A Hypercomputation in Brouwer’s Constructivism

Rasoul Ramezanian
Department of Mathematical Sciences,
Sharif University of Technology,
P. O. Box 11365-9415, Tehran, Iran
ramezanian@sharif.edu

School of Mathematics,
Institute for Research in Fundamental Sciences (IPM),
P.O. Box: 19395-5746, Tehran, Iran

Abstract

In contrast to other constructivist schools, for Brouwer, the notion of “constructive object” is not restricted to be presented as ‘words’ in some finite alphabet of symbols, and choice sequences which are non-predetermined and unfinished objects are legitimate constructive objects. In this way, Brouwer’s constructivism goes beyond Turing computability. Further, in 1999, the term hypercomputation was introduced by J. Copeland. Hypercomputation refers to models of computation which go beyond Church-Turing thesis. In this paper, we propose a hypercomputation called persistently evolutionary Turing machines based on Brouwer’s notion of being constructive.

1 Introduction

Over the last hundred years, certain mathematicians have tried to rebuilt mathematics on constructivist principles. However, there are considerable differences between various representatives of constructivist, and there exists no explicit unique answer to what a constructive method is. In contrast to other constructivist schools as Bishop’s and Markov’s, for Brouwer, the notion of “constructive object” is not restricted to be presented as ‘words’ in some finite alphabet of symbols. According to Brouwer, mathematics is a free creation of mind, mathematical objects are mental constructions, a languageless activity, and choice sequences are legitimate constructive objects (see pages 20-21, subsection 4-3 of [22]).

Further, in 1999, the term hypercomputation was introduced by J. Copeland. Hypercomputation refers to models of computation which go beyond Church-Turing thesis. Hypermachines are more powerful than Turing machines because of use of hypercomputational resources.

Based on Brouwer’s notion of being constructive, we introduce a hypercomputational resource named “free will” which Brouwer considers it to know choice sequences as legitimate constructive object. If one does not accept free will for human being, and knows human being as a Turing machine then all choice sequences are Turing computable. Brouwer is on the opposite side.

We propose a hypercomputation called persistently evolutionary Turing machines. A persistently evolutionary Turing machine is a machine that its inner structure during its computation on any input may evolve.
2 Hypercomputational Resources

In this section, we briefly recall a collection of hypercomputational resources and those hypermachines that use these resources. We may list following items as some hypercomputational resources (see [17]):

1. non-recursive information sources,
2. infinite memory,
3. infinite specification,
4. infinite computation, and
5. interaction.

The paradigm hypercomputation starts with the \textit{o-machine}, proposed by Turing in 1939 [23]. O-machine is a Turing machine equipped with an oracle that is capable of answering questions about the membership of a specific set of natural numbers. If the oracle set is recursive then the o-machine gains no new power, but if the oracle set is not itself computable by Turing machines, the o-machine may compute an infinite number of non-recursive functions. The o-machines use non-recursive information sources [17] as hypercomputational resources [17].

\textit{Coupled Turing machine}, introduced by Copeland and Sylvan [9], is a Turing machine with one or more input channels, providing input to the machine while the computation is in progress. The specific sequence of input determines the functions that the coupled Turing machine can perform. It exceeds a Turing machine if the sequence of input is non-recursive. Like o-machines, coupled Turing machines use non-recursive information sources. Besides the above discussed hypermachines, \textit{Asynchronous network of Turing machines} [9], \textit{Error prone Turing machines} [17], and \textit{Probabilistic Turing machines} [15] are well-known hypermachines that all use non-recursive information sources as their hypercomputation resources.

Another way to expand the capabilities of a Turing machine is to allow it to begin with infinite number of symbols initially inscribed on its tape. \textit{Turing machines with initial inscription} which have an explicitly infinite amount of storage space are not physically plausible.

Another kind of Hypermachines is \textit{infinite state Turing machines}. An infinite state Turing machine is a Turing machine where the set of states is allowed to be infinite. This type of machine has an infinite amount of transitions, with only a finite number of transition from a given state. This gives the Turing machine an infinite program of which only a finite (but unbounded) amount of transitions is used in any given computation. The infinite state Turning machines require \textit{infinite specification} which does not seems physically plausible.

