A HYPERCOMPUTATIONAL APPROACH TO THE AGENT CAUSATION THEORY OF FREE WILL

SERHAN MERSİN

MARCH 2006

A HYPERCOMPUTATIONAL APPROACH TO THE AGENT CAUSATION THEORY OF FREE WILL

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF INFORMATICS

OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERHAN MERSİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COGNITIVE SCIENCE

MARCH 2006

Approval of the Graduate School of Informatics

_____

Assoc. Prof. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science/Doctor of Philosophy.

_____

Prof. Deniz Zeyrek
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science/Doctor of Philosophy.

Asst. Prof. Bilge Say	Assoc. Prof. Erdinç Sayan
Co-Supervisor	Supervisor

Examining Committee Members

Assoc. Prof. Cem Bozşahin (METU, CENG)_____

Assoc. Prof. Erdinç Sayan (METU, PHIL)_____

Asst. Prof. Bilge Say (METU, COGS)_____

Assoc. Prof. Samet Bağçe (METU, PHIL)_____

Asst. Prof. John Bolender (METU, PHIL)_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Serhan Mersin

Signature:

ABSTRACT

A HYPERCOMPUTATIONAL APPROACH TO

THE AGENT CAUSATION THEORY OF FREE WILL

Mersin, Serhan

MSc., Department of Cognitive Science

Supervisor : Assoc. Prof. Dr. Erdinç Sayan

Co-Supervisor: Asst. Prof. Dr. Bilge Say

March 2006, 106 pages

Hypercomputation, which is the general concept embracing all machinery capable of carrying out more tasks than Turing Machines and beyond the Turing Limit, has implications for various fields including mathematics, physics, computer science and philosophy. Regarding its philosophical aspects, it is necessary to reveal the position of hypercomputation relative to the classical computational theory of mind in order to clarify and broaden the scope of hypercomputation so that it encompasses some phenomena which are regarded as problematic because of their property of being uncomputable. This thesis points to a relation between hypercomputation and the agent-causation theory of free will by exploring that theory's alleged infinite-regress feature, which has been regarded by some authors as problematic and used against

the agent-causation theory. In order to cope with this problem, we propose a certain hypercomputer, viz. the reverse Zeus machine. The reverse Zeus machine can help to understand the infinite-regress aspect of agent causation better than accelerating Turing machines (or ordinary Zeus machines). Accelerating Turing machines are abstract machines which perform temporal patterning in an accelerating manner by executing each step in half the time required for the previous step. This allows them to compute infinitely many operations in finite time**.** Although reverse Zeus machines have the same working principle as accelerating Turing machines, we show that agent causation can be represented by reverse Zeus machines better than by the classical Zeus machines.

Keywords: Hypercomputation, computational theory of mind, agent causation, free will, reverse Zeus machines.

ÖZ

ÖZGÜR İRADENİN ÖZNE NEDENSELLİK TEORİSİNE

HİPERBİLİŞİMSEL BİR YAKLAŞIM

Mersin, Serhan

Yüksek Lisans, Bilişsel Bilimler Bölümü
Tez Yöneticisi : Doç. Dr. Erdinç Sayan
Ortak Tez Yöneticisi: Yrd. Doç. Dr. Bilge Say

Mart 2006, 106 Sayfa

Turing makinelerinin yapabildiğinden daha fazla işleri Turing Limiti ötesinde yapma yeteneğindeki tüm mekanizmaları içeren genel bir kavram olan hiperbilişim (hiperberim) konusunun matematik, fizik, bilgisayar bilimleri ve felsefe gibi çeşitli alanlarda uygulamaları vardır. Hiperbilişimin, berilemez oluşları nedeniyle sorunlu oldukları düşünülen bazı olayların özelliklerini aydınlatmak ve genişletmek amacıyla klasik bilişimsel (berimsel) zihin kuramına nazaran konumunu felsefi yönlerini de göz önüne alarak açığa çıkarmak gerekmektedir. Bu tez, sonsuz gerileme özelliğini barındırdığı için problemli olduğu iddia edilen özne nedensellik teorisi ve hiperbilişim arasındaki ilişkiyi açıklamayı hedeflemektedir. Bu tezde, sonsuz gerileme sorunuyla baş edebilmek için ters Zeus makinesi olarak adlandıracağımız

bir hiperbilgisayar öneriyoruz. Ters Zeus makineleri, sonsuz gerileme özelliğini, bir tür Zeus makinesi olan ivmelenmiş Turing makinelerinden daha iyi açıklayabilir. İvmelenmiş Turing makineleri, her adımı bir öncekinin yarısı kadar zamanda ivmeli bir şekilde yerine getiren zamansal davranış gösteren soyut makinelerdir. Bu özellik onlara sonlu zaman içerisinde sonsuz farklı işlemi hesaplamalarını sağlar. Bu tezde, ivmelenmiş Turing makineleriyle aynı çalışma prensiplerine sahip olsalar da ters Zeus makinelerinin özne nedensellik teorisini klasik Zeus makinelerinden daha iyi açıklayabileceğini gösteriyoruz.

Anahtar Kelimeler: Hiperbilişim, bilişimsel zihin kuramı, özne nedensellik, özgür irade, ters Zeus makinaları.

ACKNOWLEDGEMENTS

I would like to thank my supervisors Assoc. Prof. Dr. Erdinç Sayan and Asst. Prof. Dr. Bilge Say for their guidance and always helping me when I needed.

Also special thanks to Asst. Prof. John Bolender for his criticisms and very helpful comments.

I would also like to thank Informatics Institute, METU, and especially to the department secretaries Mukkades Yenilmez Kocakaya and Sibel Gülnar.

Finally, I would like to express by gratitude to my parents, my grandmother, and my aunt for never ceasing believing in me. I am also grateful to my friends, Duygu Özge, Umut Özge, Yurdagül Öztürk for their consistent support.

Tout est bien qui fini bien.

Ankara, March 29th, 2006

Serhan Mersin

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Turing machine concept has played a very significant role in studies of computer science. It is generally accepted to provide us with not only an essential criterion for computability theory but also definition of computation. Computation, as it is adopted by most cognitive scientists, is a basis for cognition, where computational processes are algorithmic and mental phenomena that find their existence within the idea of algorithms. That is, computation gives us an account of cognitive processes to be expressed in computational terms. However, Turing machines have restricted capabilities and that causes implications of computation to be limited as well. Therefore, I will look at a broader view of computation, i.e., hypercomputation, and investigate whether we can extend the implications of hypercomputation as we extend the notion of computation.

1.1 The Aim and the Scope

Hypercomputation can basically be defined as computation that transcends the limits of computation in the Turing sense. The scope of this thesis is to represent hypercomputers as computational machines beyond Turing machine capabilities and investigate the effects of some of these hypercomputers on a problematic aspect of computational theory of mind, namely free will. However, I will confine myself with

a specific theory of free will and point out a relationship between the agent causation theory and hypercomputation. I believe this relationship can be established since it is possible to find parallels with the proposed hypercomputers in the literature, viz., accelerating Turing machines and Putnam-Gold Machines, and alleged intrinsic aspect of agent causation which implies the potential of infinite regress for its accomplishment. The infinite regress aspect has been considered as a problematic issue for agent causation; however, involvement of infinite regress will not prevent us from using it for our hypercomputational approach.

This study makes very strict assumptions in order to narrow the scope of the thesis. The investigation of topics of computation, computational theory of mind and its implications, study of free will and incompatibilist[1] accounts of it are unquestionably broad topics which extend beyond the scope of one thesis project. Therefore, I will take some assumptions for granted and not endeavor to show the validity of them. First of all, I will assume that free will exists and is not compatible with determinism. I will not discuss the reasons given in favor of the controversial incompatibilist views of free will in much detail. Nevertheless, I will show why one of these incompatibilist theories, the agent causation theory, is chosen to be used for implementing on a hypercomputational device. Second, I will assume that hypercomputation is theoretically feasible although it cannot be implemented in practice. But, I will demonstrate hypercomputation is a consistent idea, a tool for extending our view of computation. Also, it is necessary to add that my starting point is hypercomputation. Agent causation theory suffices to give us some relevant features of hypercomputation. Therefore, it is possible to find correspondences

[1] Incompatibilism is a theory of free will which holds that free will is not compatible with determinism. According to incompatibilists, free will exists and therefore determinism is false. This theory will be explained in more detail in Section 4.2.3.

between a topic in computer science and mathematics (i.e., hypercomputation) another one in philosophy (i.e., agent causation).

1.2 Overview of the Chapters

This thesis begins with a short historical account of the origins and developments of the computability theory. In Chapter 2, I will discuss the historical foundations of Turing Machines, its origins emerging from Hilbert's programme and Gödel's theorems. The structure of Turing Machines, possibility of a universal machine which can simulate the behavior of any other Turing machine and equivalence of Turing's concept with some other mathematical concepts in theoretical computer science are examined. Later, I will focus on the Computational Theory of Mind and its implications regarding this study.

In Chapter 3, I will describe a non-classical machine called "oracle machine" as a way of introducing the concept of hypercomputation. Subsequently, I will discuss the common characteristics of hypercomputers, their capabilities beyond Turing machines and some notional hypercomputers.

In Chapter 4, by using Penrose's arguments (1989, 1994) to demonstrate the existence of uncomputable phenomena as an example, I explain how hypercomputers can help us to explore new notions of computability. Later, I attempt to cope with the free will issue. I will restrict my thesis here only to agent causation theory and try to find a theoretical basis for hypercomputational devices to account for agent causation. In this section, I will propose a certain hypercomputer, namely reverse Zeus machine, to explain the issues regarding infinite regress feature better than other hypercomputers.

Finally, a summary of the main arguments will be presented in the Conclusion section with a discussion on main findings and future work.

CHAPTER 2

TURING MACHINES

The concept of computation has derived from the discussions of logic in the early 20th century. These discussions had close ties with the notion of formalization presented by different mathematicians such as Frege, Peano, Russell, Hilbert and Gödel. One of the most influential studies of these mathematicians was Hilbert's programme. This chapter starts with a brief introduction to Hilbert's programme and the theorems of Gödel which solve some of the problems discussed in Hilbert's system. Gödel's theorems are of concern here since they will help us to demonstrate the existence of aspects which cannot be explained by the concept of computation.

2.1 Hilbert's Program and Gödel's Theorems

In 1928, Hilbert and Ackermann raised two very important questions about the formulation of mathematics and logic in their textbook entitled *Gründzüge der Theoretischen Logik* (Principles of Mathematical Logic). Hilbert's idea was to formulate a rigorous, clear and general program for the foundations of mathematics. This program was directed to prove the consistency[2] of powerful systems including

[2] A formal system is consistent, according to one of its definitions, if and only if not every well-formed formula, i.e., strings of symbols of the formal language correctly constructed with respect to its formation rules, of the system is a theorem.

ideal arithmetic by using mere finitary proof methods. Being the first mathematician to introduce first order logic as a subsystem of all logic, Hilbert considered the problems of consistency, independence and completeness of formal axiomatic systems.

The axioms of a system are independent of each other if none of them is logical consequence of the others, and the completeness of a formal system concerns whether every true statement of the system is provable or not within that system. Hilbert proved the consistency of propositional and first order logic. However, the problem of completeness of higher order logical calculi, e.g., first order logic, was still open. Therefore, the first important question he raised was about the issue of completeness, whether every true mathematical statement is provable or not within that system.

This question was answered positively by Gödel's Completeness Theorem one year later in his doctoral dissertation published in 1930. In his thesis, Gödel proved the semantic completeness of first order logic by showing that every valid first-order formula is provable in the system defined by Whitehead and Russell's *Principia Mathematica* (1910-1913*)*.

One year later, in 1931, Gödel published a very significant paper featuring his incompleteness theorems. Gödel's so-called Incompleteness Theorems are two theorems which are connected with each other. His First Incompleteness Theorem is about the arithmetized formal systems and can be explained informally as "All consistent axiomatic formulations of number theory include undecidable propositions" (Hofstadter 1999, p.17). Gödel's point is to show that every sufficiently powerful formal system is either inconsistent or incomplete, i.e., if any

formal system containing arithmetic is consistent, then it is incomplete or if it is complete, then it is inconsistent.

Gödel's Second Incompleteness Theorem states that if an axiomatic system containing arithmetic is consistent, this consistency cannot be proved within the system itself. This means even the consistency of arithmetic of natural numbers is not provable by finitistic methods. This was a remarkable and unexpected result since it undermines Hilbert's program which "revealed not only that there were irreparable 'holes' in the axiomatic system proposed by Russell and Whitehead, but more generally, that no axiomatic system whatsoever could produce all number-theoretical truths, unless it were an inconsistent system!" (Hofstadter 1999, p.24). Therefore, absolute proof of consistency was proved to be impossible and therefore had a negative solution.

One of the important claims presented in the proof of Incompleteness Theorem is that it is applicable to any axiomatic system that contains arithmetic, not only to the system defined by Whitehead and Russell, showing that every such system will include arithmetic statements which are either unprovable but true (i.e., the system is incomplete) or provable but false (i.e., the system is inconsistent).

The second problem raised in Hilbert's and Ackermann's book is the *Entscheidungsproblem* (decision problem), the problem of finding a general decision procedure for whether a given formula of first order logic is universally valid and satisfiable or not. An argument is valid if and only if the (assumed) truth of its premises guarantees the truth of its conclusion. That is, the conclusion logically follows from the premises. Likewise, an argument is satisfied when all the premises are true.

The decision problem is revealed as a "problem in the algebra of logic" (Gandy 1994, p.56), and therefore, in principle, can be raised for all mathematical statements. It could be asked whether we can find a general procedure (or an algorithm) to solve all problems of mathematics. Hilbert actually suggested the decision problem as the main problem of mathematical logic since it would be possible to settle any mathematical question by using such a general algorithm.

Following Hilbert's question whether a general algorithm could exist and could be designed to solve any particular mathematical problem, another issue concerning the existence of an effective method which can identify classes of intractable (very difficult to deal with or find answer to) problems was raised. Turing machines were the first examples of theoretical devices which could operate to solve these problems.

2.2 Turing Machines

It was the British mathematician Alan Turing who in his paper, "On Computable Numbers, with an Application to the *Entscheidungsproblem*" (1936), proved that *Entscheidungsproblem* (decision problem) is unsolvable by showing that there is no such general-purpose algorithm. He developed his proof by formalizing the mathematical groundwork of computability theory and the notion of Turing machines. In this paper Turing investigated the possible processes to carry out in computing a number. His purpose was to encompass the idea of what a "human computer" which is constrained by limitations of human beings, such as bounded storage capacity and memory, would do in order to follow a procedure. Taking these possible processes as algorithms (or mechanical procedures) and using a Turing machine simulating the behavior of a human computer, Turing argued that any

algorithmic process could be carried out by a Turing machine. If *Entscheidungsproblem*, like any other problem, had an algorithmic solution, then it would be possible to show it by a Turing machine. Turing, then, proved that no Turing machine could solve *Entscheidungsproblem*. This precisely showed *Entscheidungsproblem* is unsolvable.

Alan Turing developed the notion of Turing machine as a precise model of computation. But why is a Turing machine needed to show *Entscheidungsproblem* is unsolvable? What is its significance in the theory of computability? Or what makes it a precise model? In order to answer these questions, first of all, it is necessary to clarify the notions of effective procedures and algorithms.

2.2.1 Effective Procedures and Algorithms

In computer science the notion of "effective" can be used as loosely equivalent to the notions of "mechanical", "constructive", "finitistic", and "algorithmic". A procedure is effective in the sense, as cited in (Beckman, 1980) who provides an extended version of Turing's definition, that it has the following specific characteristics:

(i) An effective procedure is *deterministic:* Any effective procedure must yield the same result if it is repeated by going through the same starting conditions and same steps where each step is precisely defined.

(ii) An effective procedure is executable in *finite* time, by *finite* number of steps and by using *finite* facility. If necessary, the facility can be increased by using external medium. Hence, it becomes possible to answer any of infinitely many questions by finite means. Also, an effective procedure has finite length**.** This means the description of all the steps of the procedure is finite.

(iii) The execution of an effective procedure is *mechanical*, i.e., it must be precise, discrete and clear. Each step must be exactly or sharply defined so that this description can be used by some other medium to get identical results.

(iv) Procedures can be cast in *numeric terms*: objects can be represented by positive integers and operations on these procedures are arithmetic operations. (Beckman 1980, p.2)

In a broader sense an effective procedure can be defined as a set of instructions which tells precisely how to behave. However, in the case that these instructions (or rules) are applied exactly as they were told, without making any kind of change or innovation into, how is it possible to know that these instructions are followed correctly, or the answer is correct? Minsky (1967, p. 105) acknowledges the process is supposed to terminate in "in finite, already known, time." But, consider, for instance an infinite loop, where the fifth step of the algorithm tells to go to first step. Then the process never ends. For Minsky this problem can be solved another feature of effective processes:

> But if the length of the process isn't known in advance, then "trying" it may not be decisive, because if the process does go on forever–then at no time will we ever be sure of the answer. Our concern here is not with the question whether a process terminates with a correct answer, or even ever stops. Our concern is whether the next step is always clearly determined (ibid.).

This means the notion of effective procedure does not necessitate any kind of intelligence. In Turing's view, too, an effective procedure is an algorithm which can be carried out by human beings without the use of any complex machinery, even without intelligence. In that sense, a (classical) algorithm is one which can be implemented by a machine that computes computable functions.[3]

[3] A function is computable (or Turing-computable) if and only if there is a Turing machine that is able to compute that function.

It is possible to introduce a simple device or class of devices and realize the notion of algorithms or effective procedures as executed by this machine. This is what Turing had done: he brought a formal description to these processes. His machine served as a precise way of formulating the notion of algorithm. Consequently, "effective procedure" meant what a Turing machine carried out and "computation" meant actions of a Turing machine.

In the following section I will try to explain several features of Turing machine model that are important to understand the notion of computation.

2.2.2 The Structure of Turing Machines

Turing machines are idealized, abstract devices which have specific descriptions (Minsky, 1967). A typical Turing machine is composed of a linear *tape* which is infinite in both directions and divided into a large number of separate *squares* or cells bearing *symbols* from a finite alphabet S_0, S_1, S_2…S_n. Although it would be possible to use infinite number of symbols, in Turing's view, the symbols would differ only in very small extents in case of infinity and therefore confusion could not be avoided[4]. The tape is thought to be potentially infinite. At any time of computation a finite set of squares will contain symbols other than blank (i.e., no symbols). Hence, the cells contain only blank on both sides of the tape and a blank symbol can be written to the tape whenever needed by extending its computation beyond the original input. The infinite tape passes through a programmable head. The head is always positioned over a square or cell of the tape and is capable of inspecting and modifying the symbol on the square by its read-and-write head which

[4] For instance the difference between the symbols 001000001010001111000000100 and 001000001010001111000000110 would be indistinguishable in the first glance.

can go left or write. Finally, a Turing machine has a finite number of distinct internal states q_1, q_2, $q_3 \ldots q_n$.

At any time, the possible behavior of a Turing machine is described by its particular scanned symbol under the read-write-head and its current internal state. Figure 1 demonstrates a sample Turing machine description.

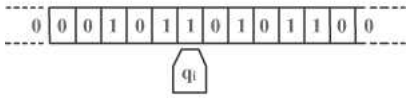

Figure 1: Depiction of a Turing machine with controller in state q_i

The scanned symbol and internal state pair is called *configuration* of the machine. The complete configuration of the machine can be determined by its scanned square, all symbols on the tape and internal state, or in the form of built-in quintuples. These quintuples can be described as follows:

(symbol read, old state, new state, symbol written, direction) , or

$$(S_i, q_i, q_j, S_j, X)$$

where S_i and S_j are symbols from the alphabet, q_i and q_j are states and X has either the value L (Left), R (Right) or halt. This quintuple can be interpreted as follows: If the tape head is scanning S_i and the machine is in state q_i, then change the state to q_j, replace the symbol S_i by S_j and move the head either one square left or right or do not do anything. If there is no more instruction to perform, the computation process ends, and the machine halts with the output written on the tape. Here, both the states q_i and q_j and the symbols S_i and S_j may be identical with each other. Moreover, since the symbols to be modified may be blank, the machine is able to reach any portion of the tape on both sides. The input and the internal state of the Turing machine strictly determine its potential behavior. During any moment of computation at most one

application can be performed: the machine can change the scanned square by shifting it just one place, to Left or Right. Also, it is possible to describe all the actions of a Turing machine on a table of rules or a "machine table" where every step is exactly defined in a tabular manner.

