematical domain, and some others were purely from theoretical physics. It would be a nice improvement to be able to present all of them in the perspective of the same general framework. Nevertheless, there were two physical models which escaped from that purpose; quantum computers and relativistic models.

- Black hole computation: it would be interesting to represent the interaction of the observer and the computer as the whole computer system. What one has to do (probably) is, to extend the general machine framework via *relative time* concept. The machine has to be defined from the angle of the observer. Assuming that computer sends signals per time unit (according to computers reference of frame), observer can calculate his distance to the event horizon, and the time difference with the computer, using the metric. Note that such working success of such model would be exempted from a particular model of black hole.

It could also be interesting to mention the stronger models of relativistic machines: SAD machines, and to investigate their computational power. It would be good to show, whether they can decide analytical predicates, or if not, why can it not simulate an Infinite Time Turing machine. And to see in which level current physical theories allow us to build such theoretical physical models. In general, another improvement which can be done, would be to investigate interrelation between purely mathematical models and physical models, under computability point of view. This would reveal some possible answers to the question *which models of hypercomputation might be more general than the other?*, or *which model can simulate the other under which circumstances?*

It would be good to mention some applications of the infinite time Turing machines to set theory and higher recursion theory, establishing they are not just toy models, but intuitive tools help us to continue mathematical research.

This text is by no means a complete reference for background information on any of the subject represented, rather is a specific information or brief introduction including necessary information to understand the related parts belonging to the body of main work.

APPENDIX A

Quantum Computation

A quantum computer is a physical system based on quantum mechanics, accepting quantum states as inputs which are to represent a superposition of different possible states and outputs a superposition of another set of states which is evolved subsequently by unitary transformations or so-called computation. Transformations may affect each element of a superposition simultaneously, exposing dramatically efficient computational power on solving some problems which are believed to be intractable in classical computation paradigms [35].

It must be noted that, by the term *quantum computer* we refer to some class of *quantum circuits* or family of quantum circuits built by quantum logic gates.

Before start to explain quantum computers, let us introduce the *Hilbert space* (finite dimensional) the *mathematical universe* where a quantum computer works.

A.1 Qubits

Single Qubit

The *quantum bit* or *qubit* is the fundamental concept where the quantum computation and quantum information theory are based on, is just the quantum mechanical *dual* or counter part for the *classical* concept 'bit' wherein the classical computation and information theory are based upon.

In contrast to its classical counter part which might have either the value 0 or 1, a qubit can be in states $|0\rangle$, $|1\rangle$ or another one; the *continuum* in between which is called *superposition* which can be expressed as a linear combination of states $|0\rangle$ and $|1\rangle$ as follows,

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \tag{A.1}$$

where α_0, α_1 are complex number *amplitudes* for $|0\rangle$ and $|1\rangle$. "$|\ \rangle$" is called the *Dirac notation*. The state of a qubit is a vector in a two-dimensional vector space,

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}, \tag{A.2}$$

and the states $|0\rangle$ and $|1\rangle$ are known as *computational basis states* which form an othonormal basis for this vector space.

Examining qubit and a classical bit is either of a noteworthy difference. In classical computation (or computers) a bit might be determined if it has either 0 or 1; however, in quantum computation we might not be able to determine (or *measure*) qubits state totally, which is when we measure it is 0 with probability $|\alpha_0|^2$ and 1 with probability $|\alpha_1|^2$ that is by probability sum, *normalization condition* (for single qubit) is expressed as $|\alpha_0|^2 + |\alpha_1|^2 = 1$. So it should be noted that the quantum bit is in superposition until it is observed. That means there are some parts of information which is not *accessible* to *measurement* regarding basic *principles of quantum mechanics* which is not the main concern of this text.

Multiple Qubits

In case of n multiple qubits, the number of *computational basis of states* and *amplitudes* increases to 2^n. For example, in case of two qubits, *computational basis states* are $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ (by writing each two digits in single *dirac paranthesis* e.g. $|01\rangle$ instead of $|0\rangle|1\rangle$) Therefore a two qubits system state is expressed as follows,

$$|\Psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}, \tag{A.3}$$

where Ψ is for the two (or multiple) qubit state symbol and $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}$ are the complex number coefficients or *amplitudes*.
In fact the expression $|0\rangle\,|1\rangle$ just refers to $|0\rangle \otimes |1\rangle$; we omit $\otimes$, the *Kronecker product* for simplicity in notation. We next give the definition of *Kronecker product*:

Definition 48 (Kronecker Product)**.** *Let A be an $m \times n$ matrix and B be an $p \times q$ matrix, then Kronecker product of A and B is defined as the $(mp) \times (nq)$ matrix*

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \dots & \dots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}. \tag{A.4}$$

For arbitrarily many qubits system, say n-ary, the state vector can be written as a summation formula,

$$|\Psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle, \tag{A.5}$$

and the *normalization* condition (which is probabilistic sum equal to one) can also be generalized as

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1, \tag{A.6}$$

A.2 Qubit Gates

As the classical computers have circuits which are constructed by logic gates, analogously quantum circuits have quantum logic gates, in order to manipulate information. In this section, we will see some basic ones and their properties.