In the last century, Bertrand Russell [18], Ralph Blake [6], and Hermann Weyl [24] independently proposed the idea of a process that performs its step in one unit of time and each subsequent step in half the time of the step before [17]. Therefore, such a process could complete an infinity of steps in two time unit. The application of this temporal patterning to Turing machines has been discussed briefly by Ian Stewart [20] and in much more depth by Copeland [8] under the name of accelerated Turing machines. The hypercomputation
resource is *infinite computation*. To achieve infinite computation through acceleration, we rapidly run into conflict with physics. For the tape head to get faster and faster, its speed converges to infinite.

Joel Hamkins and Andy Lewis presented another kind of hypermachines that use infinite computation [13], named *infinite time Turing machines*. The infinite time Turing machine is a natural extension of Turing machine to transfinite ordinal times, the machine would be able to operate for transfinite numbers of steps. An interesting case about *infinite time Turing machines* is that it has been proved $P \neq NP$ for this model of computation [19].

Among other resources: non-recursive information sources, infinite memory, infinite specification, and infinite computation, the interaction seems to be possible with our current physics. A kind of hypermachines that use the interaction as resource is the class of persistent Turing machines (PTMs), *multiple machines with a persistent worktape preserved between interactions*, independently introduced by Goldin and Wegner [11] and Kosub [14]. Consistent PTMs, a subclass of PTMs, produce the same output string for a given input string everywhere within a single interaction stream; different interaction streams may have different outputs for the same input. PTMs are a minimal extension of Turing machines that express interactive behavior. The behavior of a PTM is characterized by input-output streams instead of input-output strings. *Interaction streams* have the form $(i_1, o_1), (i_2, o_2), \ldots$, where $i$'s are input strings and $o$'s are output strings by PTM. For all $k$, $o_k$ is computed from $i_k$ but preceded and can influence $i_{k+1}$. The set of all interaction streams for a PTM $M$ consists its language, $L(M)$. Actually, PTMs extend computing to computable nonfunctions over histories rather than noncomputable functions over strings, whereas persistently evolutionary Turing machines (introduced in this paper) extends computing to interactive computable functions which are not *predetermined*.

### 3 Intuitionism

There are considerable differences between various representatives of constructivist, and there exists no explicit unique answer to what a *constructive object* or a *constructive method* is. In contrast to other constructivist schools as Bishop’s and Markov’s [7], for Brouwer, the notion of “constructive object” is not restricted to have a numerical meaning, or to be presented as ‘words’ in some finite alphabet of symbols [22]. According to Brouwer, mathematical objects are mental constructions, a *languageless* activity and independent of logic. Brouwer recognized the *choice sequences* as legitimate mathematical objects [22].

Imagine you have a collection of objects at your disposal, let’s say the natural numbers. Pick out one of them, and note the result. Put it back into the collection, and choose again. Since you have the *ability to choose freely*, you may choose a different one, or the same again. Record the result, and put it back. You may make further choices and keeping on. A *choice sequence* is what you get if you think of the sequence you are making as potentially infinite [5]. Initial segments are always finite. We cannot make an actually infinite number of choices, but we can always extend an initial segment by making a further choice. The following characteristic of the choice sequences is crucial in our consideration:

The subject successively chooses objects, restriction on future choices, restriction on restriction of future choices, etc. [4] page 6)
The object of classical mathematics have their properties independently from us and are static. Choice sequences, in contrast, depends on the subject (who has to make the choice), and they change through time. They are individual dynamic objects that come into being, at the moment that the subject decides to intend them, and with each further choice, they grow, and they are not necessarily predetermined by some law.

static-dynamic An object is static exactly if at no moments parts are added to it, or removed from it. It is dynamic exactly if at some moment parts are added to it, or removed from it. ([2], page 12)

Since choice sequences are dynamic objects and are accepted as intuitionistic mathematical objects, the notion of “constructive method” cannot be captured by Turing computability in Brouwer’s point of view. In addition, in intuitionism, the notion of decidability differs from the notion of recursiveness. Although any recursive set is decidable, the converse is not true. From intuitionistic view, a subset $A$ of $\mathbb{N}$, is decidable if and only if there exists a sequence $\alpha \in 2^{\mathbb{N}}$, such that, for every $n, \alpha(n) = 1$ if and only if $\alpha(n) = 1$. It is not required that the sequence $\alpha$ is given by a finite algorithm, it can be a choice sequence.