To represent how a Turing machine works, the following example (from Minsky 1967, pp.120-121) will be helpful to understand the underlying mechanism. This example involves a Turing machine investigating whether a given number of strings of 1's is even or odd. If the number is even, it will return 1 and otherwise, 0. Consider a series of 1's and 0's on the tape and assume all other cells on the tape to be 0. The machine in its initial state starts processing by scanning the leftmost 1 on the input tape (Figure 2).

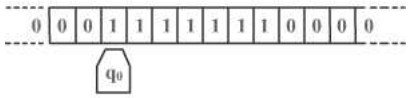

Figure 2: The representation of the input string in the Turing machine's tape

The machine will move to the right one square at a time by scanning all 1's and replacing them with 0's. It, eventually, will arrive at an empty square and replace 0 on that square with 1 if the number of consecutive 1's is even or print 0 if it is odd. Finally, it will move one square to the left and halt. The output tape will be the resulting square (Figure 3). The intermediate steps and the formal description of this computation can be seen in Appendix A.

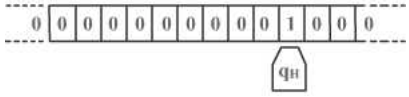

Figure 3: The representation of the output string in the Turing machine's tape

However, it would also be possible to have a slight change in the configuration of the tape. For example, instead of having the output tape on the rightmost of consecutive 1's, we could have it on the first 1 of the series. After scanning all the input tape and arriving at the empty square, instead of printing the output on that cell, we would move back, replacing all 1's with 0's and arrive at the leftmost cell of the input tape and change it with 1 if the number of the consecutive series is even or with 0, if odd. However, this also shows that there are always other possible solutions during computation. For example, it is possible to get 5 either by adding 2 with 3 or adding 3 with 2. Both of these ways and even some other ways are possible. Nevertheless, the idea and processing of computation is the same.

This description of Turing machines shows that the Turing machine is a model of finite computation. The finiteness feature is presented in different ways. For instance, Turing (1936) explicitly discussed the finite but unbounded tape or memory of his machine model. Likewise, the number of states should also be finite in the sense that the number of instructions is finite. The number of computational parts of the Turing machines should be finite as well. Furthermore, the number of symbols from the alphabet, since these symbols can be manipulated to be arbitrarily close to each other as they would differ only in very small extents in case of infinity, is finite, too. That is, although the tape is considered to be infinitely long as a mathematical idealization, the input, output and calculation (or computation) is finite. Turing, however, did not refer to the time of the operation of his abstract machines. The time factor was not taken to be a crucial feature of Turing machines.

It is possible to go one step further and discuss a particular machine which can imitate any other Turing machine.

2.2.3 Universal Turing Machines (UTM)

The notion of Turing machine led Turing to conceive of a "single machine which can be used to compute any computable sequences" (Turing 1936, p.127). Actually, it is easy to see that the number of possible Turing machines is infinite since a new one can be constructed by introducing a new algorithm to the computation. This means that each Turing machine can compute only one particular function. However, Turing assumed that a single machine could do anything that any Turing machine could do. He advocated the idea that an appropriately programmed Turing machine could be used to perform all possible computations (thus imitating any other Turing machine) in a standard form. He called this particular machine Universal Computing Machine (or *Universal Turing Machine* as it is widely called now).

Turing claimed that a Universal Turing Machine could simulate the behavior of any Turing machine whose code number of instructions or quintuples are given as input data to it. Turing showed in his paper how the quintuples of that machine could be produced. This idea enlivened the concept of a general purpose computer of which a Universal Turing Machine can be represented as a model. In that sense, any modern time general purpose computer is equivalent to Universal Turing Machine because there exists a single machine with which the appropriate program can perform all tasks that can be carried out and store the instructions into its writable memory (Davis, 2000). A Universal Turing Machine is an example of an "interpretive" program as well since it works out the instructions or quintuples of a given program by carrying out the given procedure. It is possible to extend further these interaction points or analogies between a Universal Turing Machine and a real computer. As Davis states, first of all, a Turing machine simply can be considered as

a machine with mechanical parts or *hardware* in the modern jargon (Davis 2000, p.165). Second, any Universal machine can be coded with a set of quintuples for the machine by following a step-by-step procedure in order to carry out a computation and thus functions as a *program*. Finally, the codes implemented by sequences can be executed as *data*. Therefore, it would not be difficult to program a Turing machine supplied with storage facilities such as unlimited internal storage capacity.

2.3 The Halting Problem

The acknowledgement of the unsolvability of the *Entscheidungsproblem* showed that there exist unsolvable decision and computation problems which do not have algorithmic solution. Any problem of Turing machines that has an algorithmic solution is called solvable problem and an algorithm that solves a problem is called decision procedure. One of the most famous examples of unsolvable problem for Turing machines is the *halting problem*: to determine for a given Turing machine M and input tape w, whether M will eventually halt on input w. Minsky has presented a proof of the unsolvability of halting problem in his theoretical computer science textbook which became a classic on the theory of computability (Minsky 1967, pp. 148-149). In order to show that there is no effective procedure to solve the problem, he used *reductio ad absurdum*. He assumed that there exists a machine which can decide whether or not any Turing machine computation will ever halt and thus solve the halting problem. Call this machine D, and given the description pair with the Turing machine M and input w as (M, w) it will give an answer YES (or 1) if M eventually halts, given input tape w, and gives NO (or 0) if M never halts given input tape w. We can identify D with the input string <M, w> as follows:

$$D(<M,w>) = \begin{cases} \textit{YES or 1} \text{ if M eventually halts on w} \\ \textit{NO or 0} \text{ if M never halts on w} \end{cases}$$

Then it is possible to modify D and get another machine E. E behaves like D and requires only the description of M's own description (M,w). It then takes D to describe what M does when input to M is its own description (M,w_d)**.** E would be in a state that it could enter in a loop instead of that of M which would stop and give the answer YES. Thus, the machine E can be described such that:

$$E(<M, w_d>) = \begin{cases} \textit{YES or 1} \text{ if M never halts on } (M, w_d) \\ \textit{NO or 0} \text{ if M eventually halts on } (M, w_d) \end{cases}$$

Now, it is still possible to modify E and produce E* which will duplicate its own description (M,w_E) as input. Similar to the previous case, it follows E* will have a behavior that when it is applied to w_d it eventually halts, if M applied to w_d goes into an endless loop, i.e., never halts and it will never halt if M applied to w_d eventually halts. Therefore, in this new state, if E* is applied to w_E does not halt, E* halts and E* does not halt if E* applied to the w_E halts. This situation can be expressed as follows:

$$E^*(<M,w_E>) = \begin{cases} \textit{YES or 1} \text{ if M never halts on } (M, w_E) \\ \textit{NO or 0} \text{ if M eventually halts on } (M, w_E) \end{cases}$$

Here, therefore, a contradiction has been reached. Consequently, our first assumption is false and we can conclude that none of the machines D, E, E* exist.

As a result of the undecidability of the halting problem, there are a large number of problems which can be deduced to be unsolvable. The following examples are some variations of the halting problem:

(i) Given a Turing machine M, does M halt on the empty tape?

(ii) Given a Turing machine M, is there any string at all on which M halts?

(iii) Given a Turing machine M, does M halt on every input string?

(iv) Given two Turing machines M_1 and M_2, do they halt on the same input strings? (Lewis & Papadimitrou 1992, p.255)

Moreover, Minsky (1967, pp.150-152) gives other examples of related unsolvable problems:

(v) Does machine T ever print the symbol S_0 when started on tape t? ("Printing problem")

(vi) Given a Turing machine T, contrary to other Turing machines which are considered to start a given initial state with only finite inscription on the tape, is there an internal state Q and some infinitely inscribed tape for which T will not halt when started on that tape in state Q? ("infinitely printed tape problem")

The negative solution of the halting problem indicates the existence of uncomputable functions and hence unsolvable computation problems. Any Turing machine making computations should theoretically be capable of terminating when it is programmed to do so. However, due to existence of uncomputable functions, there may exist some non-terminating cases which make it *impossible* to find a general method to decide for every program whether M will stop or not.

2.4 Church-Turing Thesis

In an attempt to find a solution to *Entscheidungsproblem,* Alonzo Church (1936), independently of Turing, proved the insolvability of this problem by using a different approach. Church used the concept of lambda(λ)-definability[5] instead of Turing computability *where* computability was defined from a standpoint of Turing machines which are formally equivalent to the intuitive notion of algorithms. He demonstrated that there were algorithmically unsolvable problems and the decision problem was arising in the theory of λ-definability. In Church's analysis the crucial notion was "effective calculability." His purpose was to define the notion of effectively calculable function of positive integers by identifying it with the notion of a recursive function (in the sense of Gödel (1934) and Herbrand (1932)) of positive integers (or of a λ-definable function of positive integers) (Church 1936, p.100). This definition is two folded: (i) a function of positive integers are effectively calculable if it is λ-definable and (ii) a function of positive integers are effectively calculable if it is recursive. Church, in his paper, proved every recursive function is λ-definable. The converse was also shown to be true. Church asserted that while being widely different but equally natural by definition, the equivalence of the notions of recursiveness and λ-definability proves the strength and consistency of his notion of effective calculability.

Turing showed that his notion of computability and Church's assertion of λ–definability are equivalent. This equivalence is now known as *Church-Turing's Thesis*:

[5] Functions which are definable in lambda calculus. Lambda calculus is formalism for representing functions and ways of combining functions, invented around 1930s by Alonzo Church. It formalizes the concept of effective computability and is universal in the sense that any computable function can be expressed and evaluated using this formalism. (Oxford Dictionary of Computing, 1996)

> Any effectively computable function (of positive integers) is Turing computable.

Therefore, the class of effectively computable functions can be identified with the class of Turing machine computable functions. Consequently, as Turing's notion of computability by Turing machines is equivalent to Church's and as Church has proved that his notion of effective computability is identical with Gödel and Herbrand's notion of (general) recursiveness theory, it is possible to say that all these different systems describe the same class of mathematical functions. Church confirmed the equivalence of these systems and admitted that Turing's thesis is more satisfying than his:

> [Computability by a Turing machine] has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately – i.e., without the necessity of proving preliminary theorems. (Church 1937, pp.42-43)

2.4.1 Misunderstandings of the Church-Turing Thesis

There have been several misunderstandings of Church-Turing Thesis among cognitive scientists and philosophers of mind. Copeland collects these improper and mistaken versions, which concern the extent of machine calculation, under the rubric "so-called Church-Turing Thesis" and mentions them in several articles (Copeland 1997, 1998a, 1998b, 1999, Copeland & Proudfoot 2000, Copeland 2000b, Copeland 2002). He states that the actual thesis that Church and Turing put forward concerns "the functions that are in principle computable by an idealized human being unaided by machinery" and "the limits of what an ideal human mathematician can compute coincide with the limits of what a universal Turing machine can compute, and carries no implication concerning the limits what a machine can compute" (Copeland 1998a, p.157). This statement acknowledges that Church-Turing Thesis has nothing to do

with calculating devices or machines but is merely stressing the idea of mechanical or effective procedures. Therefore, it does not imply a physical analysis. It does not say anything about the theoretical limits of what a machine can compute. Thus, it would be possible to conceive of machines which can compute more than Turing machines.

One of the best representations of Church-Turing thesis can be described as in Figure 4. The portion of the effective human behaviors with regard to all human behaviors covers only a small extent of these behaviors. The effective behaviors are the ones which are Turing computable. Likewise, humans have the ability to simulate computers. That is, it is possible to show a relationship between the physical behaviors of human beings and mathematical behaviors supporting formal arithmetic.

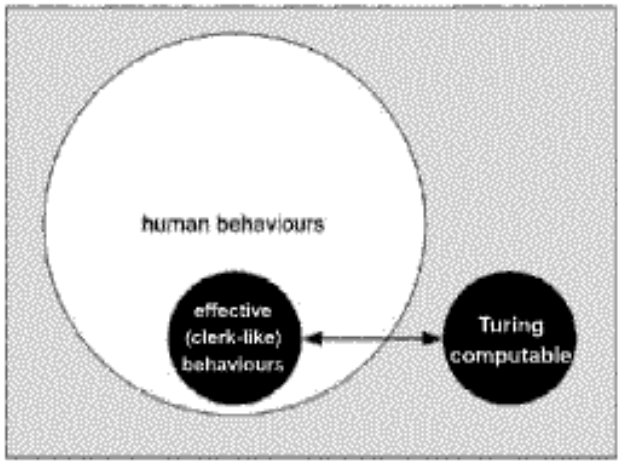

Figure 4: Church-Turing thesis (Adapted from Stannet 2003, p.119)

Second, another controversial issue called "Turing machine functionalism" is widespread in the literature. Since a Turing machine, in a very abstract sense, is an input-output device where the instructions implemented by sequences can be

executed as data and it can be coded with these instructions to follow a step-by-step procedure in order to carry out a computation, the brain can also be considered as a Turing machine. Therefore, Turing machine functionalism can be considered as machine- state functionalism: any mental state is machine state. As a result, mental states are realizable by Turing machine table states. Since anybody can determine the actions of a Turing machine by its given table of rules where every step is clearly defined, mental phenomena can be identified with complex instructions implemented within the machine table. As the logical states and the mental states have functional organizations (Putnam 1975, p.20) and are equivalent, a mind and machine relationship is provided and mind can process information by simply implementing a Turing machine.

It is this notion of equivalence that has been problematic. Many cognitive scientists and philosophers of mind used this idea to extend machine functionalism to activities of human psychology. Although this kind of functionalism gave birth to a new field of study called computationalism, these notions have different aspects. For instance, computationalism differs from machine functionalism "by locating the mental in abstract computational states rather than in the various possible machine states which could implement them" (Fodor 1975, p.27) and is neutral on whether computational relations constitute the nature of mental states. Machine functionalism attempts to characterize the mental by decomposing the whole system into components and the system is expressed in terms of these functional terms. Likewise mental processes can be decomposed into a point where they can simply be thought as processes of a Turing machine. The misunderstanding here stands as an expectation of Turing machine computable functions "to suffice also for characterizing the behavior of the rest of the universe" (Copeland, 1998a). The

halting function is a response to this misunderstanding as no Turing machine can display a systematic pattern to solve this problem. That is, despite the view that functional organizations are specified by mechanistic explanations, there is a room for the possibility of mechanistic theories of mind according to which the functional organization of the mind is not computational.

In the following section I will discuss an important topic in philosophy of mind, called computational theory of mind, which has roots in Turing machine computability.

2.5 Computational Theory of Mind

Mind design, which is a term used mostly by philosophers of mind, such as Haugeland, 1981 and 1997 to explain mind in relation to its design such as how it is built and how it works, is mainly concerned with understanding mind, producing new models and finding appropriate explanations for intelligent behaviors. One of the theories to explain mind, Computational Theory of Mind (CTM) proposes the idea that all mental phenomena can be explained in a syntactically specifiable structure systematically by algorithms. In the next section I will try to explain and state the assumptions of this theory.

CTM is born with the idea of computationalism. Basically, computationalism postulates that mental processes are nothing but computational processes, i.e. mind is a digital computer, a syntactic machine manipulating symbolic representations. This approach has its roots in Turing computability. As it was discussed in Section 2.2.1, computation is an action of a Turing machine. Alonzo Church proposed that Turing machines are, in principle, able to do anything that any computing machine can do by simple but well-formed rule-based representations of “if you are in state P and

have input Q, then do R" (Church, 1936). Using these kinds of formalizations lead us to develop a syntactic system of mind driving on semantic grounds. The idea behind this assumption emerges from the relationship between formalization and syntax: formalization shows how to link semantics to syntax, and computation (in Turing sense) shows how to link syntax to causal mechanisms. That is, all mental processes are causal processes for which a causal mechanism could be specified in the sense that a purely physical system (i.e., digital computer) "carry out symbolic inferences that respect the semantics of the symbols without recourse to a homunculus[6] or to any other nonphysical agency" (MITECS, 2002).

2.5.1 Language of Thought Hypothesis

The symbolic approach of CTM has been described in an environment called Language of Thought (LOT). LOT is a mental language which provides a medium for thinking. This language is innate and consists of internal representations and rules which form the syntactic structure of our thoughts with appropriate semantics. When Fodor offered his hypothesis in his book *The Language of Thought* (1975), he established his ideas to be grounded in scientific theories and cognitive models.

Since LOT Hypothesis (LOTH) has a framework for explaining thought on representational systems, it is also defined as representationalism. Representational realism is posited in Representational Theory of Mind (RTM) which attempts to explain all psychological states and processes in terms of mental representation. For that reason, a mentalese sentence is like having a representational token with its semantic content. When a person has a thought that "X is tall" the content of that thought has a representation in the person's mind by a sentence. Some properties of a LOT have similarities with construction of natural languages. For instance,

[6] Literally "a little man" but can also be understood as "an entity or agent".

mentalese sentences have grammatical or syntactic structures, i.e. rules which make any sentence well-formed. Syntax is concerned with the ways in which words are combined together to form sentences. However, the same set of words with very slightly different ordering can have completely different meaning. For example, the sentences "What does it taste like?" and "What taste does it like?" have both different meanings although the sets of words used are same. This shows that syntax is not the only component which is sufficient for all mentalese sentences: they should also have a semantic content. Semantics, in general, is the study of linguistic aspects of meaning. Any sentence which is well formed with correct grammar must have a further feature called meaning. Chomsky's (1957, p.15) famous example "Colorless green ideas sleep furiously" is totally grammatical but meaningless. The words used together can have meanings when used in their appropriate context. The example sentences cited above "What does it taste like?" and "What taste does it like?" shows not only the necessity of a semantic content but also *systemacity* of language. The ability to understand or produce a sentence helps to understand or produce many others. This is an intrinsic capability for all language users; if you know the syntax and have lexicon, you will be able to combine words together and construct sentences, and even understand sentences which you have never heard before. It would be impossible to memorize all possible phrases which a person would ever use in his life.

The same is valid for thought. Thoughts are essentially combinatorial and this situation allows atomic units to be combined in more complex structures to create new thoughts. Certain thoughts orient to others and you are able to connect them with each other. Besides systemacity, thought has *productivity* feature. In principle, it is possible to produce infinite number of sentences with finite set of vocabulary and

syntactic rules. For Fodor (1975), to account for the productivity, language of thought must have also compositionality of syntax and semantics. By compositionality, Fodor means systematically connected thoughts are semantically related to other thoughts which are composed of the same semantic elements. Put in other words, when it is said "Mary loves John" this sentence is meaningful with its components and productive to form another sentence such as "John loves Mary", but irrelevant to the sentence "The weather is sunny." Thoughts connected together have a combinatorial structure where similar thoughts are semantically coherent with each other. Consequently, when the diversity of thought and capabilities to produce new thought are considered, LOTH proposes a valid system since thought is assumed to be linguistic.

2.5.2 Different Senses of Computation

Having discussed the representational system underlying computational theory of mind, it is now possible to focus on procedures of computation. There are different modes of explaining computation. This section gives information about these modes. We cannot understand computability theory without knowing what the notion of computation stands for. As it was shown before in Section 2.2.2, the standard view of computation is based on Turing-machine-computability. The assumptions of CTM are also expressed in this standard view. However, we should not disregard other possibilities, which are not based on Turing machines. These alternative modes can supplement CTM in the ways which will be mentioned.