Single Qubit Gates

As it can be understood by its name, a single qubit gate is a logic gate manipulating information on a single qubit. Let us start with the one that we know from *classical computation*; the single qubit gate NOT. Rather than taking just the opposite of a qubit as $|0\rangle$ to $|1\rangle$ or vice versa, as a classical NOT would behave, quantum NOT gate takes *linearly* inverse of a qubit state:

$$\alpha\,|0\rangle + \beta\,|1\rangle \to \alpha\,|1\rangle + \beta\,|0\rangle \tag{A.7}$$

or the other way around. It is important to point out that, just as the *quantum states*, the *quantum gates* are also represented by matrices. Specially, a single qubit gate can be represented by a 2 by 2 matrix. So here is the matrix representation for NOT:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{A.8}$$

So it does the operation on quantum qubit $\alpha\,|0\rangle + \beta\,|1\rangle$ which is $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ as

$$X \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}. \tag{A.9}$$

The only constraint is that matrix which represents the gate should be *unitary*. We say that U is **unitary** if $UU^\dagger = I$ holds where I is the *identity matrix* and $U^\dagger$ is the adjoint of U (complex conjugated and transposed). The following

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{A.10}$$

is the 2 by 2 identity matrix. Note that X is indeed unitary.

Let us continue with another single qubit gate

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{A.11}$$

which leaves the $|0\rangle$'s as they are, but changes the signs of $|1\rangle$ to $|-1\rangle$. The following matrix

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{A.12}$$

together with X and Z are called *Pauli matrices*. The set of all elements in complex space of 2 by 2 matrices can be represented as a linear combination of Pauli matrices.

Another important quantum gate which will be used is *Hadamard gate* defined as follows

$$H = 1/\sqrt{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{A.13}$$

In quantum circuits, an Hadamard gate is used to transform a qubit basis states $|0\rangle$ and $|1\rangle$ to two superposition states with equal weight of the computational basis states $|0\rangle$ and $|1\rangle$.

$$\alpha\,|0\rangle + \beta\,|1\rangle \;\text{—}\boxed{H}\text{—}\; \alpha\tfrac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\tfrac{|0\rangle-|1\rangle}{\sqrt{2}} \tag{A.14}$$

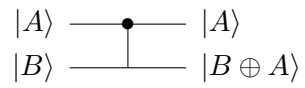
Using its definition one can also confirm that, applying it twice in succession would set the qubit back into its initial state.

$$\alpha\tfrac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\tfrac{|0\rangle-|1\rangle}{\sqrt{2}} \;\text{—}\boxed{H}\text{—}\; \alpha\,|0\rangle + \beta\,|1\rangle \tag{A.15}$$

Multiple Qubit Gates

In addition to single qubit gates, there are some others which are multiple qubit gates, however the ones that we know from classical circuits like NAND, NOR, XOR, OR are not multiple qubit gates. Since their matrix representations are not unitary (because they are not invertible) there are cases where we cannot know, for a given output, which input was originally accepted. This means some information is deleted, and cannot be recovered. The necessity of

preservation of information leads to a very important concept in computation (in general) and a feature of quantum computation which is called *reversibility*. Therefore regarding quantum computation, we have universal computation power [9], besides some extra beneficiary consequences.[1] In general, we say that if a computation can be performed reversibly, it can be done by a quantum circuit which involves only the gates which have a unitary matrix representation. There is such a multiple qubit gate which is called CNOT or *controlled*-NOT gate:



(A.16)

which two input qubits known as the *control* and the *target* qubit (from top to bottom in the diagram). One can describe it as target qubit does not change if the control qubit is set to $|0\rangle$, flipped if it is set to $|1\rangle$. $\oplus$ is XOR; And its matrix representation is U_{Cnot}:

$$U_{Cnot} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{A.17}$$

Here we will give another important concept which is called *universality*. Any finite set of gates are said to be **universal** in the sense that any possible unitary operation can be expressed as a composition[2] of gates from that set. CNOT and single qubit gates are such a set.
Another gate we will need is so-called SWAP gate, which swaps two qubits simply, here we give the matrix representation :

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.18}$$

A.3 Quantum Circuits

Actually, we had already given an example of a quantum circuit, namely the $CNOT$ circuit in the previous section. There are few more things which should be known. First, the circuits are read in a fashion from left to the right, so that inputs are on the left side and outputs are on the right. Second, we had

[1] Landau's principle: no energy dissipation requirement in reversible computations

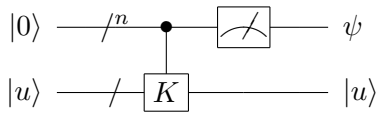
[2] We imply an approximation instead of ideal representation, since the set of all possible gates are uncountable whereas the gate sequence from any finite set can only be countable.

already mentioned the restriction that we do not want any content or combination which causes the circuit to be irreversible. This implies, loops are not allowed, say, feedback interaction between some different parts so that we assume, our circuit should be *acyclic*.

Let us introduce one more quantum circuit element, which is the measurement instrument. We will explain it without going into so much detail. We had already mentioned a little bit of the measuring the states like $\alpha|0\rangle + \beta|1\rangle$, yielding 0 and 1 with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively. There can be superpositioned states such as $\frac{\alpha|0\rangle+\beta|1\rangle}{\sqrt{2}}$ and $\frac{\alpha|0\rangle-\beta|1\rangle}{\sqrt{2}}$ satisfying the normalization condition ($|\alpha|^2 + |\beta|^2 = 1$) so that

$$\alpha|0\rangle + \beta|1\rangle = \frac{(\alpha+\beta)}{\sqrt{2}}\frac{\alpha|0\rangle + \beta|1\rangle}{\sqrt{2}} + \frac{(\alpha-\beta)}{\sqrt{2}}\frac{\alpha|0\rangle - \beta|1\rangle}{\sqrt{2}}, \tag{A.19}$$

in this case the probability of measuring former state with probability $\left|\frac{\alpha+\beta}{2}\right|^2$ and latter with probability $\left|\frac{\alpha-\beta}{2}\right|^2$ is expected. In the example below, we see the measuring instrument symbol in the circuit before the output ψ:



(A.20)

and the symbol $/^n$ expressing that there are n many qubits of $|0\rangle$ and / at the bottom, expresses there are indefinitely many of $|u\rangle$.