Choice sequences are not Turing computable. In constructing choice sequences one uses a hypercomputational resource which Brouwer calls it choice. Brouwer assumes that the subject has the ability of choosing freely, and by this assumption the subject is not a Turing machine. So besides the hypercomputational resources discussed in session [2] we consider free will as a hypercomputational resource, and introduce Brouwer’s hypercomputation.

A sequence of natural numbers is function from $\mathbb{N}$ to $\mathbb{N}$. So choice sequences are subject dependent and non-predetermined functions. In the sequel, we present hypermachines named Persistent Evolutionary Turing Machines which compute subject dependent and non-predetermined functions.

4 Persistent Evolutionary Turing Machines

The notion of being constructive in Brouwer’s intuitionism goes beyond the Church-Turing thesis. Inspired by Brouwer’s choice sequences, we aim to introduce persistently evolutionary Turing machines. A persistently evolutionary Turing machine is a machine that its inner structure during its computation on any input may evolve. But this evolution is in the way that if a computist (a user) does not have access to the inner structure of the machine then he cannot recognize whether the machine evolves or not.

Definition 4.1 Let $M_1$ and $M_2$ be two (deterministic) Turing machines, and $x \in \Sigma^*$ be arbitrary. We say $M_1$ and $M_2$ are $x$-equivalent, denoted by $M_1 \equiv_x M_2$ whenever if one of the two machines $M_1$ and $M_2$ outputs $y$ for input $x$, then the other one also outputs the same $y$ for the same input $x$.

Definition 4.2 A Persistently evolutionary Turing machine is a couple $N = (\langle z_0, z_1, ..., z_i \rangle, f)$ where $\langle z_0, z_1, ..., z_i \rangle$ is a growing sequence of codes of deterministic Turing machines, and $f$ (called the persistently evolutionary function) is a computable partial function from $\Sigma^* \times \Sigma^*$ to $\Sigma^*$ such that for any code of a Turing machine, say $y$, and any string $x \in \Sigma^*$, if $y$ halts on $x$ then $f(y, x)$ is defined and it is a code of a new Turing machine. The function $f$ satisfies the following condition (that we call it the persistent condition):
- for every finite sequence \((x_1, x_2, \ldots, x_n, x_{n+1})\) of \(\Sigma^*\), and every Turing machine \(y_0\), if \(y_1 = f(y_0, x_1), y_2 = f(y_1, x_2), \ldots, y_n = f(y_{n-1}, x_n)\) and \(y_{n+1} = f(y_n, x_{n+1})\) are defined then we have for all \(0 \leq i \leq n\), \(y_{n+1} = x_{i+1} y_i\).

Whenever an input \(x\) is given to \(N\), the output of the evolutionary machine \(N\) is computed according to the \(z_i\) (\(z_1 = f(z_0, x_1), z_2 = f(z_1, x_2), \ldots, z_i = f(z_{i-1}, x_i)\), where \(x_1, x_2, \ldots, x_i\) are strings that sequentially have given as inputs to \(N\) until now, and \(N\) has halted for all of them), and the machine evolves to \(N = ((z_0, z_1, \ldots, z_i, z_{i+1} = f(z_i, x)), f)\) (if \(z_i\) halts for \(x\)).

Note that since the evolution happens persistently, as soon as \(N\) provides an output for an input \(x\), if we will input the same \(x\) to \(N\) again in future, then \(N\) provides the same output as before. It says that, the machine \(N\) behaves well-defined as an input-output black box.