The symbolic model (or classical model) is the computational view of cognition which consists of formal rules for manipulating formal symbols (Boden, 1998). Two of the most significant advocates and founders of this system, Alan

Newell and Herbert Simon, state a general law of qualitative structure of symbol systems as follows: “A physical symbol system has the necessary and sufficient means for general intelligent action” (Newell and Simon, 1976, as cited in Haugeland 1997, p.87). They state intelligence in terms of rule-based manipulation of syntactically structured symbols, and intelligent behavior can be expressed by means of formal rule systems. All formal systems have some features in common such as being symbol manipulators, being digital and being independent of any kind of medium. Similar to computers, which are physical symbol systems using physical symbols or collection of symbolic structures, symbolic systems operate through rule-governed transformations of distinct functional elements (symbols), and have access only to the form of the symbols (i.e., syntax), not to their meaning (i.e., semantics). Symbolic systems are constructed of top-down organizations. Roughly put, top-down organizations are composed of well-defined and clearly understood fixed computational procedures. The hierarchical structure of the classical view makes it central to the areas where rule-following is essential, such as playing games like chess and where data is well-defined and precise. The success of symbolic systems is not, of course, restricted to areas mentioned above. Problem solving, natural language processing, robotics, learning, perception, vision are some fields the classical approach can study quite well (Boden, 1998). Another aspect of most symbolic systems is their serial architecture (ibid.). They use sequential programming; do their computations step-by-step by proceeding from a given input to reach a desired goal. This structure may cause symbolic systems to be inefficient when they encounter a problem and cause them to learn slowly. Symbolic models take problem solving as an important aspect of intelligence. The issue of how to

reason for accomplishing goals is an indicator of computational power. For that reason problem solving is an important concept for the symbolic camp.

Another mode of computation is connectionist approach. Unlike symbolic mechanisms of mind, the connectionist model (or artificial neural network model or parallel distributed processing model) advocates the brain model of mind. The main idea of connectionism is that "cognition can be modeled as the simultaneous interaction of many highly interconnected neuron like units" (Franklin, p. 122). In that sense, the connectionist model is mostly a mathematical model based on the typical structure of the nervous system of human beings[7]. For that reason, it tries to form its architecture by using artificial neural networks. Artificial neural networks are composed of a large number of highly interconnected processing elements that are analogous to the functional structure of brain cells called neurons and are tied together with weighted connections that are analogous to synapses. Learning in connectionist models can be viewed as the change of the weight (units which are connected by synaptic strengths are called weights) of the links between units. After the network learns to produce the desired output then it may also learn to generalize the behavior and succeed in giving exact output for inputs unknown previously. Here, an important point is that connectionist networks work in a parallel fashion, i.e. they have a system of computations which work independently from each other simultaneously.

Connectionist models offer different systems of representation than symbolic approaches. Regarding a specific type of connectionism, called parallel distributed

[7] It should be expressed that this is not always correct. The connectionist models used for cognitive modeling, unlike the models in computational neuroscience, can have very abstract neural units as compared to human neurons. However, some common properties exist, such as plasticity and graceful degradation.

processing, some connectionist networks use distributed representation rather than local representation. They remain at sub-symbolic level and information-processing patterns are brain-like. They have flexibility in response to new situations and by the help of their multiple structure, they would attempt to solve the problem by dividing it into smaller units so that the failure would affect only this specific unit gradually without collapse of the whole system. Connectionist approaches use bottom-up organizations rather than top-down since these systems do not have well-defined rules which are specified before (the rules and symbolic representation can be emergent properties, though) and the system's architecture requires it to learn and increase its knowledge by experience (in a trial and error procedure of back propagation).

Besides classical and connectionist models of computation, there exists dynamic approaches to cognition. These approaches are investigated under the name of dynamic hypothesis in cognitive science, or simply dynamicism. Dynamicism, as its name implies, explains mind as a dynamic system (Thagard, 1996). The dynamic system, without postulating a set of representations and processes, provides powerful set of tools for understanding complex systems by, for instance, following examples from physics and biology and develop differential equations to show how the mind changes over time (ibid.). Time plays an important role here. Time variable in dynamical models is different than discrete orders but is identified with a quantitative and continuos approximation to the real time events within dynamical laws described by some sets of equations (van Gelder, 1999).

The dynamic system encloses three different ways in which mind has been viewed (Thagard, 1996). First of all, all dynamic systems exhibit an explanatory pattern applied to cognition by using a small number of variables and equations such

as the ones applied in decision making and language growth. Second, dynamic systems can be used metaphorically in order to describe changes in complex systems when they are not successful to specify variables and equations. These changes can be identified in state space (which is a set of states to be determined by the variables that are used to measure it), phase transitions and chaos. Third, connectionist networks are generally dynamical systems (van Gelder, 1999) and therefore it is available to use connectionist models as they are described in dynamical system terms (Thagard, 1996).

The dynamic systems, according to dynamicists, are advantageous to computationalist and connectionist systems since they provide new set of ideas for describing changes in intelligent systems. For dynamicist researches they can explain the nonrepresentational aspects of human behaviors such as the motor control and moods well. Moreover, they constitute a big system of mind and world where they are combined together. They provide a commonality between the world, the body and cognition (Port, 2001).

The opponents of dynamical systems hypothesis focus on the differentiation of dynamicism from connectionism. Although it is generally accepted that dynamicists have successfully distinguished themselves from the computationalist approaches, it is not still clear that why dynamical systems approach should replace the connectionist approaches. Therefore, instead of seeing it as a new paradigm, dynamical systems hypothesis should better be seen as an adjunct to connectionism (Eliasmith, 1996, Thagard, 1996). Besides its relation to connectionism, dynamicism is also criticized for its rejection of representation and computation. In spite of the fact that dynamics systems approach may be effective in explaining the

nonrepresentational aspects of human behavior, there are undoubtedly some representational behaviors such as problem solving.

It is worth emphasizing that a combination of these models can be used in different ways. Some models carry both the aspects of symbolic and connectionist approaches. For instance, many connectionist models are able to represent structures through mechanisms such as dynamic temporal binding. These hybrid models (such as in Harnad, 1993) can combine aspects of both sides.

Consequently, although their natures of computations differ, these different approaches based on the arguments of CTM assist our understanding of mind. Studying these senses of computation can provide an understanding of at least what problems each mode brings about, and specification of the methods of solutions.

2.5.3 Arguments Against Computational Theory of Mind

The CTM has been under attack from various directions. The opponents of CTM such as Searle (1980) claim that mind cannot be a machine, all computers just do is to help us to understand mind, and computation is not sufficient for understanding. They think that there exists semantic structures (for which syntactic structures are not themselves sufficient) and non-algorithmic aspects.

In his article "Minds, Brains, and Programs" (1980), Searle focuses on understanding and tries to discredit the claims of strong artificial intelligence (i.e., computationalism) through his Chinese room argument. Shortly, Chinese room argument is directed at the claim that having a mind is having a program. In this thought experiment, Searle, who is an English speaker and does not know any Chinese, is locked in a room equipped with a pile of Chinese writings (i.e., a sort of database) which are "all Greek to him" and another pile of instructions or rules in

English which helps him to correlate the second one with the first. From a door another pile of instructions in English are given full of Chinese symbols which enables him to correlate this third pile with the first two piles. The rules or instructions tell how to give back Chinese symbols with different kinds of shapes (i.e., input) as a response to different kinds of shapes given in the third pile (i.e., output). Chinese executors of the experiment outside the room call the first pile of papers as "script", second pile as "story" and third pile as "questions" or and the symbols he gives back as "answers to the questions" and the set of rules in English as "program", unknown to Searle. After a time, Searle gets so good at answering questions in Chinese that from the point of view of a third person he is thought to know Chinese very well. The point is here, Searle claims, that he does not understand any word of Chinese. All he does is, analogous to computers, producing "answers by manipulating uninterpreted formal symbols" (Searle 1980, p.69). For him, understanding requires something else since it "has nothing to do with computer programs, that is, with computational operations on purely formally specified elements" (ibid.). The reason for that is, he explains, formal models, such as programs, inherently lack intentionality since intentionality is a biological phenomenon. Therefore, programs, or any inorganic stuff like metal or silicon, do not have the right causal powers to understand whereas brains do.

Undoubtedly Searle is not the only one against the theses of CTM. Penrose (1994, 1997), for instance, criticizes CTM from a different perspective. I will discuss Penrose's arguments and relevancy to my thesis in Chapter 4. Alternative models of computation (i.e., connectionist and dynamic approaches), which were discussed before, extend the symbolic/computational descriptions on which the CTM is based. That is, these models question the bases of symbolic approaches and look into the

subject matter from different perspectives. Others, such as Dreyfus (1972, 1992), argues that human knowledge and competence cannot be reduced to mere algorithmic procedures and therefore cannot be reduced to computer program. Human cognition necessitates a particularly expert knowledge which cannot be captured in algorithmic procedure (Horst, 2005). Moreover, Putnam (1980) and Searle (1990b) point out that syntax does not explain semantics, since syntax is insufficient for semantics.

Chinese room argument is a response to CTM which is based on a standard understanding of computability. If, however, we replace this understanding with a more general view of computability whose description is not restricted with mere manipulation of formal symbols, then it can be possible to escape from the asserted problems. Thus, we accept the logical existence of some other devices, such as o-machines, which can carry out applications that Turing machines are incapable of doing. In the following section, I will discuss the extended notion of computability and identify the features of these devices, namely hypercomputers.

CHAPTER 3

ORACLE MACHINES AND HYPERCOMPUTATION

The idea of conceiving devices which can extend conventional conception of computation does not require the elimination of the conventional models. On the contrary, classical and nonclassical models can complete each other to obtain a wider view of computability. In this chapter, the nonclassical models and the machines proposed to perform computations which are proved to be impossible by classical machines will be introduced. These machines will establish a basis for implications to specific field of study which will be explained in the following chapter.

3.1 Oracle Machines

3.1.1 "Systems of Logic Based on Ordinals"

Turing, in his dissertation (1938) which was later published as "Systems of Logic Based on Ordinals" (1939), introduced the idea of a nonclassical computing machine implementing uncomputable functions. Having called these new kinds of machines "o-machines" (an abbreviation for "oracle machines"), Turing intended to explore the possibility of escaping the effects of Gödel's incompleteness theorems by

means of transfinite[8] sequences of formal systems and utilizing "ordinal logics"[9]. Gödel's theorem shows that, as it was stated before in Section 2.1, every system of logic that contains arithmetic is incomplete, i.e., it is impossible to obtain a complete system of logic containing arithmetic. However, it is still possible to obtain a logical system which is more complete than other systems of logic by adding new set of axioms to the system. This makes possible not only to prove statements which are intuitively true but unprovable in the original system but also to create stronger and stronger logics of a more complete and extended system.

Turing's purpose in using ordinal logics was to establish a non-constructive system where steps in a proof of the system would be mechanical as well as intuitive. Gödel's theorems showed that a constructive system necessitates intuition and ingenuity and it is not possible to eliminate the necessity of intuition from formal logic. Therefore, it is reasonable to turn instead to a nonconstructive system of logic. Ordinal logics provide examples of such a system (Feferman 1994, p.118). By a complete ordinal logic, Turing expected to prove any theorem of a formal system by using mathematical reasoning combining "intuition" and "ingenuity" with mechanical steps. In his investigation, Turing introduced the key notion of "oracle" which was capable of supplying answer to one particular unsolvable problem: "Let us suppose that we are supplied with some unspecified means of solving number-theoretic problems; a kind of oracle as it were." (Turing 1939, p.166)

[8] Transfinite numbers are also known as infinite numbers.
[9] Turing's aim was to extend logical systems by including each unprovable statement as a new rule and thus create new systems recursively. Each such new system would infinitely also include unprovable statements, but each system would also be 'more complete' than the previous, capable of proving more statements. Turing called this "ordinal logic" (Copeland, 2004).

3.1.2 The Structure of O-Machines

O-machines can roughly be defined as Turing machines with increased power and thus enabled to return by means of its oracle the corresponding values of functions that are not Turing Machine computable. The oracle helps to answer some of the questions which are beyond the capability of any ordinary Turing machine.

Although Turing did not mention the underlying mechanism for the structure of o-machines, we can at least have an idea about what could be the organization of an o-machine could be. First, all primitive operations that help to perform actions are supervised by a black box, which is referred to as oracle (Figure 5).

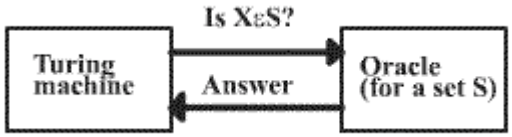

Figure 5: A Turing machine equipped with an oracle (From Beckman 1980, p.246**)**

The machine with its oracle, briefly, gives answers to the questions which are beyond Turing machine computability. For the given argument, the oracle returns the value of the characteristic function by indicating whether or not the given number is in the prescribed set (Beckman, 1980). On the o-machine, there is a special state called *call state.* Furthermore, a special *marker symbol* identifies the start and the end of input string. If the instruction in the machine program orders to do so, the machine enters into the call state. The call state executes one of the primitive actions p of the machine by sending the input query to the oracle. The oracle returns the value of the function by either entering into *1-state* or *0-state*. The machine will enter into 1-state if the call to the p has its corresponding value in the input string or will enter into 0-state otherwise.

One particular o-machine has two parts: a classical part as implemented by a Universal Turing Machine and a non-classical part which calculates the values of Turing's uncomputable halting function H(x, y). For that reason, they can also behave like halting function machines (Copeland, 1998a). (Halting function machines can be considered as machines solving the halting problem defined in Section 2.3.)

Now consider an example regarding how a Turing machine equipped with an oracle will make a computation. As it was stated in the earlier sections, it is possible to define the action of a Turing machine with a set of instructions in the form of quintuples. This mechanism is also valid for an o-machine by describing such an additional quintuple of a particular set S which maybe computable or not, but has primarily importance when it is uncomputable (Davis 1958, p.71). The quintuple can be determined as follows: (q_i, S_j, X, q_k, q_l) where S_j is a symbol from the alphabet, q_i, q_k, q_l are states and X is the direction of the tape (R for Right, L for Left and N for do nothing). This can be interpreted that Turing Machine Z will carry out this action: when Z is in state q_i (or internal configuration q_i) scanning symbol S_j, it will either enter the state q_k (internal configuration q_k) or q_l (internal configuration q_l) depending on the content *C* of the tape. If the number of 1's on the tape is in the set S it will enter state q_k; if not, it will enter state q_l. This decision is made with reference to the oracle. In Turing's view, the moves of the o-machine can be implemented by a table as usual, as in the case of ordinary Turing machines except in the case of the moves from a certain internal configuration q_i.

Depending on this structure, Beckman (1980, pp. 247-248) gives an example of an o-machine which is designed to answer the question whether there are n consecutive 7's in the infinite decimal expansion of π. Set S is the set of numbers *n*

having this property. The o-machine Z_0 (equal to Turing machine equipped with an oracle) will decide the membership in S. The formal description of this computation is presented in Appendix B.

One of the quintuples in the configuration $q_2 1 N q_3 q_4$ indicates that when Z_0 is in state q_2 and scanning 1, it will either enter state q_3 or q_4 depending on the content *c* of the tape. If *c* includes consecutive n 7's in the decimal expansion of π then it will enter the state q_3, otherwise it enters state q_4. After that, Z_0 with the input representation of number n will give the answer 0 if $n \in S$ or the answer 1 if $n \notin S$. It can easily be seen that the oracle stands in the core of the computation where the rest is trivial. This is because when the set of functions are computable, the oracle adds nothing new to the computation power of the Turing machine. However, if the sets of functions are not computable then any machine with its oracle can do tasks which cannot be executed by a Turing machine.

The emergence of o-machines, in spite of being hypothetical, provided a valuable tool for extending and analyzing the concept of computation and had an effect on the subject of general recursiveness theory. Turing tried to represent the behavior of o-machines during computation within a type of problem which is not number-theoretic. Number-theoretic functions are functions whose domain is the set of positive integers, i.e., natural numbers. By definition, the o-machine will have one of its fundamental processes as solving a given number-theoretic problem. As the statement "every number-theoretic function $\Phi(a_1 \ldots . a_n)$ for which there is an algorithm … is Turing computable" (Kleene 1994, p.26) implies, number-theoretic functions are computable in some special cases. Turing tried to establish the impossibility of constructing an o-machine which will determine whether an o-

machine given the description of another o-machine is o-circle free.[10] He proved that the circle freedom problem of o-machines is not number- theoretic and hence is not solvable by an o-machine. As o-machines are formed with the help of oracle this problem is also unsolvable by the oracle itself. Actually, one of the important consequences of this process was to identify the existence of degrees or classes of unsolvability and eventually this represented the idea of transforming computability (or unsolvability) from an *absolute* notion into a *relative* one. Furthermore, although Turing did not mention any other function than these number-theoretic functions, in the later publications "the notion of an o-machine has been widened to include fundamental processes that produce values of any function on integers that is not Turing-machine computable." (Copeland 2000b, p.19)

By the notion of o-machines, Turing opened a new era of investigation in mathematical logic. As a mathematical tool, oracles help to explore the mathematics of uncomputability. Turing, however, did not reveal what could be the applications of these oracles in his dissertation: "We shall not go any further into the nature of this oracle apart from saying that it cannot be a machine." (1939, p.167) This remark was the reason why these oracles stayed as a disregarded notion. It prevented many other mathematicians from going into further details (Feferman, 1994).

Briefly, the notion of o-machines introduced what could be done beyond pure mechanical processes. O-machines compute functions that are considered to be uncomputable by Turing Machines. However, there still exist some other functions that are not computable by o-machines, i.e. no o-machine is sufficient to compute all functions. The classical part of an o-machine is formed of finite set of states of

[10] A machine is o-circle free if it computes a real number in the interval 0 and 1 in the sense that it successively prints out the whole binary expansion (i.e., infinite sequence of 0's and 1's) of that number.

transitions whereas non-classical oracle part is composed of possibly infinite set of number of transitions. As Cantor proved by his well-known diagonalization process, the set of real numbers is uncountable. However, Turing, by using Cantor's process, showed that the set of computable real numbers is countable which entails the fact that there are only countably infinitely many Turing machines. Actually, computable real numbers constitute a smaller portion of all real numbers since most real numbers are uncomputable. That is, o-machines cannot compute all real numbers. This entails the fact that o-machines can meet the whole concept of uncomputability because computability, as it is stated before, is a relative notion, not an absolute one. Since there are degrees of computability, any function which is uncomputable for one can nevertheless be computable for another machine.

Now, I will describe a new model of computation regarding Turing machines which emerged from the historical context of o-machines. This model includes different types of theoretical machines which can make computation beyond the bounds of Turing-computability.

3.2 Hypercomputation

Besides o-machines, there are also some other machines proposed to perform computations which are proved to be impossible by the Universal Turing machine. The study of these kinds of machines is called hypercomputation and various sorts of these machines are called hypercomputers (or hypermachines to be more precise). Hypercomputers embrace all machinery, physical or abstract, which are capable of carrying out tasks (such as computing uncomputable functions) more powerfully than Universal Turing machine and beyond the Turing Limit.[11]

[11] Turing Limit is the "level of Turing machines and their equivalents" (Bringsjord 2003, p.xxv).

The term 'hypercomputation' was first used in Copeland and Proudfoot's speculative article entitled "Alan Turing's Forgotten Ideas in Computer Science" (1999a). Yet, it would not be correct to conclude that the field of study emerged with them. Indeed, the concept was not new. It had a substantial background. All that Copeland and Proudfoot did was to name all the studies transcending conventional models of computation. Actually, oracle machines, which were explained in the previous section, were probably the earliest example of hypercomputers.

Just as there are infinite numbers of Turing Machines capable of carrying out computation, there can be infinite numbers of hypercomputers. In the literature, many hypercomputers and models of hypercomputation are featured. Before advancing on explaining how some particular machines are implemented and the points at which they are separated from each other, let us emphasize the common properties of different kinds of hypercomputers. This will also make it easier to understand what makes a hypercomputer a hypercomputer.