A.4 Quantum Fourier Transform

The Fourier transform is an operation widely used in applied mathematics and engineering. In a bright manner, besides its key importance in many quantum algorithms, it also lays behind the well-known *speed-up*s in algorithms on particular *problems*, regarding computational complexity.

A specific type of Fourier transform is the *discrete Fourier transform*, which is extensively used in *Fourier analysis* in mathematics. Basically, it takes a fix sized, say N, complex vector (a sequence of complex numbers), x_0,...,x_{N-1} as input, and outputs another complex vector $y_0, ..., y_{N-1}$ with the same size. The transformed vector (output) is defined as

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i j k}{N}} \tag{A.21}$$

where $x_j \in \mathbb{C}$, i is $\sqrt{-1}$.
The *quantum Fourier transform* is nothing but the same information, including arrangements regarding to quantum specific convention. The n-qubit

fourier transform is defined as follows,

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle \tag{A.22}$$

Quantum Fourier transform can be implemented by quantum gates, which means that the transform is a unitary operation.

In the sequel, we will show quantum circuit design for the transformation. In order to do that, it will be useful to initiate N as 2^n since $|0\rangle, ..., |2^n - 1\rangle$ is the computational basis for n-qubit quantum computer. Also, it will be convenient to adopt the state to be transformed in binary representation i.e. $|j_0, .., j_n\rangle$ instead of $|j\rangle$ By some algebraic derivation from [25], which is shown below,

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle \tag{A.23}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi ij\left(\Sigma_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \dots k_n\rangle \tag{A.24}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi ijk_l 2^{-l}} |k_l\rangle \tag{A.25}$$

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left(\sum_{k_l}^{1} e^{2\pi ijk_l 2^{-l}} |k_l\rangle \right) \tag{A.26}$$

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi ij2^{-l}} |1\rangle \right) \tag{A.27}$$

$$= \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)}{\sqrt{2^n}}$$

we get the *tensor product representation* of the transformation (QFT),

$$QFT|j_1, \dots, j_n\rangle = \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)}{\sqrt{2^n}} \tag{A.28}$$

which is highly useful to understand the idea behind the design of the circuit.

In the following, we give the quantum circuit for quantum Fourier transform, on purpose to clarify the calculation of its computational complexity

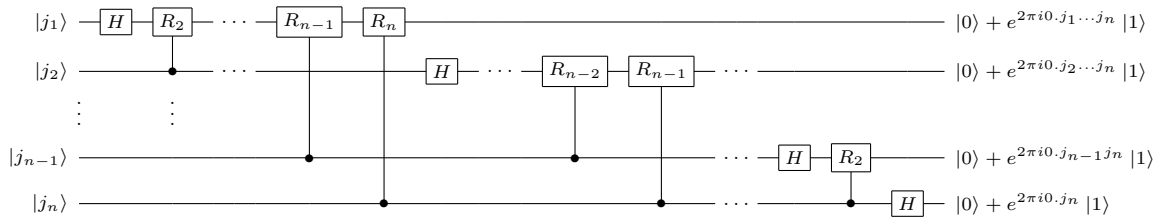


Figure A.1: The quantum circuit which performs Fourier transform.

The figure above, is taken from [25]. It is noteworthy to realize the similarity between the circuit output and the tensor product representation. [3]

Now lets check how the circuit works closely. We apply the Hadamard gate to the qubit to bring it into a uniform superposition state; namely when a Hadamard gate is applied to any input qubit $|j_i\rangle$ with $1 \leq i \leq n$, it takes the form

$$\frac{|0\rangle + e^{2\pi i 0.j_i} |1\rangle}{\sqrt{2}}, \tag{A.29}$$

because $e^{2\pi i 0.j_i}$ equals to -1 when j_i is 1, and 0 otherwise[4] Another instrument which we use in the circuit is the controlled-gate R,

$$R_n \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^n}} \end{pmatrix}, \tag{A.30}$$

which transforms its input from

$$\frac{|0\rangle + e^{2\pi i 0.j_i} |1\rangle}{\sqrt{2}},$$

to

$$\frac{|0\rangle + e^{2\pi i 0.j_i j_{i+1}} |1\rangle}{\sqrt{2}}$$

which is just to add an extra binary fraction bit to the *phase* of the coefficient of $|1\rangle / \sqrt{2}$. So for particular input with multiple R gates, the process becomes as follows

$$\frac{|0\rangle + e^{2\pi i 0.j_1 \cdots j_n} |1\rangle}{\sqrt{2}} \quad \boxed{R_1} \cdots \boxed{R_k} \quad \frac{|0\rangle + e^{2\pi i 0.j_1 \cdots j_n \cdots j_{n+k}} |1\rangle}{\sqrt{2}} \tag{A.31}$$

which looks *generalized* version of any single wire from *main circuit* without the Hadamard gate.

Computational Complexity of Quantum Fourier Transform

It is not hard to evaluate the computational complexity of a quantum circuit. Indeed, it is enough to count the number of operations –gates, which is used.

[3] Note that the swap operators for reverse ordering and normalization factors by $\frac{1}{\sqrt{2}}$, on outputs are omitted for the sake of simplicity.