The persistently evolutionary Turing computation could be assumed as one of forms of Hypercomputation [16, 21]. Note that at each moment of time, a persistently evolutionary Turing machine has a finite structure, but during computations on inputs, its structure may change. So it is not possible to encode a persistently evolutionary Turing machine in a finite word. In Brouwer’s constructivism, choice sequences are accepted as constructive objects. But choice sequences can not be represented in finite codes. A persistently evolutionary Turing machine is an interactive machine (see Chapter 5 of [21], and [10]) that evolves according to how it interacts with its users. One may compare persistently evolutionary Turing machines with persistent Turing machines (PTM). The difference between these two kinds of machines is that PTM’s are static machines that transform input streams (infinite sequence of strings) to output streams persistently [12, 11, 14], whereas persistently evolutionary Turing machines are evolutionary machines that transform input strings to output strings.

We may start with two equal persistently evolutionary Turing machines \(N_1 = (z_0, f)\) and \(N_2 = (z'_0, f')\) where \(z_0 = z'_0\) and \(f = f'\), but as we input strings to two machines in different orders the machines may evolve in different ways.

**Example 4.3** Let \(z_0\) be a code of an arbitrary Turing machine, and \(I : \Sigma^* \to \Sigma^*\) be the function \(I(y, x) = y\) for all \(x, y \in \Sigma^*\). Then \(N = (z_0, I)\) is a persistently evolutionary Turing machine which acts exactly like the Turing machine \(z_0\). Thus any Turing machine can be considered as a persistently evolutionary Turing machine as well.

Similar to choice sequences which are subject dependent, persistent evolutionary Turing machines are user dependent. A user who just has access to the input-output behavior of an evolutionary Turing machine \(N\), cannot become conscious whether the machine \(N\) evolves or not. In other words, if we put a persistently evolutionary machine in a black box, a computist (a user) can never be aware that whether it is a (static) Turing machine in the black box, or a persistently evolutionary one.

In the next example, we let \(NFA_1\) be the class of all nondeterministic finite automata that for each \(M \in NFA_1\), each state \(q\) of \(M\), and \(a \in \Sigma\), there exists at most one transition from \(q\) with label \(a\). We take advantage of the persistent evolutionary Turing machine introduced in this example [14] in the sequel of the paper.

**Example 4.4** We define a function \(h : NFA_1 \times \Sigma^* \to NFA_1\) as follows. Let \(M \in NFA_1\), \(M = (Q, q_0, \Sigma = \{0, 1\}, \delta : Q \times \Sigma \to Q, F \subseteq Q)\), and \(x \in \Sigma^*\). Suppose \(x = a_0a_1 \cdots a_k\) where \(a_i \in \Sigma\). Applying the automata \(M\) on \(x\), one of the three following cases may happen:
1. The automata $M$ could read all $a_0, a_1, \ldots, a_k$ successfully and stops in an accepting state. Then we let $h(M, x) = M$.

2. The automata $M$ could read all $a_0, a_1, \ldots, a_k$ successfully and stops in a state $p$ which is not an accepting state. If the automata $M$ can transit from $p$ to an accepting state by reading one alphabet, then we define $h(M, x) = M$. If it cannot transit (to an accepting state) then we define $h(M, x)$ to be a new automata $M' = \langle Q, q'_0, \Sigma = \{0, 1\}, \delta' : Q' \times \Sigma \rightarrow Q', F' \subseteq Q' \rangle$, where $Q' = Q, \delta' = \delta, F' = F \cup \{p\}$.

3. The automata $M$ cannot read all $a_0, a_1, \ldots, a_k$ successfully, and after reading a part of $x$, say $a_0 a_1 \cdots a_i, 0 \leq i \leq k$, it crashes in a state $q$ that $\delta(q, a_{i+1})$ is not defined. In this case, we let $h(M, x)$ be a new automata $M' = \langle Q, q'_0, \Sigma = \{0, 1\}, \delta' : Q' \times \Sigma \rightarrow Q', F' \subseteq Q' \rangle$, where $Q' = Q \cup \{s_{i+1}, s_{i+2}, \ldots, s_k\}$ (all $s_{i+1}, s_{i+2}, \ldots, s_k$ are different states that does not belong to $Q$), $\delta' = \delta \cup \{(q, a_{i+1}, s_{i+1}), (s_{i+1}, a_{i+2}, s_{i+2}), \ldots, (s_k-1, a_k, s_k)\}$, and $F' = F \cup \{s_k\}$.