3.2.1 Common Characteristics of Hypercomputers

Since hypercomputers can do more than standard models of Universal Turing machine can, it is true that they have additional power. Copeland (2002) classifies two ways of defining this additional power. The first of his arguments about this classification is as follows:

> The additional power of a hypercomputer may arise because the machine possesses, among its repertoire of fundamental operations, one or more operations that no human being unaided by machinery can perform. (Copeland 2002, p.462)

An example is o-machines. Additional power of an o-machine is due to its additional device (i.e., black box) which is referred to as oracle. This new fundamental operation of o-machines provides them with capability to solve different kinds of

problems than ordinary Turing machines. The second feature common to all hypercomputers is about their formation. All hypercomputers have some properties which extend the standard view of computability based on Turing machines. However, hypercomputers do not have to possess these features at the same time; indeed, either the one below or the one we mentioned above is sufficient for hypercomputers:

> Or the additional power may arise because certain of the restrictions customarily imposed on the human computer[12] are absent in the case of hypercomputer – for example the restrictions that data take the form of symbols on paper, that all data be supplied in advance of the computation, and the rules followed by the computer remain fixed for the duration of computation. (Copeland 2002, p.462)

Recall that all these restrictions are due to the standard view of Turing-machine-computability. Actually, it is possible to list specific features of any Turing machines by its definition. Turing machines are

- *discrete state* machines,
- strictly *deterministic* and *closed,* and
- *formal systems*

When a machine is freed from one (or more) of the restrictions defined by a Turing machine (such as from having a discrete state configuration) it will be possible to focus on production of more powerful computational devices, i.e., hypercomputers. First, let us investigate the property of "being a discrete state machine" of Turing machines and indicate which hypercomputers freed from this restriction are featured in the literature.

[12] By the notion of Turing machines, Turing's concern was to explore the theoretical limits of what a human being could compute.

3.2.1.1 Analog Computation

Turing machines are discrete systems. Computation in Turing sense can always be expressed in terms of sequential and serial steps in fixed length. The whole operation can easily be transformed into binary representation. Additionally, the concept of stored program which was introduced by Universal Turing machine facilitates executing the appropriate program to perform all tasks that can be carried out and then store the instructions into its writable memory. Therefore, it is no wonder that Turing machines can be considered as a foundation for development of digital computers. However, not all systems are discrete. There are continuous systems as well, which Turing machines cannot model for instance; analog systems have different characteristics which make them distinguishable from digital systems. In analog computation, models are "defined on a continuos phase space (e.g. where the variables x may assume analog values)" (Siegelmann 2003, p.106) contrary to the digital model having discrete phase space. Physical system in analog computational models is characterized by real constants instead of constants in principle accessible to the programmer (ibid.). For digital computers, the output is completely defined by the input at the beginning of computation, and in principle the measurement which can be done by an outside observer will be the same when it is repeated as long as it is desired. However, for analog systems, although the output is determined by input state as in the case of digital systems, the physical dynamics has a characteristic which is not observable and therefore not accessible by the observer. Additionally, the dynamics of the physical system is said to be "locally continuous" (ibid.). That is, statements of the forms, such as "if $x > 0$" follow one computation or "if $x < 0$" follow another computation, is not allowed in analog systems. Furthermore, idealized

analog computers can operate on real numbers in polynomial time whereas digital computers operate on computable numbers. All these main properties distinguishing analog models from discrete models allows beyond Turing machine power.

In the literature there have been analog models which are more powerful than Turing machines such as Analog Recurrent Neural Network Model (Siegelmann and Sontag, 1994) , Analog X-machines (Stannett, 1990) and Scarpellini type machines. A summary of their origins and their capabilities beyond Turing machines are provided in Table 1 below.

3.2.1.2 Non-Deterministic Computation

All Turing machines are deterministic and therefore are closed to outside effects and systems. However, they can also be allowed to act non-deterministically. During computation, non-deterministic Turing machine is allowed to execute one of the possible "branches" of instructions. Whereas a (deterministic) Turing machine has only one option to follow which is determined before computation, a non-deterministic Turing machine can choose one of the possible branches. However, no increase in the computational power is obtained.

Any deterministic Turing machine can be programmed to simulate a non-deterministic one. This is also true for Probabilistic Turing machines (de Leeuw, Moore, Shannon, Shapiro 1956). When a probabilistic Turing machine is put into operation the machine chooses the action it will execute randomly among finitely many alternatives with an equal probability for every option. Nevertheless, this machine can only calculate computable functions. Therefore, like non-deterministic Turing machines, they do not have more computational power than ordinary Turing machines.

Table 1: The summary table of the origins and capabilities of analog hypermachines

Hypermachine	Origin	Capability beyond Turing computability
Analog Recurrent Neural Network Model (have the same structure with "Analog Chaotic Neural Nets Model")	Siegelmann and Sontag 1994	• Networks of neurons performing by analog means (i.e., continuous phase space, local continuity and real constants) in polynomial time. • Allow interconnection weights of networks to be irrational numbers. • Memory and processing units are coupled, not separated.
Analog X-machines	Stannett 1990	Involves asynchronous real-time concurrent computation where continuity of time is defined in the beginning
Scarpellini type machines	Scarpellini 1963 (in Copeland 2002b)	Allow analog performance which can generate functions for which the predicate is not decidable by the Turing machine where analog machine itself decides it

Having the same structure with Probabilistic Turing machines, the Stochastic Turing machines (Liskiewicz and Reischuk, 1997) can perform random moves and thus the output state is random. Just as Probabilistic Turing machines, they have a probability distribution function for possible outputs.

Turing, too, considered the notion of machines with random characteristics and he called them partial random machines. In his viewpoint, such machines could have properties similar to human beings:

> An interesting variant on the idea of a digital computer is a 'digital computer with a random element'. … Sometimes such a machine is described as having free will (though I would not use this phrase myself) (Turing 1950, p.438).

However, randomness characteristic is a subject of debate for free will. The issues concerning free will and hypercomputation will be discussed in Chapter 4. Nevertheless, partially random machines (Turing 1948, Copeland 2000) carrying out an infinite sequence of binary digits that is random is kind of hypermachine having the property of free will. Table 2 below summarizes the origins and capabilities beyond Turing capabilities of the machines mentioned:

Table 2: The summary table of the origins and capabilities of non-deterministic hypermachines

Hypermachine	Origin	Capability beyond Turing computability
Partially random machines	Turing 1948, Copeland 2000	Some actions are the outcome of random influences but the operation is otherwise determined, e.g., by a program
Probabilistic Turing machines	de Leeuw et al 1956	Choose randomly among finitely many alternatives

3.2.1.3 Informal and Infinite Computation

Turing machines are formal systems and finiteness characteristic is directly related to this property. In Turing's sense, for an operation to be performed effectively means to be executable in *finite* time, by *finite* number of steps and by using *finite* facility. For example he explains how computable numbers are calculated: "The 'computable' numbers may be described as the real numbers whose expressions as a decimal are calculable by finite means" (Turing 1937, p.116). Hypercomputers can be freed from finiteness characteristic by performing infinitely many steps of computation (even in finite time, at an accelerated rate). These hypercomputers provide a developed model for analyzing the theoretical capabilities

and limitations of super-task computation.[13] A very famous example of super-task is Thomson's lamp which gives us a contradiction:

> There are certain reading-lamps that have a button in the base. If the lamp is off and you press the button the lamp goes on, and if the lamp is on and you press the button the lamp goes off. So if the lamp was originally off, and you pressed the button an odd number of times, the lamp is on, and if you pressed the button an even number of times the lamp is off. Suppose now that the lamp is off, and I succeed in pressing the button an infinite number of times, perhaps making one jab in one minute, another jab in the next half-minute, and so on, according to Russell's recipe. After I have completed the whole infinite sequence of jabs, i.e., at the end of the two minutes, is the lamp on or off? It seems impossible to answer this question. It cannot be on, because I did not ever turn it on without at once turning it off. It cannot be off, because I did in the first place turn it on, and thereafter I never turned it off without at once turning it on. But the lamp must be either on or off. This is a contradiction. (Thomson 1954, p.5)

Thomson's lamp example focuses on the state of the device after the alleged completion of the super-task. Although it provides a good basis against the existence of such kind of machines, it is still possible to introduce theoretical hypercomputers which can carry out and complete infinitely many computational steps, such as infinite time Turing machines (Hamkins and Lewis, 1998). These machines are infinitely fast machines since they are able to complete infinitely many steps of computation in a finite amount of time by simply introducing a model of computability which extends the Turing machine model into transfinite ordinal time. Here, concerning the number of computational steps, infinite time Turing machines are infinitely long.

Turing machines do not accept any input after they start operating. The program of the machine provides all the input and instructions at the start of the computation. This makes the system unchangeable for any other implementation

[13] A task "whose completion involves carrying out all of an infinite number of subtasks" (Thomson 1954, p.2).

after starting. Sloman (1998) argues that this restriction affects only computing systems which have strict limitations on what they can do:

> …that there are limits to what a particular computing system can do, …[is] irrelevant to the problem of what sorts of intelligent mechanism can be designed: for all these theorems are relevant only to 'closed' systems. i.e., systems without means of communication with teacher, etc. (Sloman 1998, p.104)

Any system which is not bounded by these restrictions can exceed it. In our case, hypercomputation (such as the one which can accept input while operating) seems to be more "realistic" than the standard view of computing system since in the real world any operation is always open to real world constraints such as noise. Coupled Turing machines, proposed by Copeland (1997b, 2002a, 2002b), which accept input via an input channel while operating, are examples to these kinds of machines.

There are also some other models proposed to transcend the Turing limit in the literature. One of these models includes quantum mechanics principles which were discussed in Kieu (2002), Hogarth (1992), Stannet (2001). Quantum computation is a model for improving the standard view of computation by using quantum mechanics (Lloyd, 1998). The principle behind quantum computers is that they utilize quantum properties of particles to represent and structure data where the classical computer is inadequate. The challenging area of quantum computation is born with idea of simulating the quantum equations on a probabilistic computer (or on a probabilistic Turing machine) in an efficient way. Today, quantum computers are used to solve many problems fast, efficient, effectively regarding quantum phenomena (Kieu, 2002). New types of quantum computation are being introduced to the field with various advantages over classical computation (Ekert & Macchiavello, 1998). A sequence of studies in order to use quantum computation

was manifested by physicists and computer scientists. Very briefly, the most important applications of quantum computers reside in cryptography, Grover's algorithm (which is based on database search), Shor's algorithm (which is about efficient factorizing large numbers very rapidly) and effective quantum simulations (Benjamin & Ekert, 2006, Ekert & Macchiavello, 1998). However, there is still an ongoing debate about whether quantum computers can be programmed to do more than Turing machines; though it is a general view that they can do it faster. Kieu (2002) proposes an unconventional model of quantum computation (a quantum hypercomputer) whose quantum algorithm solves Hilbert's tenth problem[14], a problem which does not have a solution classically. A variant of a quantum computer is a Timed X-machine (Stannett, 2001a) within a general model of time (an extended version of analog X-machine), which also combines other models of computation, such as analog, discrete and hybrid.

As mentioned before, the capabilities extending the Turing machine computability lead to plausibility of the idea to enlarge the notion of computability. In the literature, there are other models of informal hypercomputers which recover from the intrinsic restrictions of standard models. Putnam and Gold machines, accelerating Turing machines, asynchronous networks of Turing machines, extended Turing machines, error prone Turing machines, and accumulator machines introduce different property or properties which distinguish them from ordinary Turing machines. Table 3 below summarizes the properties of these unconventional machines.

[14] Hilbert's tenth problem is a decision problem which says "given any polynomial equation with any number of unknowns and with integer coefficients: To devise a universal process according to which it can be determined by a finite number of operations whether the equation has integer solutions" (Kieu, 2002).

Table 3: The summary table of the origins and capabilities of infinite and informal hypermachines

Hypermachine	Origin	Capability beyond Turing computability
Putnam Gold Machines (also called "Trial and Error Machines")	Putnam 1965, Gold 1965	• Take the last output the machine produces as its result where the outputs are not certain or fully completed since the machine can "change its mind". • The output is produced in the limit • Allow for unlimited periods of time
Accelerating Turing machines (have the same structure with "Rapidly Accelerating Computer", "Zeus Machines")	Copeland 2002a, Stewart 1991, Boolos and Jeffrey 1974	Each primitive operation is not fixed in duration, i.e., infinite sequences of distinct acts are performed within a finite time (actually less than 2 moments of operating time) with an accelerating rate
Coupled Turing machines	Copeland 1997	Accept input via an input channel while operating
Infinite Time Turing machines	Hamkins and Lewis 1998	An infinitely fast computer which is able to complete infinitely many computational steps in a finite time by extending the concept of an ordinary Turing machine into the realm of transfinite ordinal time
Timed X-machines	Stannett 2001a	Allow systems to be modeled that use both discrete and continuous times for different phases of operations and thus combining quantum computation with standard and analog computation
Asynchronous Networks of Turing machines	Sloman 1996, Copeland and Sylvan 1999	Allow finite assembly of Turing machines which are not operating in synchrony
Extended Turing machines	Abramson 1971 (in Copeland 2002b)	Store a real number on a single square of its tape
Error Prone Turing machines	Ord 2002	Prints a different symbol to the one intended during computation defined by an error function . This error function can be manipulated to compute halting function.
Accumulator machines	Copeland 1997, 2002	Compute addition over real numbers (including any arbitrary pair) and other functions inaccessible to Turing machines

Despite the grouping of hypercompution into 3 different categories viz. analog, non-deterministic and, informal and infinite, it is not always correct to have such solid distinctions. This is due to the fact that some hypermachines may combine

different aspects such as being infinite and informal as well as being discrete and show temporal patterning at the same time. A typical example is Timed X-machine which is compatible with unified models of analog, quantum and discrete computation. This information gives us the idea to explore the different aspects of some hypermachines in respect to others. Some hypercomputers may involve temporal aspects while operating. This feature is categorized under temporal patterning. Additionally, some hypermachines are informal in the sense that they are Turing machines with an informal aspect involved in. Hence, a higher level of classification of hypercomputers is possible (Table 4).

Now I want to proceed with two examples of hypercomputers, namely accelerating Turing machines and Putnam-Gold machines, which will be related to the subject matter of this thesis. The reason for choosing these hypercomputers is to give detailed information about the principles of how a hypercomputer works. Furthermore, these hypercomputers will facilitate to understand the principles of the hypercomputer we propose in respect to its implications to the chosen theory of free will. The subject matter will be covered in the following chapter.

3.2.2 Accelerating Turing Machines

Standard Turing machines execute computation without a specified temporal patterning. Each primitive (or atomic) operation (such as “halt computation”, “move the square one cell right/left”) takes place without a reference to duration between each sequential step. Since Turing machines are highly idealized abstract devices, there is no problem with conceiving of the operation time between each step as one unit time.

Table 4: Different aspects of hypercomputers gathered together ("X" represents the existence of the shown aspect)

Hyper-machine	Discrete	Analog	Quantum	Non-deterministic	Informal	Infinite	Temporal patternig
Putnam Gold Machines	X						X
Accelerating Turing machines	X				X	X	X
Analog Recurrent Neural Network Model		X					X
Coupled Turing machines	X				X		
Infinite Time Turing machines	X				X	X	X
Analog X-machines		X					X
Timed X-machines	X	X	X				X
Asynchronous Networks of Turing machines	X				X		
Accumulator machines	X						
Extended Turing machines	X				X		
Scarpellini type machines		X					
Partially random machines	X			X		X	
Probabilistic Turing machines	X			X	X		
Error Prone Turing machines	X				X		

Therefore, Turing did not consider time as a critical factor for his machines. However, time can still be seen as a utilizable property in order to propose unconventional types of Turing machines. The common property of all these atypical machines is that they can be programmed to perform tasks which are beyond the capacity of an ordinary Turing machine, such as computing the halting function.

One of these machines is a Zeus machine which was described by Boolos and Jeffrey (1974, pp.14-15). Zeus can enumerate the set of natural numbers N in one "moment", by implementing first entry in 1/2, second entry 1/4, third entry 1/8 moments of operating time, and so on. Since

$$1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots\left(\frac{1}{2}\right)^{n}+\cdots$$

is a mathematical series which can be defined as follows:

$$\sum_{n=1}^{\infty}\left(\frac{1}{2}\right)^{n-1}=2$$

That is, at the end of two moments of operating time it enumerates natural numbers completely. Taken for granted that this series converges and has sum of two, thus, an infinite number of operations are completed in a finite time by working faster and faster. The crucial part of Zeus machines is that they operate in an accelerating manner.

Stewart (1991, pp.8-9) also proposed an infinitely fast computer called rapidly accelerating computer which can complete an infinite number of operations in one second. This computer, therefore, can compute functions which are not computable in Turing sense. A form of rapidly accelerating computer that can be described fully by a Turing machine is called accelerating Turing machine (Copeland 1998c, 2002a, 2002b). An accelerating Turing machine (ATM) executes the program on its tape at an accelerating rate, just as Zeus machines or rapidly accelerating computer, performing each atomic operation that the program calls for in half the time that was taken for the immediately preceding atomic operation.

Despite their similar working principles, accelerating Turing machines and Zeus machines differ in some respects. All ATMs are Zeus machines but not vice

versa. The concept of Zeus machines is more general than ATMs. ATMs are Turing machines with an accelerating manner; however, Zeus machines are *any* machines performing temporal patterning in an accelerating manner. There can be machines which exhibit patterns of Zeus machines. For instance, an o-machine that exhibits Boolos and Jeffrey temporal patterning is a kind of Zeus machine but not a (accelerating) Turing machine. Unlike ATMs, Zeus machines with their different temporal patterning are not Turing machines at all. This distinction is made clear in Copeland (1997b).

Now consider how an ATM which is programmed to simulate the behavior of a given Turing machine can compute the halting function. From section 2.3, the halting function is a function having either 0 or 1 as its value. An example accelerating Turing machine is composed of the alphabet [1,0]. The initial square or any specified square of the tape of ATM is inscribed for the display of output of computation. In the start of the computation, by default, the initial square has the value of 0. If the Turing machine halts on the input then the Turing machine scanning the input tape returns to the initial (or specified) square and change the value written there to 1. If the Turing machine does not halt, then ATM leaves this square unchanged and the scanner never returns to the initial square. (It is this unchanged square which is the result of computation.) Either way, the value of the halting function for this Turing machine is computed on its initial square by the end of two time units.

3.2.3 Putnam-Gold Machines

Putnam-Gold machines actually have the same underlying mechanism or hardware configuration as standard Turing machines. However, there are some differences in terms of the interpretation of the concept of computability. For a

Putnam-Gold[15] machine, the output for the entire computation is produced in the limit. The notion "computability in the limit" belongs to Gold (1965, p.28) who proposed an algorithm which can generate long sequence of guesses for an infinitely long decision procedure. The problem generated for this specific procedure will be solved in the limit, i.e., after a certain point in the sequence all the guesses for this output will be correct, having an entire sequence of same correct answer. Thus, the machine will be able to compute the function with correct value eventually. However, the crucial issue in that procedure is the point after which the machine will produce correct output. During this process, the output produced for this machine will not be certain or fully completed and stay undefined because the machine has the capability to change its mind after a finite number of times. In terms of computation, what the Turing machine does is a one-trial procedure. However, the Putnam-Gold machine follows a two-trial procedure: make a guess and change it in case it is necessary (Kugel, 2002).

Putnam (1965, p.49), independently from Gold, proposed this important aspect of that machine: the property of changing its mind. Putnam's machine will take the last output the machine produces as its result for each input contrary to ordinary Turing machine which takes the first output as its result. To implement this property, Putnam, in his paper, tried to modify the concept of decision procedure by allowing some significant changes. After the machine has made some mistakes (i.e., changed its mind several times) it will print out the correct value as output. However, this modified concept of procedure: there will not be a decision procedure to tell the computation has ended (unless until it is turned off by an external agent).