[4] Recall that j_i are in binary form, so is the fraction $0.j_i$ e.g. $0.j = 1/2$ for $j = 1$.

Now let us take a look at the quantum circuit which is given in previous subsection.

For the first qubit,	a Hadamard gate and $n-1$ R gate:	n
For the second qubit,	a Hadamard gate and $n-2$ R gate:	$n-1$
$\vdots$	$\vdots$	$\vdots$
For the nth qubit,	a Hadamard gate:	1

In that way, we get $n(n+1)/2$ many gates. We would need extra $n/2$ gates at most for swapping n qubits, thus the number of gates throughout the whole circuit would be $n(n+2)/2$ which is asymptotically $O(n^2)$; the computational complexity of the whole transform.

A.5 Quantum Phase Estimation

Phase estimation is an application area for quantum Fourier transform, which is by no means important to *core* computer science but to physics. Nevertheless, it is needed for understanding working principles of some quantum algorithms e.g. *order finding* or *factoring*.

We need to clarify first what is to be understood by the word *phase*, in the intended context.[5] Assuming that we have unitary operator U with the eigenvector $|u\rangle$ and the eigenvalue $e^{i\psi}$, where $0 \leq \psi \leq 2\pi$. ψ is the *phase* here and the goal is to find it (estimation with the best accuracy of $n$bit). [6] We have shown the quantum circuit which operates *phase estimation* in the diagram below.

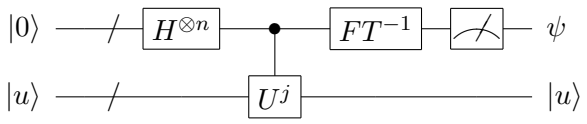


Figure A.2: Quantum circuit for phase estimation.

We use two qubit registers (groups of qubits). First register (n-qubits) is at state $|0\rangle$ as input and will become ψ as output state, so the number n is determined by the need of how many qubits we do require to represent ψ. The second register is for $|u\rangle$. $|u\rangle$ is the control register, so to say, as remains unchanged between input and output states. It can be considered to be completed in three (hypothetical) stages:

[5] That is important for involving non-physics backgrounded researchers into the discussion.

[6] In assumption that we are able to prepare the state $|u\rangle$ and availability of a *black box* being able to carry out operations of controlled-U^{2^j}.

i) applying the Hadamard gate (n-many) to the first register.

ii) applying controlled-U (c-U) operations on the second register

iii) taking the inverse Fourier transform on the first register.

Now lets check the first two stages closer. In doing that, one must clarify the action taking place by applying c-U^{2^j} operations on the second register. When a single such operation on a state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}|u\rangle$ is applied, it does[7]

$$\begin{aligned}\frac{|0\rangle+|1\rangle}{\sqrt{2}}|u\rangle &\xrightarrow{c-U^{2^j}} \frac{|0\rangle+|1\rangle}{\sqrt{2}}U^{2^j}|u\rangle \\ &= \frac{|0\rangle|u\rangle+|1\rangle e^{i2^j\psi}|u\rangle}{\sqrt{2}} \\ &= \frac{|0\rangle+e^{i2^j\psi}|1\rangle}{\sqrt{2}}|u\rangle \end{aligned} \tag{A.32}$$

So the overall output of the circuit would be,

$$\begin{aligned}&\frac{\left(|0\rangle+e^{i(2^{n-1}\psi)}|1\rangle\right)\left(|0\rangle+e^{i(2^{n-2}\psi)}|1\rangle\right)\ldots\left(|0\rangle+e^{i(2^0\psi)}|1\rangle\right)}{\sqrt{2^n}}|u\rangle \\ &\qquad = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}e^{ij\psi}|j\rangle|u\rangle \end{aligned} \tag{A.33}$$

An issue that we have to deal with, is changing the representation of ψ. Specifying $\tilde{\psi}$ in the definition of phase ψ (concerning the design of the circuit in close scale), where the relation is $\psi = 2\pi\tilde{\psi}$ and where $\tilde{\psi}$ provides mod 2π representation in n-bit binary fractions i.e. $\tilde{\psi} = 0.j_1j_2..j_n$.[8] In the sequel, we will use ψ instead of $\tilde{\psi}$ for simplicity reasons. So the modified version of the previous equation is,

$$\begin{aligned}&\frac{\left(|0\rangle+e^{2\pi i(2^{n-1}\psi)}|1\rangle\right)\left(|0\rangle+e^{2\pi i(2^{n-2}\psi)}|1\rangle\right)\ldots\left(|0\rangle+e^{2\pi i(2^0\psi)}|1\rangle\right)}{\sqrt{2^n}}|u\rangle \\ &\qquad = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}e^{2\pi ij\psi}|j\rangle|u\rangle \end{aligned} \tag{A.34}$$

In the figure below, you will see the sub-circuit showing first two stages in more detail,

[7] Recall that $|u\rangle$ is an eigenvector of U.

[8] Interested reader might check further details from [1].

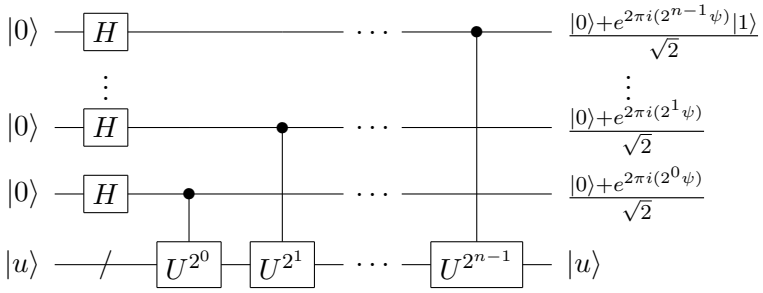


Figure A.3: The first two stage of the quantum phase estimation in detail.