For each $M \in NFA_1$, we let $T_M$ be a Turing machine that for each input $x \in \Sigma^*$, the machine $T_M$ first constructs the automata $h(M, x)$, and if $h(M, x)$ accepts $x$, then $T_M$ outputs 1, else it outputs 0.

We define the persistently evolutionary Turing machine $PT = \langle[T_{M_0}], f \rangle$, where $M_0 = \langle Q^0 = \{q_0\}, q_0, \Sigma = \{0, 1\}, \delta^0 = \emptyset, F^0 = \emptyset \rangle$, and $f([T_{M}], x) = [T_{h(M, x)}]$.

**Definition 4.5** Let $N = (\langle z_0, z_1, \ldots, z_i \rangle, f)$ be a persistently evolutionary Turing machine. A language $L \subseteq \Sigma^*$ intended by the subject (the user) via $N$ is the set of all strings which the subject chooses and inputs them to $N$, and $N$ outputs 1 for them.

A language which is intended by the user (the subject) via a persistently evolutionary Turing machine is an unfinished and subject-dependent object (similar to Brouwer’s choice sequences). At each stage of time, only a finite part of it, is recognized with the subject who intends the language. Persistently evolutionary Turing machines exist in time and are temporal dynamic mental constructions (see page 16 of [3]). Also a language intended by a persistently evolutionary Turing machine is *user-dependent*. Two users with two persistently evolutionary Turing machines $N_1$ and $N_2$ with the same initial structure may intend two different languages. For a language defined through a persistently evolutionary Turing machine, membership status of an element is not predetermined and is dependent to the free will of the user.

The initial structure $N = (z_0, f)$ of a persistently evolutionary Turing machine is constructed at a particular moment of time, and then evolves as the user chooses further strings to input. For persistently evolutionary Turing machines what remains invariant is the character of the machine as an evolutionary machine, an evolution that started at a particular point in time and preserves well-definedness. The machine evolves but it is the same machine that evolves and the machine is an individual unfinished object. Note that the user is not allowed to reset the machine and goes back to past. It is because that the evolution is a characteristic of persistently evolutionary Turing machines.

Hypercomputation extends the capabilities of Turing computation via using new resources such as 1- infinite memory, 2- infinite specification, 3- infinite computation and 4- the interaction. Among these four resources, the resources 1, 2, and 3 do not seem physically plausible as they have infinite structures. But the forth one, the interaction, seems
physically plausible for the human being. The human being interacts with its environment and it could be possible that its environment persistently evolve because of interaction. The persistently evolutionary Turing machines use two resources to be Hypercomputations 1-evolution, and 2-interaction which arise from considering free will for the subject who intends languages with persistent evolutionary Turing machines. Both of these two resources could be accepted by the human being as physically plausible resources (Biological structures evolve in Darwin theory). One may hesitate to accept that Persistently evolutionary Turing machines are physically plausible due to his presupposition that the real world is a Turing machine, but if he releases himself from this confinement then it seems to him that persistently evolutionary Turing machines are as physically plausible as he knows Turing machines are. One can implement a persistently evolutionary Turing machine on his personal computer (assuming that the computer has an infinite memory; note that the same assumption is needed for executing Turing machines on a computer). The difference between Turing machines and persistently evolutionary Turing machines is that the languages that the first category recognize are predetermined, whereas in the second category, we can recognize an unfinished and non-predicted language which are subject-dependent.

We may list the following items for why persistently evolutionary Turing machines seems plausible as a constructive approach:

1- The persistently evolutionary Turing computation is a kind of interactive computation which today is accepted as a new paradigm of computation by some computer scientist.

2- The persistently evolutionary Turing machines are as plausible as Turing machines are. Both machines can be simulated by a personal computer (assuming that the computer has an infinite memory).

3- In Brouwer’s intuitionism, choice sequences are accepted as constructive mathematical objects. Choice sequences are growing, unfinished objects. Therefore, the persistently evolutionary Turing machines as growing unfinished objects are acceptable as constructive mathematical objects in Brouwer’s intuitionism.