[15] Putnam-Gold machines are also called Trial and Error machines. This name implies the characteristic property of these machines, following Putnam (1965), which allow decision procedure by using a class of trial and error predicates.

Although its structure might seem unharnessable (i.e., unable to be used to produce useful power) at first sight, Putnam-Gold machines have several advantages when compared to ordinary Turing machines. First of all, these machines have all the machinery Turing machines have and can effectively do every task that any Turing machine can. Kugel argues that talking about the property of computing in the limit "resembles the way we compute the values of irrational numbers like $\sqrt{2}$ and π in the limit – getting closer and closer to the exact result at each step, but never getting its decimal expansion exactly right" (Kugel 2002). Furthermore, the property of changing its mind gives an idea about the formation of a property which is unfamiliar to traditional Turing machines: possibility of making mistakes. Actually, this is what Turing considered when he was talking about genuine intelligence (ibid.).

The relevancy of these hypercomputers (i.e., accelerating Turing machines and Putnam-Gold machines) to my study is their compliance with the working principles of the specific hypercomputer I propose. Briefly, the temporal patterning of accelerating Turing machines and capability of computation in the limit and existence of trial procedures which make possible to bring about the action or prevent it are the decisive reasons to choose them.

3.2.4 Computability in different senses

The consequence of the computation in accelerating Turing machine is very important because we have a Turing machine which can compute the halting function. On the other hand, according to the halting theorem halting function is not Turing-computable. It seems there is a contradiction here. Not for Copeland, however (Copeland, 2002a). He explains this problem by arguing that the halting theorem is a weaker proposition than is supposed. For him, there exist two senses of

computability: computability in the *internal sense* and computability in the *external sense*. In his view the internal sense of computability can be expressed as follows:

> A function is computable by a machine in the internal sense just in the case the machine can produce values from arguments (for all argument in the domain), halting once any value has been produced, where what counts as halting can be specified in terms of features internal to the machine and without reference to the behaviour of some device or system - e.g. a clock- that is external to the machine. (Copeland 2002a, p.484, also Copeland 1998, 2002b)

And computability in the external sense can be defined as follows:

> A function is computable by a machine in the external sense just in case the machine can produce values from arguments (for all arguments in the domain), displaying each value at a designated location some pre-specified number of moments after the corresponding argument is presented. (Copeland 2002a, p.484, also Copeland 1998, 2002b)

The most important distinction between different senses of computability is the halting of the machine. In the internal sense the machine halts and numerous behaviors on the part of the machine can be counted as halting, such as the cessation of the activity or emitting a hoot. In the external sense, the machine may or may not halt once the value has been displayed although the value is prespecified before. It is in the internal sense, not the external sense, that halting function is not computable by a Turing machine. That is why halting theorem is a weaker proposition than is thought.

An accelerating Turing machine performing its operations in the given structure above can solve halting theorem in the external sense since the value can be read from the output tape at a designated (specified) square. Or an o-machine with an accelerating Turing machine as its oracle will solve the halting problem in the internal sense since once it produces the corresponding value of the function (i.e., 0 or 1) it halts without reference to any other machine. Indeed, any machine, which is capable of computing a function in the external sense, can be converted into a

machine which has the property of computation in the internal sense by simply adding some equipment such as a clock to the machine (Copeland 1998a, Copeland 2002a). (However, a Turing machine with extra equipment assembled to it is not a Turing machine anymore). A new machine organized by putting together an accelerating Turing machine with a clock is an example to this case. Furthermore, when the necessary conditions in the internal senses are lifted, a Turing machine will be able to make computations with functions which are not Turing computable.

I believe the distinction between the computability in the internal sense and external senses strengthens the logical possibility of hypercomputation since computability notion can be extended by using this distinction which makes possible organize machines showing capabilities of hypercomputers. The idea behind the distinction between external and internal senses also provide a basis for the degrees of computability, any function which is uncomputable for one can nevertheless be computable for another machine. However, as it was stated before for description of o-machines, the whole concept of hypercomputation is very wide and no o-machine is sufficient to compute all functions. As there are infinitely many Turing machines, there are infinitely many hypercomputers as well. Moreover, hypercomputers presented in the literature is only a small subset of logically possible hypercomputers. The idea of pan-computability, which compromises the computability in Turing sense and hypercomputation, is demonstrated in Figure 6**.** Pan-computability is a general term which covers all different forms of computability, i.e., standard views (computability in Turing sense) and non-standard views (hyperomputability).

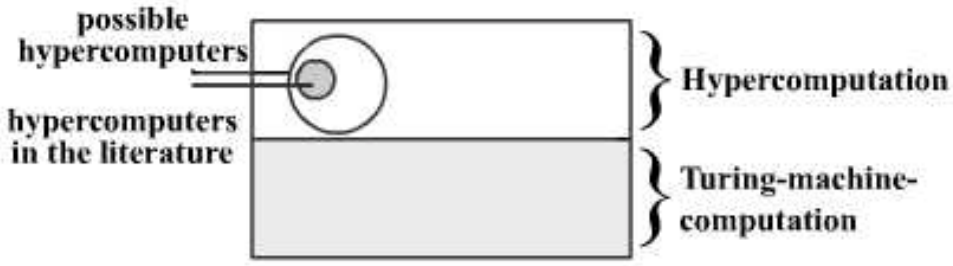

Figure 6: Pan-computability and place of hypercomputers

Undoubtedly, hypercomputation is pure abstraction; however, it is not a concept which can manage the impossible, such as "computing the uncomputable." Yet, any Turing machine with a specific organization (such as a Turing machine showing properties of hypercomputers, e.g., ATMs) can compute functions in the external sense which cannot be computed by Turing machines in the internal sense. That is, since there is no contradiction with the results of Turing-machine-computation, hypercomputation is not deprived of a concrete theoretical basis.

To sum up, its theoretical viability validate the implications of hypercomputability. In the following chapter, I will discuss one of these implications on a specific issue called the agent causation theory of free will.

CHAPTER 4

HYPERCOMPUTATION AND FREE WILL

The study of hypercomputation has emerged from unsolvable mathematical problems. However, our interest in this field should not be limited to mathematical phenomena. Other research areas can also benefit the advantages of this distinctive idea. In this chapter, I will try to establish a relationship between hypercomputation and agent causation theory of free will. To accomplish this relationship, I will try to explore whether free will is computable from the point of view of classical computation or not. Then, I will propose some hypercomputational devices which will assist in building up this relationship.

If hypercomputation provides a theoretical explanation to the uncomputable phenomena (such as the halting problem in mathematics, as a starting point), hypercomputational theory of mind can give a "more flexible, more independent, more adaptable and more human" (Kugel, 2002) structure. Actually, this opinion is related with a broader sense of understanding of hypercomputation. In a broader sense, it is possible to question every assumption the Turing machine view of computability is based on. Hence, we can reach a new theory of mind regarding issues depending on beyond Turing-machine-computability which is analogous to Computational Theory of Mind (CTM) depending on standard view of Turing-

machine-computability. Here, we can find implications not only on mathematics but also on philosophy, computer science, and physics. In a narrower sense, hypercomputational investigations on computation of functions which are proved to be uncomputable in Turing sense may have implication on just mathematics. Broader sense view seems to be the answer to the arguments of opponents, such that hypercomputation can not only solve algorithmic aspects (such as thinking, since thinking can be reduced into clear and determined set of rules, i.e., computational procedures) by its classical part but also provides a basis for non-algorithmic processes by its non-classical part. Thus, some philosophical issues (since our concern is the philosophical aspects only), such as qualia, free will or consciousness, which have been attributed to be problematic by opponents of CTM may be better understood.

As is it stated in Section 2.5, the portion of human behavior which can effectively be simulated by a computer (i.e., computationally) is only a small extent of all human behavior. On the other hand, hypercomputational models stand to be explored as well as the other portion of human behaviors. That is, the notion of computer behaviors which can be simulated effectively can possibly be extended to cover all behaviors. We know this idea is sound in the context of CTM since it does not require a new scheme to our understanding of human behaviors. It just needs an extension from standard models of computation to unconventional models, viz. hypercomputation. However, Bringsjord and Zenzen (2003) who acknowledge a Turing limit which separates the notions of computability and hypercomputability describe another scheme. They represent, instead of a simple two-folded computability and hypercomputability distinction, an additional space for phenomena which are not capturable in any third-person scheme. The third-person scheme part

includes things that human beings do but that cannot be described in any symbolic system. Persons have powers beyond the reach of mere computation. For them, the so-called super minds exist as an intersection of all three spaces (Figure 7). The arguments of this hypothesis can successfully be used against CTM. It is compatible with Searle's Chinese room argument: we can find parallels since both discuss on the uncomputable aspects of cognition.

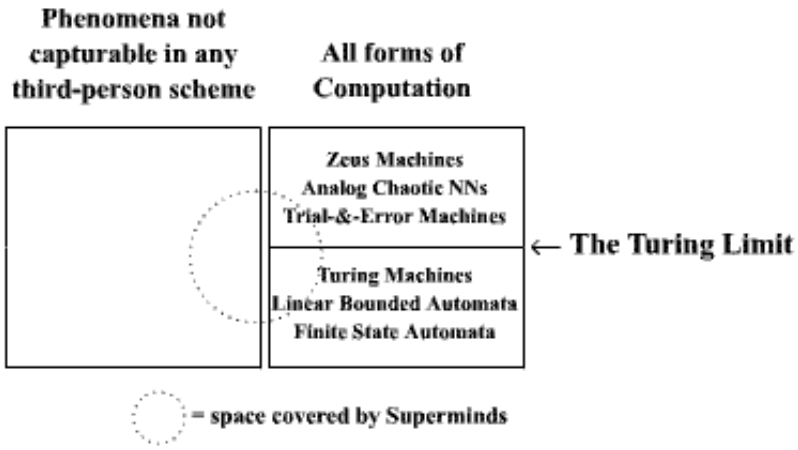

Figure 7: Superminds include three parts of spaces (From Bringsjord & Zenzen, 2003)

In the following section I will discuss how to provide a basis for the extension of standard models. This extension can be used for explaining certain issues in human behaviors. However, before explaining how to extend standard models, it is necessary to answer the question why to extend them. If we provide a theoretical basis for this extension, then we can use it on exploring different aspects of mind.

4.1 Penrose on Computability

In order to support my view it is first necessary focus on Roger Penrose's arguments. The relevancy of Penrose's arguments to my thesis is that he gives a detailed account of arguments against CTM. Thus, he assists to extend the computational model. Here, my view is two folded: First, it must be demonstrated that uncomputable (or non-computable) phenomena exists so that mind involves ingredients which cannot be explained by mere computation. I will focus on subject matter which is presented on books of Penrose entitled *Shadows of the Mind: A Search for the Missing Science of Consciousness* (1994) and its forerunner *Emperor's New Mind* (1989). Penrose expressed different point of views concerning the relation between computation and conscious thinking as general statements below:

A. All thinking is computation: in particular, feelings of conscious awareness are evoked merely by the carrying out of appropriate computations.
B. Awareness is a feature of the brain's physical action; and whereas any physical action can be simulated computationally, computational simulation cannot by itself evoke awareness.
C. Appropriate physical action of the brain evokes awareness, but this physical action cannot even be properly simulated computationally.
D. Awareness cannot be explained by physical, computational, or any other scientific terms. (Penrose 1994, p.12)

Penrose's own point of view is (C). (A) can be expressed as computationalism or Strong Artificial Intelligence thesis of which the assumptions are discussed in the scope of CTM. The (possible) existence of non-algorithmic mental states or apparently irreducible aspects or elements of some mental states such as qualia or free will can discredit the effect of this thesis. (B) is called Weak Artificial Intelligence thesis and was supported by (Searle in 1990a; 1990b). For Searle, according to weak Artificial Intelligence, all what computers can do is simulating brain processes computationally so that we can find helpful clues in studying mind.

(D) is a thesis which refutes any kind of physicalism at all and regards mind as something which cannot be explained in any scientific terms but only in religious mysticism.

Penrose, in his book taking consciousness as an example (or base) for explaining uncomputable phenomena, tries to show that our conscious mentality cannot be understood in terms of computational models. Accordingly, the mind must indeed be something that cannot be described in any kind of computational terms. Penrose uses, unlike Searle, Gödel's theorem which asserts that in arithmetic there are propositions which cannot be proved or disproved within the system. In other words, any formal system containing arithmetic is subject to the limitation of what has been called incompleteness.

Penrose shows that mathematical thought (or insight) is not mechanical by deducing a conclusion so-called "Gödel-Turing incompleteness theorem" G, which states "human mathematicians are not using a knowably sound algorithm in order to ascertain mathematical truth." (Penrose 1994, p.76) Thus, if human reasoning is capturable by a formal system F which is sound[16], then F cannot be used to ascertain the truth of the true statement G(F). That is, it will not be possible to see that whether F is consistent or not. As the mathematical insight depends on consciousness, consciousness can enable one to make certain the truth of a statement in a way the no algorithm could. Thus, Penrose concludes that human understanding and insight cannot be reduced to a set of computational rules. He claims that we perform non-computational actions when we consciously understand. In order to show that, he combines mysteries (or paradoxes) in quantum theory and Einstein's gravitational theories and concludes that classical physics is insufficient to get some insight into

[16] F is sound since it were unsound, then it would be falsifiable.

how the brain works. Therefore, it is necessary to look "outside the known physics" so as to find the basis of non-computable actions[17].

Penrose tries to explain uncomputability on a materialist or physicalist basis. Being a physicist, he argues that science is insufficient today and therefore he asserts the necessity of a new physics to explain brain actions. To do that, he advocates the idea of forming a bridge between quantum and classical physics.

Penrose's Gödelian challenge to CTM is only one of the arguments against this theory. There are also, of course, counter views against Penrose's arguments such as Bringsjord and Xiao (1997), Chalmers (1996), Feferman (1996), Klein (1996), and Thagard (1996). Thagard, for instance, rejects the idea that Penrose has shown the mathematical knowledge is not computational. In order to construct a full cognitive model of human mathematician, Thagard, proposes a cognitive model called CAM (*Cognitive Arithmetic Model*) which includes a full range of representations and processes (1996, p.177). However, this task will be a difficult one since constructing a Turing machine which is equivalent to CAM will include a full set of algorithms which nobody will be able to understand. In case it is possible to discover a Turing machine equivalent of CAM, then there will be no way to show CAM is sound or consistent. Even if CAM is sound, Thagard discusses, it will computationally be very difficult to deal with or find solution to show that it is sound. CAM is not using a knowably sound algorithm, as Penrose's G, it is not different than a human mathematician. Since Penrose has not shown an example that a human mathematician can do which a computer cannot, computational models of mind can deny his arguments. Moreover, Thagard denies Penrose's presuming

[17] Although Penrose does not use the term, this new physics should undoubtedly involve hypercomputational components.

computational model as a knowably sound system and proposes to use dynamic systems to expand and add to computational representational system rather than abandoning it (ibid.).

I will not cover all the other responses to Penrose since it can take us out of context. My point of view is to provide a basis for a broader concept which encompasses Penrose's arguments. Penrose asserts a proof that CTM is impossible. Despite the fact that I am using Penrose's arguments against CTM, my approach is not to refute it. On the contrary, I try to show that CTM can be extended and this extension refers to a hypercomputational model. Moreover, this extension can be utilized for a specific theory of free will. Computability theory of today emerges from the idea of Turing machines in 1936. If we accept the machines with more computational power and free Turing machines from restrictions necessary to make such current impossible computations, then it is plausible to accept the possible existence of new kinds of machines (i.e., hypercomputers). Thus, our new goal is to put into use the possibility of hypercomputers (such as analog hypercomputers, quantum hypercomputers, accelerating Turing machines, etc.) which can do tasks that no Turing machine can execute by extending the concept of Turing-computability. Here, I come to my second view. Then, it might be possible to use this new kind of theory of mind (i.e., "Hypercomputational Theory of Mind[18]") to explain the mental faculties which cannot be computed. Hence, it is possible to explore new study topics and find examples from philosophy such as the agent causation theory of free will where the insufficiencies of the CTM may be supplemented by this new theory. My idea is to focus on this specific topic of agent causation but not into this new theory of mind since a potential hypercomputational theory of mind will cover

[18] This term is used in several references such as (Ord, 2002).

more than the scope of this thesis. Before getting into details of this specific topic, it is necessary to discuss the issues regarding free will.

4.2 An Overview of Free Will

Free will can be defined, roughly, as the capacity of rational agents to choose a course of action from among various alternatives. The problems of free will have been subject to considerable and ongoing debate among philosophers. The core of the contemporary debates about free will can be discussed under these four intimately related questions, as Kane (1996) puts forward:

> The Compatibility Question: Is free will compatible with determinism?
> The Significance Question: Why do we, or should we, want to possess a free will that is incompatible with determinism? Is it a kind of freedom "worth wanting"… and, if so, why?
> The Intelligibility Question: Can we make sense of freedom or free will that is incompatible with determinism? Is such a freedom coherent or intelligible? Or is it, as many critics claim, essentially mysterious and terminally obscure?
> The Existence Question: Does such a freedom actually exist in natural order, and if so, where?

Actually, Kane (1996) groups these four questions into two: first two, i.e., compatibility and significance questions, can be answered together, whereas intelligibility and existence questions form another group. These questions demonstrate the debate for free will has been mostly gathered around the concept of determinism. Determinism is commonly understood as the doctrine that every event has a cause and "everything that happens is necessitated by what has already gone before, in such a way that nothing can happen otherwise than it does" (Butterfield, 1998). It is possible to assert a theory to be deterministic if and only if any two of its models that agree at a time *t* on the state of their objects, also agree at all times future to *t*. (ibid.) The first two of the proposed questions above try to find a basis for a free

will which is compatible with determinism and the other two imply incompatibilism with determinism.

My motivation, by displaying these questions, is not to endeavor to answer them. These questions can give us a general view about the ongoing discussions regarding free will. I will not attempt to take side for the good of one of these questions. However, I will assume that intelligibility question is significant for constructing a hypercomputational approach for this thesis project since the agency theory, in order to obtain a peculiar idea of free will, complies with incompatibility. Before constructing this approach, it is time to investigate the relationship between determinism and the concept of computability.

4.2.1 Determinism vs. Computability

Is free will computational or is it possible to identify free will in a computational structure? It will be presented in the following section that the most important question of free will problem is whether free will is compatible with determinism or not. However, in order to obtain a hypercomputational point of view of free will we have to start by exploring the relationship between determinism and computation. This relationship will help us to answer the question of whether free will is computational or not.

In order to maintain a scientific basis which can provide a rational account for the solution of free will problem, it is undoubtedly necessary to look into scientific principles regarding determinism. As it was stated before, determinist principles are mostly discussed under the theories of classical physics. However, when atomic or subatomic particles are introduced, quantum theory is put forward to be a more fundamental theory which can replace classical physical theories (Encyclopædia

Britannica, 2006). The most important property of quantum physics is its (indeterministic) predictions of observations in terms of probabilities.

A first insight for exploring the relation between determinism and computability implies it is trivial to say any deterministic model is computable by nature. As it was stated in Section 2.2.1, any effective procedure is deterministic in nature since it yields the same result if it is repeated by going through the same starting conditions and same steps where each step is precisely defined. Remember that effective procedure means what a Turing machine carries out and computation means actions of a Turing machine. Thus, our view of computation is said to be deterministic from the standpoint of Turing machines. We can find straightforward examples coming from current physics, most of which are proved to be correct in the mechanistic world of Newtonian physics. In this world, for instance, if the positions, velocities and masses of particles are known (together with all of the forces acting on them), then the positions, velocities (and masses) of particles can mathematically be determined for all later times. A well-known example is the elastic collision of two billiard balls where physical behaviors of the balls can be completely determined for all other times in the future as well as in the past if the velocities and positions are known. However, this should not take us to a generalization which asserts classical physics to be deterministic and computable and quantum physics to be indeterministic. This separation is not scientifically true. First of all, contrary to suppositions, much of the Newtonian physics is not deterministic. Determinism, even in the case of Laplace's paradigm of point-masses influenced by their gravitational attractions (as it is described by Newton's law of gravitation) hold for local, very short intervals of time and break down in global cases (Butterfield, 1998). Second, it is possible to explain quantum theory as being deterministic as the asserted

indeterminism of quantum theory is very controversial and not proved. A deterministic interpretation of elementary quantum theory is entirely coherent and quite possible despite it was alleged to be impossible (as it was claimed in 1930s by the co-founders of the quantum theory) (ibid.).