The last stage of the phase estimation is based on taking the inverse quantum Fourier transform and measurement. It is given by,

$$QFT^{-1}\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1} e^{2\pi ij\psi}\,|j\rangle\,|u\rangle = \left|\tilde{\psi}\right\rangle|u\rangle \tag{A.35}$$

The explanation for the above equation is quite simple; one might look out at the original quantum Fourier transform definition and see it is formulated in just reverse direction. One important question which can be raised is *Why is this possible?*. Recall that all the operations we perform in quantum circuits are reversible, so are the ones we use to perform QFT.

In addition, $\left|\tilde{\psi}\right\rangle$ stands for the pre-measured state which is expected to be close to ψ at best accuracy. It should be noted that we have just considered the best case which is, ψ was specified by n bits which is not have to be so always, however regarding the justification for it as being a *good* accuracy will not be our main focus, since as we are interested in computational complexity and the circuit design remains the same but interested reader can find the justification in (p. 156-158) [1] and (p.223 -224) [25]. Recalling the reversibility of the unitary operations, this can be done simply by reversing the quantum circuit for the quantum Fourier transform. So the complexity remains the same which is $O(n^2)$.

APPENDIX B

Computability theory

B.1 Input duplicator Turing machine

$T_d = \{q_0 0 R q_h, q_0 1 R q_1, q_1 1 R q_2, q_1 0 R q_2, q_2 0 1 q_2, q_2 1 R q_3, q_3 0 1 q_3,$
$q_3 1 L q_4, q_4 0 L q_5, q_4 1 L q_4, q_5 1 L q_5, q_5 0 R q_6, q_6 1 0 q_6, q_6 0 R q_7, q_7 1 R q_8,$
$q_7 0 R q_{10}, q_8 0 R q_9, q_8 1 R q_8, q_9 1 R q_9, q_9 0 L q_1, q_{10} 1 R q_{10}, q_{10} 0 L q_h\}$.

T_d simply writes two 1s to right to the reserved area (2 cell after the right most 1 initially) for each 1 it deletes from the left most.

APPENDIX C

Ordinal and Cardinal Arithmetic

We will give basic definitions of ordinals, cardinals and their arithmetics together with some side remarks. We will also give some basic results e.g. theorems sometimes without proof. We start by defining well-ordering.

Definition 49. *a) A total ordering (non-strict) is a pair $\langle A, R\rangle$ where the binary relation R totally orders the set A, if the following holds for all $x, y, z \in A$:*

i. $(xRy \wedge yRz) \implies xRz$ *(transitivity)*

ii. $(xRy \wedge yRx) \implies x = y$ *(antisymmetry)*

iii. $(xRy \vee yRx)$ *(totality)*

b) $\langle A, R\rangle$ is a well-ordering if $\langle A, R\rangle$ is a total ordering and every nonempty subset of A has an R-least element.

Definition 50. *A set x is transitive if and only if every element of x is a subset of x.*

Example 1. *$\{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}\}$ is a transitive set whereas $\{\{b\}\}$ is not.*

Definition 51. *If X is a set, $\mathcal{P}(X) = \{Y \mid Y \subseteq X\}$ is said to be power set of X and $\mathcal{P}$ is said to be power set operator.*[1].

[1]Note that in ZF, for every set there is a power set, which is defined by power set axiom

C.1 Ordinals

Now we will define ordinals (the members of the class *Ord*) and introduce some of their basic properties.

Definition 52. *x is an ordinal if and only if x is transitive and well-ordered by $\in$*[2]

Definition 53. *$<$ is a total ordering of the class OrdLet α and β be any two ordinals, $\alpha < \beta$ iff $\alpha \in \beta$.*

We use finite ordinals to denote natural numbers. As we use natural numbers to count finite sets, assuming that AC [3] to be true, ordinals can be used to count every set. We will use symbol $<$ to denote strict and $\leq$ to denote the total ordering relation in the sequel.

Definition 54. *Let α be an ordinal and S be the successor operation, then $S(\alpha) = \alpha \cup \{\alpha\}$.*

In axiomatic set theory (e.g. in ZF),*everything* is defined using ordinal numbers, even natural numbers.

$$
\begin{aligned}
&0 := \emptyset,\\
&1 := S(0) = \{0\} \cup 0 = \{0\} = \{\emptyset\}\\
&2 := S(1) = \{1\} \cup 1 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}\\
&3 := S(2) = \{2\} \cup 2 = \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}\\
&\vdots\\
&\omega := \{0, 1, 2, 3, \ldots\}.
\end{aligned}
$$

Lemma 5. *For any ordinal α and β, $\alpha < S(\alpha)$ and $\beta < S(\alpha)$ iff $\beta \leq \alpha$.*

Proof. Take any ordinal α, $\alpha \in \alpha \cup \{\alpha\}$ implies $\alpha < \alpha \cup \{\alpha\}$, which implies $\alpha < S(\alpha)$ in turn (by Definition 54).
Take any ordinals α and β. $\beta < S(\alpha)$ iff $\beta \in \alpha \cup \{\alpha\}$ iff β can be at most α iff $\beta \leq \alpha$.

□

Definition 55. *α is a successor ordinal iff there is an ordinal β such that $\alpha = S(\beta)$. α is a limit ordinal iff $\alpha \neq 0$ and α is not a successor ordinal. In addition $\alpha = sup\{\beta \mid \beta < \alpha\}$*[4] *if α is a limit ordinal.*

From this definition it should be understood that, β has no predecessor. So if an ordinal is known to have no predecessor, than it is either a limit ordinal

[2]Note that $\in$ is membership relation.