4.1 Persistent Evolution

Suppose \( B \) to be an input-output black box. For an observer who does not have access to the inner structure of the black box, it is not possible to get aware that whether the inner structure of the black box persistently evolves or not. We only sense a change whenever we discover that an event which has been sensed before is not going to be sensed similar to past. Persistently evolution always respects the past. As soon as, a subject experiences an event then whenever in future he examines the same event, he will experience it similar to past. However, persistent evolution effects the future which has not been predetermined, and not experienced by the subject yet. Therefore,

\[
\text{it is not possible for an agent to distinguish between persistent evolution and being static based on the history of his observation.}
\]

\[1\text{The above statement is the same as Brouwer’s continuity principle for choice sequences.} \]
4.2 Non Pre-determinism

Suppose $N = (z_0, f)$ is a persistently evolutionary Turing machine. The machine $N$ could evolve in different ways due to the free will of the user who chooses freely strings to input to $N$. For example, consider the persistently evolutionary Turing machine $PT = \langle \lfloor T_{M_0} \rfloor, f \rangle$ introduced in example 4.4. If you inputs two strings 111, and 11 respectively, the machine $PT$ outputs 1 for the first input and outputs 0 for the second one. Inputting the string 111 makes the machine to evolve such that it cannot accept 11 anymore. The time has sink back to past, and the machine $PT$ evolved.

However, you had the free will to input first 11 and then 111, and if this case had happened, the machine would have accepted both of them.

The user of a persistently evolutionary Turing machine cannot change the past by his free will, but he can effect the future by his free choices. As soon as a persistently evolutionary Turing machine evolves, it has been evolved, and it is not possible to go back to past. When a persistently evolutionary machine evolves, it is the same machine that evolves, and the evolution is a part of the entity of the machine.

However, since the user has the free will, he can effect the future. The future is not necessarily predetermined, and the user can make lots of different futures due to his free will. For example, let $L$ be the set of all strings that the evolutionary machine $PT$, during the interaction with a user, outputs 1 for them. The language $L$ is not predetermined, and it is a growing and an unfinished object (similar to choice sequences). Consider the formula

$$\phi := (\exists k \in \mathbb{N}) (\forall n > k) (\exists x \in \Sigma^*) (|x| = n \land x \in L).$$

At each stage of time, having evidence for the truth of formula $\phi$ conflicts with the free will of the user, and we never could have evidence for truth of $\phi$. It is because that at each stage of time, only for a finite number of strings in $\Sigma^*$, it is predetermined that whether they are in $L$ or not. Let $m \in \mathbb{N}$ be such that $m$ would be greater than the length of all strings that until now are determined to be in $L$. The user via his free will can input all strings with length $m + 1$ to the machine $PT$ respectively. The machine $PT$ outputs 1 for all of them, and evolves such that $L \cap \{ x \in \Sigma^* \mid |x| = m \}$ would be empty. Also, at each stage of time having evidence for $\neg \phi$ conflicts with the free will of the subject. Again, let $m \in \mathbb{N}$ be such that $m$ would be greater than the length of all strings that until now are determined to be in $L$. The user via his free will can input a string with length $m$ to the machine $PT$. The machine $PT$ outputs 1 for it.

**Theorem 4.6** Let $L \subseteq \Sigma^*$ be the growing language intended by a user (who has free will) through the persistently evolutionary Turing machine $PT$. let

$$\phi := (\exists k \in \mathbb{N}) (\forall n > k) (\exists x \in \Sigma^*) (|x| = n \land x \in L).$$

We can never have evidence not for $\phi$ and not for $\neg \phi$.

**Proof.** See the above argument. \(\Box\)

---

2The notion of free will is appeared in mathematics by Brouwer’s choice sequences (see [1] and [5]).
5 Hyper-computable Functions

In this section, we define computable functions via persistently evolutionary Turing machines.

**Definition 5.1** Let \( N = (\langle z_0, z_1, \ldots, z_i \rangle, f) \) be a Persistently Evolutionary Turing machine. A function \( F : \Sigma^* \rightarrow \Sigma^* \) intended by the subject (the user) via \( N \) is the set of all pairs \((x, y)\) which the subject chooses \( x \) and inputs to \( N \), and \( N \) outputs \( y \) at some moment of time. We refer to the collection of all functions which are intended by the subject via a persistent evolutionary Turing machine by \( PF \).