As it was shown before, halting problem (see Section 2.4) is unsolvable (i.e., uncomputable) in an entirely deterministic system (i.e., Turing machines). Furthermore, there may be cases from physics (real or simulation) when the model is deterministic but is not computable. Penrose (1997, p. 119) gives an example of such situation. In his example, Penrose refers to a "toy universe", which is a model reflecting certain features of the Universe. In this toy universe, different states of *polyomino sets* are defined deterministically, according to a certain sets of precise rules. A polyomino is a polyform with "collection of squares all stuck together along various edges to form some plane shape" (ibid.). Some examples of polyomino sets are represented in Figure 8.

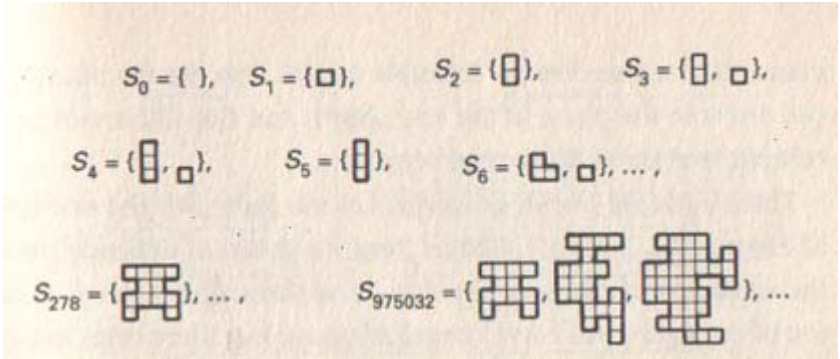

Figure 8: Some examples of polyominoes (From Penrose 1997, p. 119)

For this model, two separate finite sets of polyominoes define the state of universe at any moment. Depending on an exact rule, a given polyominoe set will tile to whole plane by using the polyominoes of that set. The problem, here, is that if it is possible to cover the entire plane without gaps or overlaps by using only the polyominoes of

the given set. However, there is no computer action which can simulate the evolution of this universe since we can never know when a polyomino set will tile the plane through a computational decision procedure. Thus this toy model universe is not computable despite the fact that it is deterministic. What this example shows that determinism and computability are different things: Determinism does not bring about computability.

Based on these results, as we accept free will to be incompatible with determinism, Turing machines or any other computational devices, due to their discrete and deterministic structures, cannot simulate free will. Despite the assertions of compatibilist accounts, free will might have uncomputable aspects even in a world where it is claimed to be entirely deterministic. Other attempts to simulate free will computationally cannot be successful as well. For instance, probabilistic Turing machines (refer to section 3.2.1), which were introduced by Leeuw et al (1955), do not differ from ordinary Turing machines but can "make independent random equiprobable choices, i.e., they can at any moment "to toss a coin" (Freivalds, 1999). However, it is shown that every function which can be computed by a probabilistic machine is also can be computed by a deterministic machine. Thus, probabilistic Turing machines cannot be adequate for simulating free will not only because they are not capable of carrying out tasks more than their deterministic counterparts but also probability does not yield free will, although it might only be one of its aspects. This is also valid for partial random machines, which were already introduced by Turing to have capabilities of showing some aspects of free will. Nevertheless, some examples of hypercomputers are also possible to be modified to compute beyond what Turing machines can do. (For examples, see (Ordy, 2002) and (Siegelmann,

2003).) That is, the free will issue can be explained better with hypercomputers, not with any kind of machines showing capabilities in the bounds of Turing machines.

Concerning the relation between determinism and computability, which demonstrates that they are different things, it is not possible to say free will is computational. In case we accept free will is compatible with determinism, it does not definitely reveal that it is computational. Likewise, if it is not compatible with determinism, then, due to the intrinsic features of Turing machines, it is still not computable. Consequently, as we free ourselves from the restriction of determinism defined by a Turing machines, then it is possible to focus on more powerful devices. Free will involves non-computational aspects and thus requires a hypercomputational approach for understanding it. Now, let us proceed with what compatibility and incompatibility represent.

4.2.2 Compatibilist accounts

If determinism is true, then the question regarding free will will be whether free will is compatible or incompatible with determinism. The compatibilist approach points out that free will is compatible with determinism since freedom is to choose or act freely when one is not constrained by external factors (such as somebody forcing you to do something) although one's actions or choices have been determined before. Determinism implies finite number of alternatives, which can be enumerated. For compatibilists, the existence of determinism does not rule out the existence of free will. This is because of two main features of free will: First of all, human beings choose what to do. "It is up to us" what we do since we can choose from an array of alternative possibilities. Since it is in our hands what we choose, we "could have done otherwise" by our free decisions or could avoid from doing it. Secondly, we are

the doers of our choices. We are the origin or the source of what we do and how we act. It is in us, not in something else which we do not have control. In sum, these features entail a power or ability to do what we will, desire or choose and an absence of constraints or impediments preventing us from what we will, desire, or choose (Kane, 2002). However, not all compatibilists agree on these features as compatibilist accounts have different variations. For instance, Frankfurt (1969) developed an argument to undermine what he calls Principle of Alternative Possibilities asserting "a person is morally responsible for what she does do only if she can do otherwise." This assumption links free will with moral responsibility and shows that a person who cannot choose or do otherwise may nevertheless cause his action in "the right kind of way" and the agent is morally responsible, but at the time of the relevant action, could not do otherwise.

Although determinist accounts are widely discussed and enormous literature has emerged around the topic of compatibility of free will with determinism, according to Kane (2002), the debate is still on the side of compatibilism. The ongoing discussions about the implications of quantum physics and its interpretations relevant to free will have not got much agreement by now. According to Kane, the orthodox implications of quantum physics is indeterministic but has been challenged by scientists from different backgrounds. Although in microscopic levels including the atomic and subatomic particles there is indeterminacy; however, this indeterminacy can comparatively be negligible for macroscopic levels when human brain and body are taken into consideration. That is, modern determinists have a pragmatic approach and argue that it is possible to continue to regard human behavior as determined at the macroscopic level for the practical purposes. Furthermore, developments in sciences, other than physics, have been showing the

views that human behavior are determined by causes beyond our control such as the influence of genetics and heredity, e.g., improvements in neuroscience which helps to get greater awareness on the biochemical influence on the brain, human moods and behaviors to drugs and the recent mapping of the human genome (ibid.).

4.2.3 Incompatibilist Accounts

On the contrary, the incompatibilist approach holds the view that if determinism is true, nobody can have free will and be morally responsible for her acts because her acts are consequences of laws of nature and events in the remote past so that nobody will have the chance to say that "I could have done otherwise". Since free will exists, according to incompatibilists, who are usually referred to as libertarians, determinism is false. However, there exist some incompatibilists such as (Honderich, 2002), so-called hard determinists, who approach from the other side of the problem of determinism and assert that determinism is true and *ipso facto* nobody can have free will.

Regarding the four questions posited before in Section 4.2, libertarians should answer the question of intelligibility and provide explanation for the existence and significance questions as well. However, another point should also be acknowledged: If free will is not compatible with determinism, then it might not be compatible with indeterminism either. When the free actions required undetermined events, the opponents of indeterminism argue that this point of view does not make us more free agents since if determinism is false, the occurrence of an event would be a matter of chance; the actions would be uncaused, capricious, arbitrary, random or uncontrolled. Thus nobody could be held morally responsible for his or her actions. Even in the case that these undetermined events have some effects on the

brain or human body, they would be unpredictable and uncontrollable by the agents. Ayer (1954) argues this point by presenting a *reductio ad absurdum* for the defense of determinism:

> Either it is an accident that I choose to act as I do or it is not. If it is not an accident, then it is merely a matter of chance that I did not choose otherwise. If it is merely a matter of chance that I did not choose otherwise, it is surely irrational to hold me morally responsible for choosing as I did. But if it is not an accident that I choose to do one thing rather than another, then presumably there is some causal explanation of my choice: and in that case we are led back to determinism. (Ayer, p.275, 1954)

However, determinism issue can also be seen as an artificial problem since it takes us nowhere as the given prior conditions would not let us to do otherwise in the future. Kane (1985) puts forward this idea as follows:

> …what I cannot understand is how I could have reasonably chosen to do otherwise, how I could have reasonably chosen B, given exactly the same prior deliberation, that led me to chose A, the same information deployed, the same consequences considered, the same assessments made, and so on. (Kane, 1985, p. 57)

Thus we have two intersection points for compatibilist and incompabitilitist accounts: First of all, from the optimistic side, we can assume that both accounts accept, except some "hard" theories from both sides, the existence of possible worlds where we have free will. For compatibilists the worlds where we have free will include deterministic worlds, and for incompatibilists the only worlds where we have free will are non-deterministic worlds. Secondly, but from pessimistic side, if we are determined agents, we are not responsible for anything. Everything had been the consequence of the remote past and we would not be able to change it. But, on the other hand, if every action is undetermined, then what we do is by chance, and we are not responsible for anything.

In the literature, there have been two broad categories of indeterministic

incompatibilist accounts (or libertarianism): agent-causal theories and teleological intelligibility theories. These theories can be distinguished regarding their explanations of actions in terms of causation. Causation is, broadly, is the principle that every event is necessitated by its antecedent (Hoefer, 2005). As the cause occurs, given the same conditions of the causal relationship, the effect also occurs. That is, the occurrence of the first event (i.e., cause) explain the occurrence of the succeeding event (i.e., effect). In a narrower sense, causation requires a causal connection between the agent's reasons and actions.

4.2.4 Agent Causation

Agent causal theory accepts "a sui generis form of causation by an agent that is irreducible (ontologically as well as conceptually) to event-causal processes within the agent" (O'Connor 1995, p.7). Agent causation theory supports the idea that when the agent acts with free will the action decided by the agent is caused by only the agent itself and this causation is itself not causally determined by any prior events. Thus, the agent is the strict originator of her action, a "prime mover unmoved" (Chisholm, 1982), "an uncaused cause of her behavior" (Clarke, 1995) or as O'Connor puts forward, the agent is "rather than [her] activity's being a product of external conditions that impinge on [her] in various ways, establishing internal states that in turn cause the behavior, [she is] quite literally the cause (source, point of origination) of [her] own behavior" (O'Connor, 1995). This situation is similar to as it is in the Aristotle's famous example in his *Physics*: " The stick moves the stone and is moved by the hand, which is again moved by the man." (Kane, 1995) A very rough but informative attempt to figure this action might look like as in Figure 9. Our motive to find the source of the action, viz. Staff's moving the stone, is ceased by the

agent or the person. On the other hand, the whole series of action (or events) is initiated by the agent, not any other event. The agent is the point of origination of her own actions.

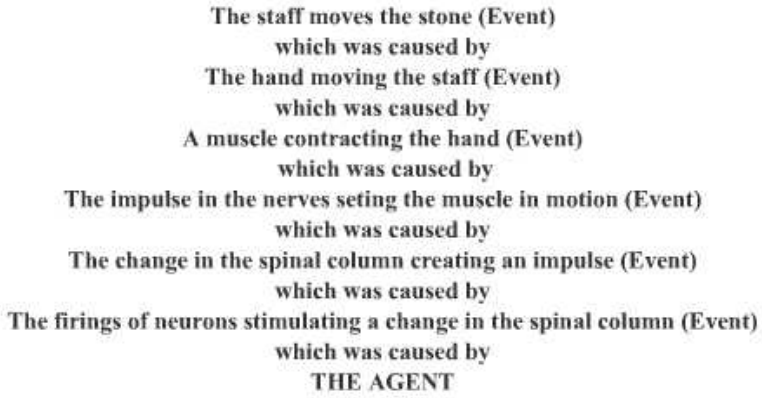

Figure 9: The agent causation sequence of an action (Adapted from McKeever, 2006)

However, the question how do the agent make events in her brain happen is controversial one and can make agent causation theory mysterious. There is no way of understanding the difference between the actions which were initiated with the random neural firing and the agent's causing a neural fire. Chisholm (1982) explains this problem by simply asserting an *endeavor* or *undertake* to make an action happen. For instance, for moving a stone, the agent undertakes to make happen moving the stone. By undertaking this, the agent directly makes certain things happen (such as some neural firings) in her brain, that then cause to move the stone. That is, for Chisholm, making things happen is quite different than things happening. In his later publications (such as 1986) he added desires, beliefs and motives as the necessary conditions for undertakings and these undertakings are non-random.

Agent causation theory requires the action of the agent to be performed through different choices, each of which is naturally possible. Thus, the causation

can be exercised in both directions, by either choosing to do or not. The agent has the control of her choices so that "she determines that she performs that action, and that determination by her is not determined beyond her control" (Clarke, 1995). Another factor which agent causation theory requires is a non-reductionist account of causation which posits a causal factor which cannot be reduced to any form. For example, an event that occurred prior to actual event can only be caused by the agent itself. The non-reductionist causation is not constituted by patterns which are determined by events (or states of affairs) but by ontologically basic sort of relation. This notion of agent causation necessitates the particular to be capable of representing possible courses of action to himself and having certain desires and beliefs regarding these alternatives. The difference of this account with event causation is that the causal power it provides is not characterized by any "function from circumstances to effect" as it is done in event causal powers. This alternative picture of agent-causal paradigm denies *an object's possession of property P in circumstance C* necessitates a certain effect, but favors the idea that it is only by a property of the right sort which make possible the direct bringing about of an effect *by the agent* who bears it. Thus, this sort of causal power is exercised at will by the agent but not that of necessity (O'Connor, 1995).

4.2.5 Teleological Theories

The other incompabilist accounts are discussed under the name of teleological[19] theory. The teleological theory aims to maintain "undetermined free actions intelligibly in terms of reasons and motives, intentions and purposes, without invoking extra entities or special forms of causation." (Kane 2002, p.416)

[19] Teleology can be defined as "the belief that all things and events were specially planned to fulfil a purpose." (Longman Dictionary of Contemporary English, 1987)

Teleological theory can be divided into two categories: causal indeterminist (or event-causal) theory and simple indeterminist (or noncausalist) theory.

Causal indeterminist theorists claim that an agent causes its free actions depending on reasons but in an indeterministic way. Roughly speaking, they take the requirements of compabilitist accounts and add an indeterministic ingredient where agent-involving events that cause the action must nondeterministically cause it (Clarke, 2005). They reject the irreducible form of agent causation but affirm one's decision making or reasoning should be causal and the causal relations hold between reasons and actions. The most salient characteristic of causal indeterminism is that relevant causal relationship need not be deterministic since "undetermined" need not mean "uncaused" (Kane 2002, p.26). This relationship can be such that an event causes the other event even if the former does not determine the latter. That is, in order to take a free action, there should be a nondeterministic causation of the free actions. Kane (2002) argues for taking a free action, the agent should be ultimately responsible for a decision which is causally determined by some character "building acts" or what he calls as "self-forming actions" (or "self-forming willings"). These self-forming actions will determine the choices of the agent by his freely formed character, between two courses of action, to do or not to do. If the agent chose one course of actions, she chose it without no act of randomness, but in case the agent did choose not to do, she was not determined to choose since it was at indeterminacy that the agent would choose. Therefore, it is not compatible with determinism.

However, the probability of the decision will affect the outcome. A change in the probability would change the path taken, i.e., to do or not to do. The agent's decision would be reduced to a neural event which would have the same probability. In case a hypercomputer would explore this situation it would be a Probabilistic

Turing machine since it can make independent random equiprobable choices, i.e., toss a coin with probability of 0.5 at all times.

The most important difference between the agent causation theory and causal indeterminism sets in the origin of the action. The agent causation theory implies that the source of the action is the agent itself whereas causal indeterminist theory posits the notion of event causality which acknowledges the agent's causing the action as an event as well.

Simple indeterminist theory asserts no relation between an agent and its free actions. They are described as non-causal accounts since free decisions or free actions do not have any cause at all, or be nondeterministically caused by other events. For noncausal account, every action is or begins with a basic mental action (Clarke 2002, p. 357). The basic action is volition, which defines agents's willing to act. Being the defender of this theory, Ginet (1990) asserts that an event is a basic action with its capability of possessing some noncausal intrinsic feature which is described as "actish phenomenal quality." This quality provides the agent "as if she is directly producing, making happen, or determining the event that has this quality" (ibid.). But, this quality would be problematic since if the agent, unlike agent causation theory, does not determine the occurrence of the action, then it is not under agent's control.

Free will is a very broad concept and I will not go deep into the subject of free will by considering other arguments (such as religious or neurophilosophical views). Actually, I will confine myself with the problem of determinism and libertarian accounts. I will try to find a relationship between them in order to reveal the fact the free will issue can have a hypercomputational explanation through one of its theories. Here, my motivation will be not to take side and support the arguments

of one camp or other. My goal is to explore the possibility of a different model based on incompatibilist libertarian views.

In attempt to give an example of how a hypercomputer can cope with a philosophical problem, I will try to demonstrate the basis of how Accelerating Turing machines and Putnam-Gold machines, hypercomputers with specific properties beyond Turing-machine-computability, can simulate or represent plausible models of agent causation theory of free will and introduce a specific hypercomputer called reverse Zeus machine.

4.3 A Hypercomputational Device for Agent Causation

Now let us proceed with discussing how a hypercomputational system could be used to cope with a specific theory of free will, namely the agent causation theory. However, first of all, the question why agent causation is chosen as a reasonable theory for a hypercomputational approach should be answered. What I will try to represent here is the properties of agent causation which makes it a plausible model to be used and examples of hypercomputational devices proposed.

In order to proceed to an example of hypercomputational device for antecedent events, it is reasonable to settle down necessary preliminaries. As it was stated in Section 3.2.1, Turing machines, besides being discrete and deterministic, are formal and finite systems. For example in his (1937) paper Turing said that "The 'computable' numbers may be described as the real numbers whose expressions as a decimal are calculable by finite means." (Turing 1937, p.116) Although decimal expression of a computable number can be infinite and the machine will run without terminating, the input to the machine is finite and there is an effective procedure to compute it. This is a property intrinsic to all Turing machines. Therefore, one of the

preliminaries is that Turing machines must enter a finite number of configurations during some interval $[t_i, t_k]$ (Bringsjord, 1992).

The most salient reason to pick up agent causation theory for implementing in a hypercomputational device is the existence of the origin or the start of an agent's all actions. The agent is the first element of an action series. Likewise, for instance, accelerating Turing machines which perform the operations called for by the program taking only one moment among their temporal patterning start with a first element. Moreover, the agent causation theory is consistent with the idea that the agent is an "undetermined determinant of one's action." (Clarke, p.203, 1995) That means, the agent determines which actions she performs. The free action is performed by a causal connection of prior events. This causal chain can be extended in both directions, i.e., backwards and forwards in an imaginary time path. In both cases potential problems can be eliminated by the inherent characteristics of hypercomputation. As it will be explained later, there will be certain hypercomputers to cope with the extension in both directions. The occurrence of certain prior events does not prevent performance of an agent's causing a certain event. The occurrence of certain succeeding events does not, either. Besides these properties, an essential probabilistic causal role in agent causation theory does not weaken the predictive and explanatory significance of event-causes (ibid.). Thus, it also compromises with some form of event causation since probabilistic issues concerning causal role in agent causal theories can also be used to simulate the behavior of free will.