[3]AC for Axiom of Choice is known to be equivalent of Well-Ordering theorem which states: Every set can be well-ordered.

[4]**Sup** is least upper bound (supremum).

or 0.

Note that above it is already seen that the ordinal which corresponds to the set of natural numbers, is denoted by ω. It is the least limit ordinal, since any ordinal $\alpha < \omega$ is a finite ordinal and thus a successor ordinal. Now we define the ordinal arithmetics.

Addition

Definition 56. *For any ordinals $\alpha, \beta, \gamma, \xi$,*

i. $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$.

ii. $\alpha + 0 = \alpha$.

iii. $\alpha + 1 = S(\alpha)$.

iv. $\alpha + S(\beta) = S(\alpha + \beta)$.

v. $\alpha + \beta = \sup\{\alpha + \xi \mid \xi < \beta\}$, *if β is a limit ordinal.*

Recall that ordinals are well-ordered sets, so their arithmetic respects well-ordering. However some of the properties that we have in standard addition (addition of natural numbers) are no longer valid. Commutativity is one of them. Consider, $\omega + 1$:
$\omega + 1$ is a successor ordinal ($S(\omega) = \omega + 1$),

$$\omega + 1 \neq \omega = 1 + \omega$$

by 56.*iii* and 56.*v* respectively

Multiplication

Definition 57. *For any ordinal $\alpha, \beta, \gamma, \xi$,*

i. $\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$.

ii. $\alpha \cdot 0 = 0$.

iii. $\alpha \cdot 1 = \alpha$.

iv. $\alpha \cdot S(\beta) = \alpha \cdot \beta + \alpha$.

v. $\alpha \cdot \beta = \sup\{\alpha \cdot \xi \mid \xi < \beta\}$ *if β is a limit ordinal.*

vi. $\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$

Again some properties we got used to do not hold for multiplication. One of them is commutativity. Consider $\omega \cdot 2 = \omega + \omega$ which can be read as 'two times ω' if it helps. On the other hand $2 \cdot \omega$ is 'ω times two' which is $2 + 2 + 2 \ldots = \omega$. Thus $\omega \cdot 2 \neq 2 \cdot \omega$.

Exponentiation

Definition 58. *For any ordinal α, successor ordinal β and limit ordinal ζ*

i. $\alpha^0 = 1$.

ii. $\alpha^\beta = \alpha^{\beta-1} \cdot \alpha$.

iii. $\alpha^\zeta = \sup\{\alpha^\beta \mid \beta < \zeta\}$.

C.2 Cardinals

Cardinal numbers are used to determine the size of a set. Normally, the cardinality of a set is said to be the number of its elements. One may denote the cardinality of a set A by $|A|$ or $card(A)$. One uses 1-1 functions to compare the cardinalities of two sets. In particular, $|A| \leq |B|$ if there exists a 1-1 function from A into B.

Definition 59. *Cardinality of a set A, denoted by $|A|$ is defined as*

$$|A| = min\{\alpha | \alpha \text{ is an ordinal and } A \sim_b \alpha\}$$

where $A \sim_b B$ if there exist a bijection from A to B.

In that sense, we can say that cardinality is the equivalence class under bijection. $|A| = |B|$ holds if $|A| \leq |B|$ and $|B| \leq |A|$ hold (Cantor-Bernstein Theorem). In parallel, $|A| < |B|$ is defined as $|A| \leq |B|$ and $|A| \neq |B|$.

We call the cardinals which are used to denote the size of finite sets as *finite cardinals*, and infinite sets as *infinite cardinals* or *transfinite cardinals*. We will assume that AC holds.[5] So it implies that every natural number is a cardinal number (finite cardinal), and ω is a cardinal. However not every ordinal is a cardinal as it is expected so, for example $\omega + 1$ is not a cardinal, because cardinality of such a set is ω i.e. recall that $\omega + 1 \neq \omega$ for ordinal ω. In brief, there exists no set whose cardinality is $\omega + 1$. Let us give the formal definition of a cardinal number.

Definition 60. *An ordinal α is a cardinal iff α=$|\alpha|$.*

We can express cardinality of a set which is equal to the above definition as: For every well ordered set A,

$$|A| = min\{\alpha \mid |A| = |\alpha|\}.$$

This is the reason behind how the *cardinal arithmetic* will be different then ordinal arithmetic.

[5] So the cardinals we will be talking about, are the cardinals in ZFC.

Every transfinite cardinal is a limit ordinal. Transfinite cardinals are denoted by $\aleph$'s (alephs). The least transfinite cardinal is $\aleph_0$ (corresponds to ordinal ω) which is the cardinality of $\mathbb{N}$ (natural numbers) (also $\mathbb{Z}$ -integers also $\mathbb{Q}$ -rational numbers). $\aleph_1$ which is a higher cardinality is the cardinality of $\mathbb{R}$ -real numbers or *continuum*. A set A is *countable* if $|A| < \aleph_0$ or $|A| = \aleph_0$, and is called *uncountable* if it is not countable. $\aleph_1$ is the least uncountable cardinal.

Cardinal Arithmetic

Definition 61. *Let A and B be two sets, then*

i) $|A| + |B| = |A \times \{0\} \cup B \times \{1\}|$.

ii) $A \cdot B = |A \times B|$,

iii) $A^B = \{f : f \text{ is a function } \wedge D(f) = B \wedge ran(f) \subset A\}$[6].

One can show that $+$ and $\cdot$ is associative, commutative and distributive. Consider that taking $A \times \{0\} \cup B \times \{1\}$ instead of $A \cup B$ in i), is just a trick in order to make two sets disjoint if they are not, obviously this does not effect the result if they were already disjoint.

Recall that A^B is the set of all functions from B to A. So iii) is just straightforward compared to ordinal exponentiation. Consider ordinal exponentiation of $2^\omega = \sup\{2^\xi \mid \xi < \omega\} = \bigcup_{\xi<\omega} 2^\xi = \omega$.[7] However for cardinal numbers 2 and ω, 2^ω is just itself (by definition) which is bigger than ω by means of cardinals.

Lemma 6. *For every set A, $|\mathcal{P}(A)| = 2^{|A|}$.*

Proof. We have to show that there exists a bijective function between $\mathcal{P}(A)$ and 2^A. For every $B \subset A$, let χ_B be defined as

$$\chi_B(x) = \begin{cases} 0 & \text{if } x \in B, \\ 1 & \text{if } x \in \bar{B}. \end{cases}$$

Then take $f : \mathcal{P}(A) \longrightarrow \{0,1\}^A$ defined as $f(B) = \chi_B$ for each such B. So every subset of A is assigned to a binary string in length of $|A|$. f is 1-1 and onto. □

Theorem 15 (Cantor)**.** *Let X be any set, then $|X| < |\mathcal{P}(X)|$.*

[6] In axiomatic set theory, A^B exists since, $A^B \subset \mathcal{P}(A \times B)$ which exists by the Power Set Axiom (i.e. $\forall x \exists y \forall (z \subset x \rightarrow z \in y)$) [22].

[7] 2^ω would not be an well-ordered set if it would be defined as set of mappings from ω to 2: Take f_n to be an infinite string n many 0's followed by infinite 1's, where f_0 is the maximal element. This set has no minimal element, however is a subset of the set $\{f : f$ is a function from ω to $\{0,1\}\}$, implies superset is not well-ordered too.

Proof. (By Contradiction). It is enough to show that given any set X, no surjective function f from X to $\mathcal{P}(X)$ be defined, which means there is at least one subset of X which is not an element of the image of X under f. Assume that there is such an f. Consider the following set:

$$Y = \{x \in X \mid x \notin f(x)\}.$$

Obviously, Y is a subset of X, now we have two possibilities for any x:

i) $x \in f(x)$; means $x \notin Y$, therefore $f(x) \neq Y$.

ii) $x \notin f(x)$; means $x \in Y$, hence $f(x) \neq Y$.

It follows that for any x, $f(x) \neq Y$ means f is not surjective. Contradiction. □

The technique which has been used above is called *diagonalization*. Now we will give some properties of cardinal arithmetic without proving them.[8] We will use κ, λ and μ to range over cardinals.

1. $0^\kappa = 0$ if $\kappa > 0$.
2. $\kappa^0 = 1, 1^\kappa = 1$.
3. $(\kappa \cdot \lambda)^\mu = \kappa^\mu \cdot \lambda^\mu$.
4. $\kappa^{\lambda+\mu} = \kappa^\lambda \cdot \kappa^\mu$.
5. $\left(\kappa^\lambda\right)^\mu = \kappa^{\lambda \cdot \mu}$.
6. If $\kappa \leq \lambda$, then $\kappa^\mu \leq \lambda^\mu$.
7. If $0 < \lambda \leq \mu$, then $\kappa^\lambda \leq \kappa^\mu$.
8. If κ and λ are transfinite cardinals then $\kappa + \lambda = \kappa \cdot \lambda = \max\{\kappa, \lambda\}$

Lemma 7. *For every cardinal α and set A of cardinals,*

i) There exists a cardinal β such that $\alpha < \beta$.

ii) $\sup A$ *is a cardinal.*

i). Follows from Cantor's theorem and Power set Axiom. *ii).* Let $\kappa = \sup A$. Let us assume that there is a bijective f to $\beta < \kappa$. Let us take α such that $\beta < \alpha \leq \kappa$, then $|\alpha| = |\{f(\xi) : \xi < \alpha\}| \leq |\beta|$. Contradiction. □

From the above result, we can give an increasing enumeration of transfinite cardinal numbers indexed by ordinals. We will use it to define *successor cardinals* and *limit cardinals* as last of this section.

[8] Some of their proofs are absolutely not trivial.

- $\aleph_0 = \omega_0 = \omega$.
- $\aleph_{\alpha+1} = \omega_{\alpha+1}$.
- $\aleph_\alpha = \omega_\alpha = \sup\{\omega_\gamma \mid \gamma < \alpha\}$ if α is a limit ordinal.

ω's are said to be *order type*.

Definition 62. *A cardinal $\aleph_\alpha$ is a successor cardinal iff α is a successor ordinal (α is in the form of $\alpha' + 1$ for an ordinal α') and an limit cardinal iff α is a limit ordinal.*

We can just mention, the famous historical statement regarding the cardinalities in brief. Cantor asked if there was an infinite cardinality between $\aleph_0$ and $2^{\aleph_0}$. As he couldn't settle the existence of such infinite cardinality, after long times of searching, he conjectured that $2^{\aleph_0}$, which was known to be the cardinality of reals, is the next cardinality after $\aleph_0$, hence it is called as $\aleph_1$. He could neither prove nor refute it. This famous assertion is called *continuum hypothesis* (CH). It can be stated as follows.

Definition 63 (Continuum Hypothesis)**.** $\aleph_1 = 2^{\aleph_0}$.