However, this conformity of agent causality with event causality is said to create a further problem. O'Connor (1995) puts forward the idea of agents causing his agent-causing (regarding Chisholm's general commitments) as follows:

(1) An agent S bears responsibility for an event x only if S has causally contributed to the occurrence of x.

(2) Any instance of an agent's causing an event is itself an event.
(3) Agents are responsible for their agent-causings.
(4) Agents cause the events which are their agent-causings. (O'Connor, p.187, 1995)

That means that the agent is responsible for an act by agent causing it, then he is responsible for this further event of his causing of agent causing (ibid.). It entails that the agent causes an infinite number of exertions or she has to complete an infinite number of choices simultaneously. Nonetheless, involvement of infinite regress in agent causation does not prevent us from using it for our hypercomputational approach due to reasons which will be explained later.

The Turing machine enters finite number of states in finite time. But, if human beings are not Turing machines, they should extend the capability of Turing machines by changing this feature. Thus, it is possible to show that human beings can enter infinite number of mental states over some interval $[t_i, t_k]$. Gödel is one of those who support the idea that human mind is capable of entering infinite number of states in finite time:

> ...the mind, in its use, is not static, but constantly developing.... Although at each stage of the mind's development the number of its possible states is finite, there is no reason why this number should not converge to infinity in the course of its development (Kugel, 2002).

This infinitude characteristic can be utilized as a specific variant of agent causation theory called "iterative agent causation" (the argument is originally from Zimmerman 1984 and discussed in Bringsjord 1992). This account of agent causation is called iterative due to involvement of regress in it (Bringsjord, 1992). If we take the potential infinite regress feature of iterative agent causation to be implemented on a hypercomputer, then we can establish a model which can present issues concerning free will. Iterative agent causation can be defined as follows:

> Iterative agent-causation, AC_I, is the thesis that there is a special relation (call it agent causation) which sometimes obtains between a person s and an event (state of affairs, proposition, …) ϕ which is such that if s agent-causes ϕ, then
>
> (i) s is a person and ϕ is a state of affairs (event proposition);
> (ii) ϕ obtains;
> (iii) there is no state of affairs ψ other than ϕ which event-caused ϕ to obtain;
> (iv) s agent-causes the event [s agent-causes ϕ];
> (v) there is no ψ such that ϕ =[s agent-causes ψ], and s decides to agent-cause ϕ. (Zimmerman 1992, p.283)

These features imply that if there exists an s such that s agent causes ϕ, ϕ would not be event-caused (i.e., determined), then ϕ would be undetermined. That is, the existence of s acknowledges indeterminism. If there exists at least one event that is involved in the act which is not caused by another event then it can be caused by the agent itself.

4.3.1 Reverse Zeus Machines

However, another issue emerges through this description. Then, it would be possible to have a statement like [s agent causes ϕ] to be represented as [s agent causes [s agent causes [s agent causes ϕ]]…] which is possible to continue infinitely. As the Figure 10 implies, agents causing the event ϕ can also be represented as an event (for instance E_1). Nevertheless, in order to get an agent causation account, an agent should cause this event as well, so that we recover from the claim that this is merely an implementation of event causation. This procedure may go to infinite (although it is not necessary) until a point where the agent causes all these chains of events eventually.

Now, the question concerning this problem will be how this formation can be represented on a hypercomputational model. In the first glance, the answer is by an

accelerating Turing machine (or Zeus machine, since any machine having necessary organization is sufficient). Accelerating Turing machines can make computation with a time pattern; they can compute infinitely many operations in finite time by performing each operation in half the time of the previous operation.

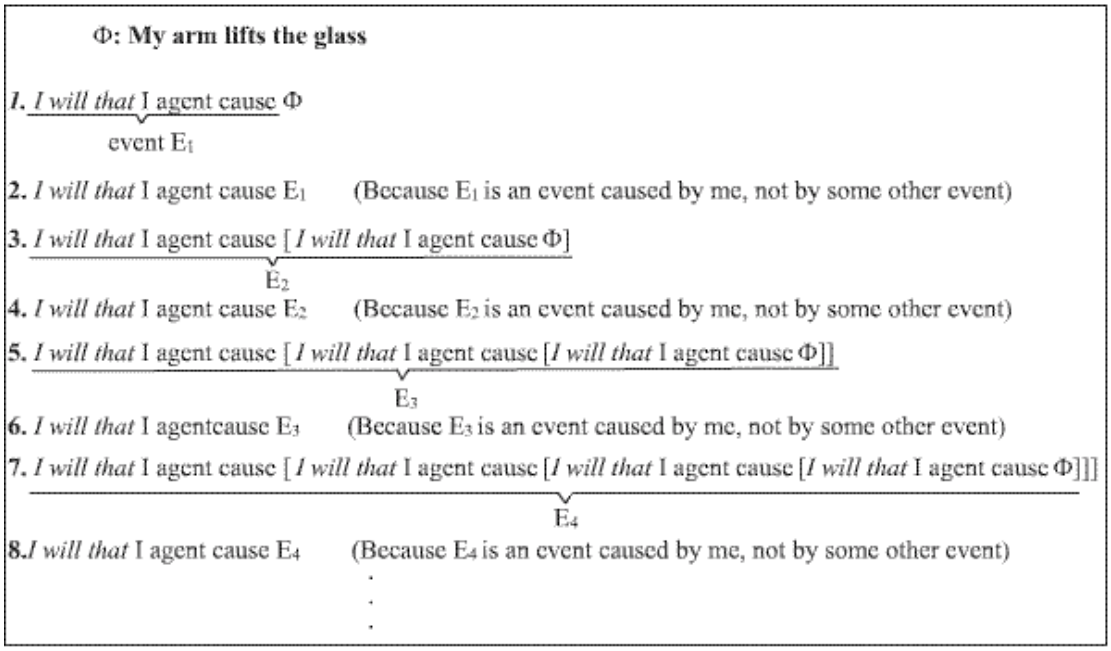

Figure 10: Free action through iterative agent causation

The idea of opponents of agent causation (such as Strawson (1995) and Rowe (1995)) who argue that free will is impossible because it requires that for every choice made by the agent, that the agent chose to make the choice either leads to an infinite regress backwards or it implies that the agent chose before his first choice. Although it seems unreasonable at first sight, it is not unsound if we understand it as a kind of reverse accelerating machine, or some sort of *reverse Zeus machine* in which there is an infinite series with no first member or, to state mathematically, finding the first member of the series where the limit is known (Figure 11). I propose the name reverse Zeus machine due to machine's temporal patterning which is

toward backwards. This model is more reasonable than a potential standard Zeus machine (which moves forward in the causal chain) due to the reasons I will assert.

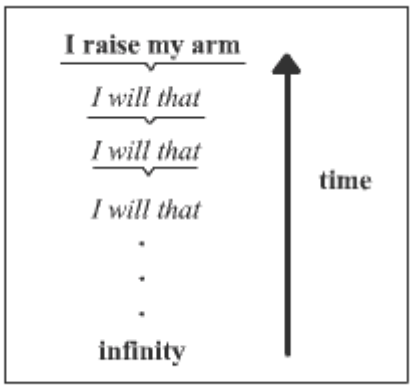

Figure 11: Action through reverse Zeus machine

This situation is similar to implementation of a super-task. An example is given as Thomson's lamp (Thomson 1954, p.5) which asserts impossibility of performance of super-tasks as we saw in Section 3.2.1. However, the reverse Zeus machine which is capable of implementing the action will carry properties of being a kind of anti-Thomson's lamp machine. Similar to Thomson's lamp example, where the first element of the task is known but the last element of the whole infinite sequence of jabs is not, the first element of the sequence of the reverse Zeus machine is not known but the last element goes to infinite. However, the difference is in the implementation path of the series, i.e., backwards, unlike to Thomson's lamp super-task.

The time (t) and action (ϕ) patterns of the reverse Zeus machine (Figure 12) show that the action is from infinite towards the first element, i.e., from the agent's causing the event in the end to the action carried out by the agent. The action is carried out at time *t,* however, the decision to choose the action is before the action.

This is also in compliance with neurophysiological findings of different studies, such as in (Schultz, 1999) and (Libet, 2002). Recording the electrical state of individual neurons, Schultz (1999) asserts that the activitations regarding the processing of the reward information might reflect the evaluation of outcome before the behavioral reaction is executed. Likewise, discussing the experimental studies in human subjects regarding the human activities to the appearance of willed or conscious actions, Libet (2002) emphasizes that voluntary acts begins several hundred milliseconds before the human subjects become consciously aware to act by a specific electrical charge in the brain. That is, the agent's choice has already been caused before her action. In the time path, then it is reasonable to say when the action is concerned, the agent choose to do that action before her action, with a regress in time. If we accept the time the action has been carried out as zero point, then the decision to choose to do that action stands in a minus point in the path of time.

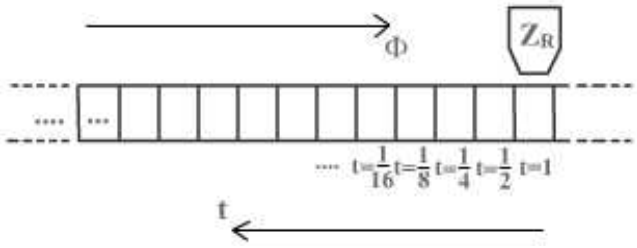

Figure 12: Time and action pattern of reverse Zeus machine

Similar to Thomson's lamp example, it is possible to combine the time pattern in reverse Zeus machines with its action pattern. That is, at time t=1, the action happens, at t=1/2, the agent S agent causes the action ϕ, at t=1/4, S agent causes its agent causing, i.e., [S agent causes ϕ], at t=1/8, S agent causes [S agent causes [S agent causes ϕ]], and so on (Figure 13). The crucial point here is that it is not known

when this sequence of agent causing ends. However, the action will eventually be agent caused in a way.

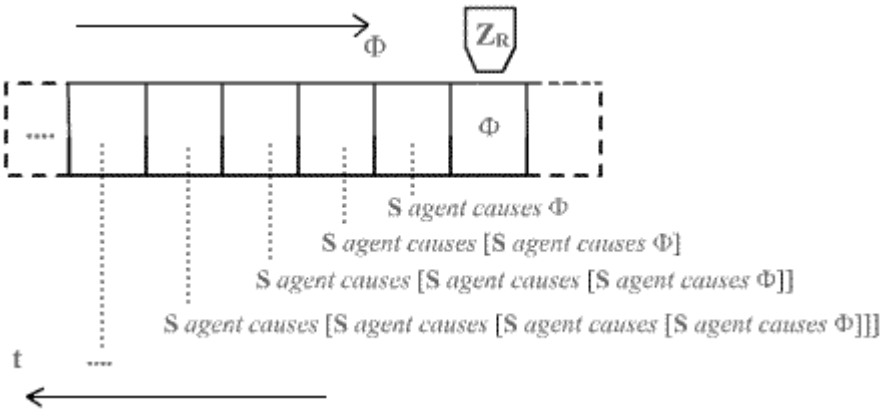

Figure 13: Agent causation in reverse Zeus machines

How a given reverse Zeus machine computes whether it agent causes the action or not is similar to working principle of accelerating Turing machine (as it is given in Section 3.2.2). The value of the famous halting function can also be computed with the reverse Zeus machine. In order to adapt the halting problem to our reverse Zeus machine, we change the implementation of the halting function. From section 2.3, we know that the halting function is a function having either 0 or 1 as its value and an example reverse Zeus machine is composed of the alphabet [1,0]. Again from the definition of halting function 1 (or YES) is used for the halting of the Turing machine M on the input w, and *0* (or *NO)* if M never stops. In our reverse Zeus machine, 1 and 0 are modified to tell whether the computation of the sequence of agent causings end or not. Given a reverse Zeus machine Z_R, and given the description pair with the Turing machine M and input action w as (M, w), Z_R with the input string <M, w> is identified as follows:

$$Z_R(\langle M,w\rangle) = \begin{cases} YES \text{ or } 1 \text{ if M eventually causes w} \\ NO \text{ or } 0 \text{ if M never causes w} \end{cases}$$

In the beginning of the computation, the program of reverse Zeus machine is printed on the tape. Likewise, before the reverse Zeus machine is set into motion, the *last* square is inscribed for the display of the output of computation and by default is picked up as 0. A restriction for reverse Zeus machine can be described such that the last square is the result square and the tape of the reverse Zeus machine can never pass to right of this square. If the reverse Zeus machine halts on the input actions series, then reverse Zeus machine scanning the input tape returns to the last square and change the value written there to 1, which means the agent has caused the action. If the reverse Zeus machine does not halt on the input actions series, reverse Zeus machine leaves this square unchanged and the scanner never returns to the last square, which means the action has not been agent caused. Either way, we can know whether the agent has caused the action or not by the end of two units of time. However, due to intrinsic feature of reverse Zeus machine, the time path will be backwards, from the action to agent causings. This feature, nevertheless, will not change the working principle of reverse Zeus machines. A crucial point here is that the action series is imposed by the notion of temporality due to our definition of reverse Zeus machines. Likewise, the time series provides the acceleration aspect of these machines which has a mathematical solution. Comparatively, the action series might be spatial as well. However, this would introduce a circular tape which could simulate the same pattern.

The features of reverse Zeus machine makes it more advantageous than Zeus machines since in case of Thompson's lamp super task problem we will never be able to know if the lamp is on or off although we can calculate it in finite time. Zeus

machines, undoubtedly, provide a developed model for analyzing the theoretical capabilities and limitations of super tasks. However, the reverse Zeus machine, since it takes its first element as its last element of the standard Zeus machine will be able to tell if the lamp is on or off.

What we have is an infinite number of steps in a finite length of time, a peculiar combination of determinism and indeterminism. The infinite series is deterministic since *each member* has a cause. But the *series itself* could not have been predicted prior to the span of time in which it exists. The reverse accelerating hypercomputation shows the objection against the theory of agent causation is not problematic: infinite regress involved in choosing is conceptually possible because there is conceptually possible a hypercomputer that can do the relevant computation. A reverse Zeus machine does not refute the sort of causation according to which events have causes. This feature of it is attractive since it makes possible to apply reverse Zeus machines to event causation.

Moreover, since the possible statements of iterative agent causation system we referred to in Figure 10 and Figure 11 can be infinitely long, we can simulate its behavior on a machine which can make computation "in the limit". This machine is Putnam-Gold machine and can be effective for explaining infinitely long decision processes involved in iterative agent causation. The output, which is the action of the agent, will be represented as an n-trial procedure. The procedure allows an unlimited number of tries as the last output is the machine's output and the hypermachine can change its mind (i.e., change its choice) many times. That is, the output is the consequence of the preceding sequence of reverse Zeus machine but we can never be sure of this sequence will result in that specific action since through this sequence of agent causings, the Putnam-Gold machine can decide to bring about some other

action or prevent the result of the preceding action. However, the capabilities of Putnam-Gold machines are restricted. The agent causing sequence of the n-trial procedure of the Putnam-Gold machine will only show it is possible to control the outcome of the willed actions.

Although reverse Zeus machines provide a logical basis for infinite regress problem regarding theory of agent causation, it is possible to raise some objections to its structure.

First of all, all the objections for hypercomputers are also valid for reverse Zeus machines. Hypercomputers are purely theoretic machines with an additional power or changes in the structure of standard model of Turing machines. Therefore, all these machines lack physical feasibility in current physics. The nature does not permit to use, for instance, infinite memory or be faster than speed of light as in the case of Zeus machines. Zeus machines (or accelerating Turing machines) necessitate each step to be carried out faster than the one before, which means to exceed the speed of light in further steps. However, the evolution of current physics into quantum theory, as it has evolved from Newtonian physics before, will hopefully reveal the plausibility of implementations of processes that cannot be simulated by standard models. A series of number of discussion papers published in recent years (such as (Stannet, 2001) and (Kieu, 2002)) can help to explore these possibilities.

Regarding the reverse Zeus machines' implementation on infinite regress issue there might seem to be a contradiction. The action which the agent caused is a physical event. We can see it, feel it, and understand it. However, explaining this event with an abstract and physically infeasible notion, called infinite regress, may not seem explicit in respect of the correlation between them. Moreover, the hypothesis which acknowledges infinite states of mind to be implemented in finite

time by the intrinsic feature of reverse Zeus machine can be said to be unconvincing. However, if we accept hypercomputational models, at least, logically, we can use this concept successfully without a physical basis for the time being.

To sum up, our purpose of establishing a relationship between agent causation and hypercomputation seems promising since it is possible to find parallelisms between the intrinsic features of agent causation and certain features of hypercomputers. The existence of hypercomputers (though they are only theoretical machines for the time being) helps us to understand and explore the theory of agent causation. If we provide a strong basis for the principles of hypercomputation, our knowledge about agent causation increases as well. Hopefully, the proposed hypercomputers will possess this capability.

CHAPTER 5

CONCLUSION

5.1 Discussion

As a conclusion, the theory of hypercomputation can provide us with valuable tools for looking at the problems of computational theory of mind, and therefore, with a concrete basis for getting a reasonable extension of it. Covering all accounts that computational theory of mind depends essentially upon, it is possible to broaden our perspective by the concept of hypercomputation. This thesis just aimed to explore a starting point by presenting in-depth exploration of a specific theory of free will from the perspective of some specific hypercomputational machines.

The proposed reverse Zeus machine can explain the so-called infinite regress problem and even establishes a basis for using for the theory of agent causation. This machine can explain the agent's causing the action through its backwards sequence of actions. Hence, it helps to eliminate a philosophical problem for the well-being of theory of agent causation.

This thesis looks into computationalism from the perspectives of Turing machine paradigm and hopes to open new discussion areas in respect to the topics covered in cognitive science and philosophy of mind. The starting point was to

investigate hypercomputation and to discuss its applications in different fields. Later, I came up with agent causation theory and its problematic phenomenon. My aim was to unify these two different concepts through the infinite regress issue involved in it. Undoubtedly, other indeterministic theories, such as event causation theory, would be used as well. However, in the literature infinite regress issue is an argument mostly used against agent causation theory. Nevertheless, it may be possible to analyze agent causation in terms of event causation, and vice versa, insofar there exists an infinite regress involved.

Besides the reverse Zeus machines, it can be possible to find different abstract machines or models for explaining indeterministic theories of free will. Different hypercomputers entail different capabilities and features. Penrose's arguments which claim that human cognitive abilities exceed those of standard computers necessitated that we look into a new physics which can explain brain processes better. Since, mere computation is insufficient to explain the ingredients which the mind involves, it is possible to look into philosophy of mind and find examples. For instance, in the first glance, partially random machines and probabilistic Turing machines can be modified to show an important aspect of free will, viz. randomness, since agent causation theory does not reject a kind of probabilistic role (Clarke, 1995). Moreover, quantum hypercomputation offers very appealing areas of interest. Following Penrose (1989, 1994), it is possible to study consciousness with respect to quantum hypercomputation.

5.2 Future Work

The goal of cognitive scientists is to understand how mind works. A broader study can also help to find explanation to other issues in philosophy of mind as well

as cognitive science. Kugel (2002) enumerates the fields which hypercomputers help the way we study intelligence: computer science, brain sciences, mathematics, artificial intelligence, cognitive science, and philosophy. Regarding philosophy of mind, different developments in these fields can feed our understanding of mind and contribute to a new theory, called "hypercomputational theory of mind".

As said before, any weak point of hypercomputation is also valid for the reverse Zeus machine. Eventually, what I propose is an abstract machine, with a logical basis. We cannot experience it on human beings; we have no means of examining its success on different subjects. Moreover, any study to prove whether free will exists or is compatible with determinism or not is also controversial. Therefore, reverse Zeus machines represent a restricted model.

The reverse Zeus machine has been used for improving our understanding of the so-called problematic position of theory of agent causation during this study. The concept of free will, undoubtedly, involves much more different aspects. Yet, there is no reason to disregard reverse Zeus machines since they establish a distinctive relationship between a specific subject field in computer science (i.e., hypercomputation) with another one in philosophy (i.e., agent causation theory of free will).