As it conjectures, the next larger cardinality of set of natural numbers is the one emerged whilst one applies the power set operation.This corresponds to cardinality of reals. One might generalize this statement by asserting it for all infinite cardinalities, and gets *generalized continuum hypothesis* (GCH).

Definition 64 (Generalized Continuum Hypothesis)**.** $\aleph_{\alpha+1} = 2^{\aleph_\alpha}$.

It has been shown after by Cohen and Gödel that CH is independent from ZFC.

Bibliography

[1] Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of Quantum Computation And Information: Basic Tools And Special Topics.* World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[2] Alonzo Church. A Note on the Entscheidungsproblem. *J. Symb. Log.*, 1(1):40–41, 1936.

[3] S. Barry Cooper. *Computability Theory.* Chapman and Hall/CRC, 2003.

[4] B. Jack Copeland and Diane Proudfoot. What Turing did after he invented the universal Turing machine. *Logic, Language and Information*, 9(4):491–509, 2000.

[5] S. Coskey and J. D. Hamkins. Infinite time Turing machines and an application to the hierarchy of equivalence relations on the reals. *ArXiv e-prints*, January 2011.

[6] Simon Plouffe David Bailey, Peter Borwein. On the rapid computation of various polylogarithmic constants. *Math. Comp.*, 66:903–913, 1997.

[7] Martin Davis. *Computability and Unsolvability.* Mcgraw-Hill, 1958.

[8] Vinay Deolalikar, Joel David Hamkins, and Ralf Schindler. P $\neq$ NP $\cap$ co-NP for Infinite Time Turing Machines. *J. Log. and Comput.*, 15:577–592, October 2005.

[9] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Royal Society (London), Proceedings, Series A - Mathematical and Physical Sciences*, 400:97–117, 1985.

[10] J. Earman and J. Norton. Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes, 1993.

[11] John Earman. *Bangs, Crunches, Whimpers, and Shrieks: Singularities and Acasualities in Relativistic Spacetimes.* Oxford University Press, 1995.

[12] E. Fredkin and T. Toffoli. Conservative Logic. *International Journey of Theoretical Physics*, 21:219–253, 1982.

[13] R. Gandy. Church's thesis and principles for mechanisms. *In J. Barwise, H. J. Keisler and K. Kunen (Eds.), The Kleene symposium. Amsterdam: North-Holland.*, 2000.

[14] Richard C. Jeffrey George S. Boolos, John P. Burgress. *Computability and Logic*. Cambridge University Press, March 4, 2002.

[15] A. Hamkins J. D., Lewis. Infinite Time Turing Machines. *J. Symbolic Logic*, 65:567–604, 2000.

[16] Ellis G. F. R. Hawking, Stephen W. *The Large Scale Structure of Space-Time*. Cambridge University Press, 1975.

[17] Jürgen Buntrock Heiner Marxen. Attacking the Busy Beaver 5. *Bulletin of the EATCS*, 40, 1990.

[18] P. G. Hinman. *Recursion-Theoretic Hierarchies*. Springer-Verlag, 1978.

[19] Efstathiou G. P. Lasenby A. N. Hobson, M. P. *General Relativity: An Introduction for Physicists*. Cambridge University Press, 2006.

[20] Stephen Cole Kleene. Introduction to metamathematics. 1952.

[21] G. Kreisel. A notion of mechanistic theory. *Synthese*, 29:1116, 1974.

[22] Kenneth Kunen. *Set Theory: An Introduction to Independence Proofs*. North Holland, 1983.

[23] Alexander Leitsch, Guenter Schachner, and Karl Svozil. How to acknowledge hypercomputation? *Complex Systems Publications*, 18:131–143, 2008.

[24] Etesi G. Nemeti, I. Non-turing computations via Malament-Hogarth space-times. *Int. J. Theor. Phys.*, 41:341–370, 2002.

[25] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quanum Information*. Cambridge U.P., 2000.

[26] P. Odiffreddi. *Classical Recursion Theory, Theory of Functions and Sets of Natural Numbers*. North Holland, 1992.

[27] Itamar Pitowsky Oron Shagrir. Physical hypercomputation and the ChurchTuring thesis, 2003.

[28] Roger Penrose. *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford University Press, 1994.

[29] Itamar Pitowsky. Physical Church Thesis and Physical Computational Complexity. *Iyun*, 39:81–99, 1990.

[30] Richards I. Pour-El, M.B. The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics*, 39:215–239, 1981.

[31] Tibor Rado. On Non-Computable Fuctions. *Bell System Technical Journal*, 41(4):877–884, 1962.

[32] Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. The MIT Press, 1987.

[33] R. Rosen. Effective processes and natural law. in the universal Turing machine. a half-century survey. *R. Herken, Ed. Kammerer and Unverzagt, Hamburg*, page 523, 1988.

[34] C.P. Schnorr. *Rekursive Funktionen und ihre Komplexität*. Teubner Studienbücher, 1974.

[35] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[36] W. Sieg and J. Byrnes. An Abstract Model for Parallel Computations: Gandy's Thesis. *The Monist 82*, pages 150–164, 1999.

[37] Clifford Spector. Recursive Well.Orderings. *The Journal of Symbolic Logic*, 20:151–163, 1955.

[38] Karl Svozil. The Church-Turing thesis as a guiding principle for physics. *Unconventional Models of Computation*, pages 371–385, 1998.

[39] A. M. Turing. Systems of Logic Based on Ordinals. *Proc. London Math. Soc.*, s2-45(1):161–228, 1939.

[40] Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.