BIBLIOGRAPHY

Abramson, F. G. (1971). "Effective Computation over the Real numbers". Twelfth Annual Symposium on Switching and Automata Theory. Northridge CA: Institute of Electrical and Electronics Engineers.

Beckman, F. S. (1980). Mathematical Foundations of Programming. Addison-Wesley Publishing Company.

Benjamin, S. & Ekert, A. (2006). Towards Quantum Information Technology. Prepared for the PdJ Production & ARTE. Retrieved January 20, 2006, from http://www.qubit.org/library/intros/nano/nano.html

Boden, M. (1998). "Artificial Intelligence". In E. Craig (Ed.), Routledge Encyclopedia of Philosophy, Version 1.0, London: Routledge.

Boolos, G. S. & Jeffrey, R. C. (1974). Computability and Logic, Second Edition. Open University Set Book, Cambridge University Press, Cambridge.

Bringsjord, S. (1992). What Robots Can and Can't Be. Kluwer Academic Publishers.

Bringsjord, S. & Zenzen, M. (2002). "Toward a Formal Philosophy of Hypercomputation". Minds and Machines, 12, pp.241-258.

Bringsjord, S., & Zenzen, M. (2003). Superminds: People Harness Hypercomputation, and More. Kluwer Academic PublishersDordrecht /Boston/London.

Butterfield, J. (1998). "Determinism and Indeterminism". In E. Craig (Ed.), Routledge Encyclopedia of Philosophy, Version 1.0, London: Routledge.

Chalmers, D. (2006). “Minds, Machines, And Mathematics A Review of Shadows of the Mind by Roger Penrose”. Retrieved January 25, 2006, from http://psyche.cs.monash.edu.au/v2/psyche-2-09-chalmers.html

Chisholm, R. M. (1982). “Human Freedom and the Self”. In G. Watson (Ed.). Free Will. Oxford: Oxford University Press.

Chisholm, R. M. (1986). “Self-Profile”. In R.J. Bogdan (Ed.). Roderick M. Chisholm (Profiles). Dordecht:Reidel.

Chomsky, N. (1957). Syntactic Structures. The Hague/Paris: Mouton.

Church, A. (1936). “An Unsolvable Problem of Elementary Number Theory”. American Journal of Mathematics, 58, pp.345-363.

Clarke, R. (1995). “Toward a Credible Agent-Causal Account of Free Will”. In Timothy O’Connor (ed). Agents, Causes, Events Essays on Indeterminism and Free Will. Oxford University Press.

Clarke, R. (2002). “Libertarian Views: Critical Survey of Noncausal and Event-Causal Accounts of Free Agency”. In Robert Kane (Ed.). The Oxford Handbook of Free Will. Oxford University Press.

Clarke, R. (2005). "Incompatibilist (Nondeterministic) Theories of Free Will". The Stanford Encyclopedia of Philosophy (Fall 2005 Edition). Edward N. Zalta (Ed.). Retrieved November 25, 2005, from http://plato.stanford.edu/archives/fall2005/entries/incompatibilism-theories/

Copeland, B. J. (1996). “What is Computation?”. Synthese, 108:335-359.

Copeland, B. J. (1997). “The Broad Conception of Computation”. American Behavioural Scientist, 40:690-716.

Copeland, B.J. (1998a). “Even Turing Machines Can Compute Uncomputable Functions”, pp. 150-164. In Cristian S. Calude, J. Casti and M. J. Dineen, (Eds.). Unconventional Models of Computation. Springer-Verlag, Singapore.

Copeland, B. J. (1998b). “Turing’s O-machines, Penrose, Searle, and the Brain”. Analysis 58, pp.129-138.

Copeland, B. J. (1998c). “Super Turing-Machines”. Complexity, 4:30-32.

Copeland, B. J. & Sylvan, R. (1999). “Beyond the Universal Turing Machine”. Australasian Journal of Philosophy, 77:46-66.

Copeland, B. J. (2000). “Narrow Versus Wide Mechanism”. Journal of Philosophy 96, pp. 5–32.

Copeland, B. J. & Proudfoot, D. (2000). “What Turing Did After He Invented the Universal Turing Machine”. Journal of Logic, Language and Information, 9, pp.491-509.

Copeland, B. J. (2002a). “Accelerating Turing Machines”. Minds and Machines, 12, pp.281-301.

Copeland, B. J. (2002b). “Hypercomputation”. Minds and Machines, 12, pp.461-502.

Copeland, B. J. (2004). The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life Plus the Secrets of Enigma. Oxford University Press, ISBN 0-19-82-50-80-0.

Davis, M.(1973). Computability and Unsolvability. Dover Publications, Inc. New York.

Davis, M. (2000). The Universal Computer: The Road from Leibniz to Turing. New York : Norton.

De Leeuw, K. & Moore, E. F. & Shannon, C. E. & Shapiro, N. (1956). “Computability by Probabilistic Machines”. In C. E. Shannon and J. McCarthy (Eds,). Automata Studies. Princeton University Press, Princeton, N.J.

Ekert, A & Macchiavello, C. (1998).“An Overview of Quantum computing”. In Unconventional Models of Computation. C.S. Calude, J. Casti, M.J. Dinneen, (Eds,). Springer, Singapore.

Eliasmith, C. (1996). The third contender: A critical examination of the dynamicist theory of cognition. Philosophical Psychology. Vol. 9 No. 4 pp. 441-463. Reprinted in P. Thagard (Ed) (1998). Mind Readings: Introductory Selections in Cognitive Science. MIT Press.

Encyclopædia Britannica (2006). "Subatomic particle". Encyclopædia Britannica Premium Service. Retrieved February 22, 2006, from http://www.britannica.com/eb/article-60742

Feferman, S. (1994). "Turing in the Land of O(z)". In Rolf Herken (Ed.). The Universal Turing Machine A Half-Century Survey Second Edition. Springer-Verlag, Wien New York.

Feferman, S. (2002). "Penrose's Gödelian Argument A Review of Shadows of the Mind by Roger Penrose". Retrieved December 17, 2002, from http://psyche.cs.monash.edu.au/v2/psyche-2-07-feferman.html

Fodor, J. A. (1975). The Language of Thought. Harvard University Press, Cambridge, MA.

Frankfurt, H. (1969). "Alternate Probabilities and Moral Responsibility". Journal of Philosophy, 66, pp. 829-39.

Franklin, S. (1995). Artificial Minds. A Bradford Book, Massachusetts Institute of Technology.

Gandy, R. (1994 and 1995.). "The Confluence of Ideas in 1936". In Rolf Herken (Ed.). The Universal Turing Machine A Half-Century Survey Second Edition. Springer-Verlag, Wien New York.

Ginet, C. (2002). "Reasons Explanations of Actions: Causalist versus Noncausalist Accounts". In Robert Kane (Ed.). The Oxford Handbook of Free Will. Oxford University Press.

Gold, E. M.(1965). "Limiting Recursion". Journal of Symbolic Logic 30, pp. 28–48.

Hamkins, J. D. & Lewis, A. (1998). "Infinite Time Turing Machines". Journal of Symbolic Logic, 65:567-604.

Harnad, S. (1993). “Grounding Symbols in the Analog World with Neural Nets”. Think (Special Issue on Machine Learning).
Haugeland, J. (1981). Mind design (Ed.). Cambridge, MA: MIT Press.

Haugeland, J. (1997). Mind Design II Second Edition (Ed.). MIT Press.

Hoefer, C. (2005). "Causal Determinism", The Stanford Encyclopedia of Philosophy (Summer 2005 Edition). Edward N. Zalta (Ed.). Retrieved October 25, 2005, from http://plato.stanford.edu/archives/sum2005/entries/determinism-causal/

Hofstadter, D. R.. (1979, 1999). Gödel, Escher, Bach: an Eternal Golden Braid. N.Y: Basic Books Inc.

Hogarth, M. L. (1992). “Does General Relativity Allow an Observer to View an Eternity in a Finite Time?”. Foundations of Physics Letters, 5:173-181.

Honderich, T. (2002). “Determinism as True, Both Compatibilism and Incompatibilsm as False,and the Real Problem”. In Robert Kane (Ed.). The Oxford Handbook of Free Will. Oxford University Press.

Horst, S. (2005). "The Computational Theory of Mind”. The Stanford Encyclopedia of Philosophy (Fall 2005 Edition). Edward N. Zalta (Ed.). Retrieved October 29, 2005, from http://plato.stanford.edu/archives/fall2005/entries/computational-mind/

Kane, R. (1985). Free Will and Values. Albany, NY: State University of New York Press.

Kane, R. (2002). The Oxford Handbook of Free Will, Oxford University Press.

Kane, R. (1995.) “Two Kinds of Incompatibilism”. In Timothy O’Connor (Ed). Agents, Causes, Events Essays on Indeterminism and Free Will. Oxford University Press.

Kieu, T.D. (2002).“Quantum Hypercomputation”. Minds and Machines, 12, pp.541-561.

Kleene, S.C. (1994 and 1995). “Turing’s analysis of Computabilty, and Major Applications of It”. In Rolf Herken (Ed.). The Universal Turing Machine A Half-Century Survey Second Edition. Springer-Verlag, Wien New York.

Klein, S.A. (2002). “Is Quantum Mechanics Relevant To Understanding Consciousness? A Review of Shadows of the Mind by Roger Penrose”. Retrieved May 30, 2003, from http://psyche.cs.monash.edu.au/v2/psyche-2-03-klein.html

Kugel, P. (2002). “Computing Machines Can’t Be Intelligent (...and Turing Said So)”. Minds and Machines, 12, pp.563-579.

Lewis, H. R. & Papadimitriou, C. H. (1992). Elements of the Theory of Computation Prentice-Hall International, Inc.

Libet, B. (2002). “Do We Have Free Will”. In Robert Kane (Ed.). The Oxford Handbook of Free Will. Oxford University Press.

Liskiewicz, M., & Reischuk, R. (1997). “Computational Limitations of Stochastic Turing Machines and Arthur-Merlin Games with Small Space Bounds”. Lecture Notes In Computer Science; Vol. 1295, pp. 91-107.

Lloyd, S. (1998). “Unconventional Quantum Computing Devices”. In Unconventional Models of Computation. C.S. Calude, J. Casti, M.J. Dinneen, (Eds.). Springer, Singapore.

McKeever, S. (2006). “Chisholm and Agent Causation”. Retrieved January 24, 2006, from http://www.ithaca.edu/faculty/smckeever/ChisholmNotes2.html

Minsky, M. L. (1967). Computation Finite and Infinite Machines. Prentice-Hall, Inc. Eaglewood Cliffs, N.J.

MITECS (2002). Language of Thought. In Wilson, R.A. and Keil, F.C. (Eds.). The MIT Encyclopedia of the Cognitive Sciences. Cambridge, Mass. : MIT Press.

Newell, A., & Simon, H.A. (1997). “Computer Science in Empirical Inquiry: Symbols and Search”. In John Haugeland (Ed.) Mind Design II Philosophy Artificial Intelligence Foundations of Cognitive Science An Anthology. A Bradford Book, Massachusetts Institute of Technology, pp.81-110.

O’Connor, T. (1995). “Agent Causation”. In Timothy O’Connor (Ed.). Agents, Causes, Events Essays on Indeterminism and Free Will. Oxford University Press.

Ord, T. (2002). "Hypercomputation: Computing More Than the Turing Machine". Honours Thesis, University of Melbourne. Retrieved October 25, 2002, from http://arXiv.org/math.LO/0209332

Oxford Dictionary of Computing Fourth Edition. (1996). Oxford University Press: Oxford.

Penrose, R. (1989). The Emperor's New Mind Concerning Computers, Minds and The Laws of Physics. Oxford University Press.

Penrose, R. (1994). Shadows of The Mind: A search for the Missing Science of Consciousness. Oxford University Press.

Penrose, R. (1997). The Large, the Small and the Human Mind. Cambridge: Cambridge University Press.

Port, R.F. (2001). "Dynamical Systems Hypothesis in Cognitive Science". In: The MacMillan Encyclopedia of Cognitive Science. Assoc. Amy Lockyer (Ed.). London: MacMillan. Retrieved March 24, 2006, from http://www.cs.indiana.edu/hyplan/port/pap/dynamic.cognition.sglspc.htm

Putnam, H. (1965). "Trial and Error Predicates and the Solution of a Problem of Mostowsky". Journal of Symbolic Logic 30, pp. 49-57.

Putnam, H. (1975). Mind, Language, and Reality. Cambridge University Press.

Rowe, W. L. (1995). "Two concepts of freedom". In Timothy O'Connor (Ed.). Agents, Causes, Events Essays on Indeterminism and Free Will. Oxford University Press.

Scarpellini, B. (1963). "Zwei Unentscheitbare Probleme der Anlysis". Zeitschrift für mathematische Logik und Grundlagen der Mathematik 9, pp.-265-289. As cited in Copeland, B.J. (2002b). "Hypercomputation". Minds and Machines, 12, pp.461-502.

Schultz, W. (1999). "The Primate Basal Ganglia and the Voluntary Control of Behaviour". In: Benjamin Libet, Anthony Freeman and Keith Sutherland (Eds.). The Volitional Brain: Towards a Nueroscience of Free Will. Imprint Academic, UK.

Searle, J.R. (1980). "Minds, Brains, and Programs". In Mind design. J. Haugeland (Ed.). Cambridge, MA: MIT Press, 1981. Also in The Philosophy of Artificial Intelligence. Margaret A. Boden (Ed.). Oxford: Oxford University Press.

Searle, J. R. (1990a). “Is the Brain a Digital Computer?”. Proceedings and Addresses of the American Philosophical Association, 64, pp. 21-37.

Searle, J. R. (1990b). “Is the Brain's Mind a Computer Program?”. Scientific American, 262(1): 20-25.

Siegelmann, H. T. & Sontag, E.D. (1994). “Analog Computation via Neural Networks”. Theoretical Computer Science, 131:331-360.

Siegelmann, H. T. (2003). “Neural and Super-Turing Computing”. Minds and Machines, 13, pp.103-114.

Stannett, M. (1990). “X-machines and the Halting Problem: Building a Super-Turing Machine”. Formal Aspects of Computing 2: 331-341.

Stannett, M. (2001a). “Computation over Arbitrary Models of Time, A Unified Model of Discrete, Analog, Quantum, and Hybrid Computation”, Technical Report CS-01-08. Department of Computer Science, Sheffield University, UK.

Stannett, M. (2001b). “Hypercomputation is Experimentally Irrefutable”. Technical Report CS-01-04. Department of Computer Science, Sheffield University, United Kingdom.

Stewart, I. (1991). “Deciding the Undecidable”. Nature, 352:664-665.

Strawson, G. (1995). “Libertarianism, Action, and Self-Determination”. In Timothy O’Connor (Ed.). Agents, Causes, Events Essays on Indeterminism and Free Will. Oxford University Press.

Thagard, P. (1996). Mind: Introduction to Cognitive Science. A Bradford Book. The MIT Press, Cambridge, Massachusetts.

Thomson, J. F. (1954). ‘‘Tasks and Super-Tasks’’. Analysis 15, pp.1-13.

Turing, A. M. (1936-7). "On Computable Numbers, With an Application to the Entscheidungsproblem". Proceedings of the London Mathematical Society, (2) 42, pp 230-265; correction ibid. 43, pp 544-546.

Turing, A. M. (1939). “Systems of Logic Based on the Ordinals”. Proceedings of the London Mathematical Society, 45:161-228.

Turing, A. M. (1950). “Computing Machinery and Intelligence”. Mind. 59:433-460.

Van Gelder, T. (1999). “Dynamic Approaches to Cognition”. The MIT Encyclopedia of the Cognitive Sciences. MIT Press, pp.243-5.

APPENDICES

Appendix A

The set of states K is $\{q_0, q_1, q_H\}$; the alphabet Σ is {0,1,B}; the initial state is q_0; the final state is q_H. B is for blank tape and H designates the halting state. B tells the machine the sequence ends. The machine needs two states one for odd and one for even, it changes states whenever it encounters a 1. The component on the machine's position, "0" means move left and "1" means move right.

The quintuples can be described as follows:

(symbol read, old state, new state, symbol written, direction) , or

(S_i, q_i, q_j, S_j, X)

The state table for state q_0 is represented as:

Table 5: The quintuples for parity counter for state q_0

S_i	q_i	q_j	S_j	X
0	0	0	0	1
1	0	1	0	1
B	0	H	0	-

The finite-state machine for state q_1 is represented as:

Table 6: The quintuples for parity counter for state q_1

S_i	q_i	q_j	S_j	X
0	1	1	0	1
1	1	0	0	1
B	1	H	1	-

The machine scans an input string from left to right. The behavior of the machine changes according to the states q_0 and q_1. The tables of each state tell the machine how to move.

The steps of the computation are as follows:

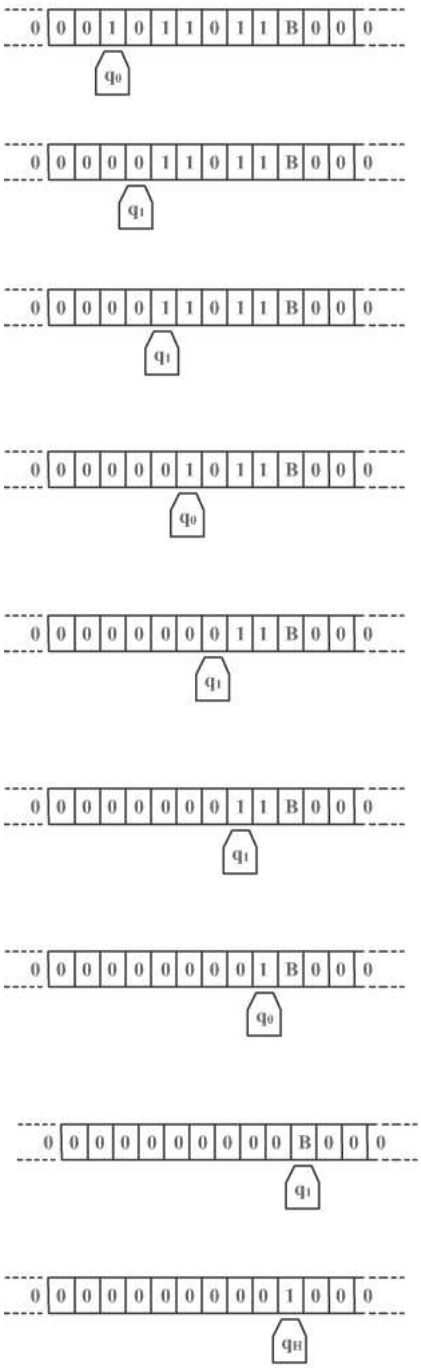

Figure 14: The steps of computation for the Turing machine

Appendix B

The set of states K is {q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}}; the alphabet Σ is {1,B, R, N, L,* }; the initial state is q_1; B is the blank symbol, * tells the machine the sequence ends, L is left, R is right, and N is do nothing.

The quintuples can be described as follows:

(old state, symbol read, symbol written, direction, new state) , or

(q_i, S_i, S_j, X, q_j)

The additional quintuples is described as follows:

(old state, symbol read, symbol written, new state, new state) , or

There is a conditional branch in execution of states. The symbols scanned and written do not change.

Turing machine Z_0 is defined as follows:

Table 7: The definition of quintuples for Turing machine Z_0

State	State
$q_1$1BRq_2	$q_4$1BRq_4
$q_2$1Nq_3q_4	q_4*BLq_8
$q_3$1BRq_3	q_8BBLq_8
q_3*BLq_5	q_8**Rq_9
q_3*BLq_5	q_9B1Rq_{10}
q_5BBLq_5	q_{10}B1Rq_{10}
q_6B1Rq_7	q_{11}B*Lq_{12}
q_7B*Lq_{13}	q_{12}11Lq_{13}