Every function \( F \in PF \) is an unfinished, dynamic, and subject dependent object. For each \( x \in \Sigma^* \) the value of \( F(x) \) is not predetermined and is determined as soon as the subject who intended \( F \) via a persistently evolutionary Turing machine, say \( N \), chooses \( x \) and inputs it in \( N \).

It is easy to check that all Turing computable functions belong to \( PF \). Brouwer’s choice sequences can be assumed as functions from \( N \) to \( N \). Let TF denotes Turing computable total functions and CN denotes choice sequences. We have

\[
\text{TF} \subsetneq \text{PF} \subsetneq \text{CN}.
\]

**Theorem 5.2** \( \text{TF} \subsetneq \text{PF} \subseteq \text{CN} \).

**Proof.**

*TF \( \subsetneq \) PF.*

Let \( F \in \text{TF} \). Then there exists a Turing machine \( T \) such that \( T \) computes \( f \). Let \( z_0 \) be the code of Turing machine \( T \), and let \( f : \mathbb{N} \rightarrow \mathbb{N} \) be such that \( f(z_0) = z_0 \). Then \( N = (\langle z_0 \rangle, f) \) computes \( F \). Thus \( \text{TF} \subseteq \text{PF} \). The characteristic function of the language \( L \) of the machine \( PT \) (see example [1,4]) belongs to \( \text{PF} \) but not \( \text{TF} \). Assuming that the characteristic function of the language \( L \) of the machine \( PT \) is Turing computable conflicts with the assumption that human being has free will.

*PF \( \subseteq \) CN.*

Let \( F \in \text{PF} \), then there exists a Persistently Evolutionary Turning machine \( N = (\langle z_0 \rangle, f) \) such that subject intends \( F \) through it. The function \( F \) can be assumed as a choice sequence \( \langle F(n) \rangle_{n \in \mathbb{N}} \) where the subject intends it by choosing freely some natural number \( n \) and do some computation on it and consider the result to be \( F(n) \). In this way, we have \( \text{PF} \subsetneq \text{CN} \).
References

[1] M. Ardeshir, R. Ramezanian, *Decidability and Specker sequences in intuitionistic mathematics*, Journal of Mathematical Logic Quarterly, 2009.

[2] M. van Atten, *Phenomenology of choice sequences*. Ph.D thesis, Utrecht University, 1999.

[3] M. van Atten, *Brouwer Meets Husserl, on the phenomenology of choice sequences*, Springer, 2007.

[4] M. van Atten, D. van Dalen. *Arguments for the continuity principle*, Journal of Symbolic Logic, Volume 8, Number 3, Sept. 2002

[5] M. van Atten, *On Brouwer*, Wadsworth Philosophers Series, 2004.

[6] R.M. Blake. *The Paradox of Temporal Process*, Journal of Philosophy, 23:645-654, 1926.

[7] D.Bridges, F.Richman. *Varieties of Constructive Mathematics*. London Mathematical Society lecture note series, 1986.

[8] B. Jack Copeland, *Even Turing machines can Compute Uncomputable Functions*, page 150-164, 1998.

[9] B. Jack Copeland and R. Sylvan, *Beyond the Universal Turing Machine*, Australian Journal of Philosophy, 77:46-66, 1999.

[10] D. Goldin, S. A. Smolka, P. Wegner (Ed.), *Interactive Computation, the new paradigm*, Springer, 2006.

[11] D. Goldin and P. Wegner. *Presisnet as a form of interaction*, Technical Report CS-98-07, Brown University, Department of Computer Science, 1998.

[12] D. Q. Goldin, *Persistent Turing Machines as a Model of Interactive Computation*, In Foundation of information and kKnowledge systems, K. D. Schewe, and B. Thalheim, eds., vol 1762 of Lecture Notes in Computer Scince. Springer-Verlag, Berlin, 2000, pp. 116-135.

[13] J. D. Hamkins and A. Lewis. *Infinite Time Turing Machines*, Journal of Symbolic Logic, 65: 567-604, 1998.

[14] Sven Kosub. *Persistent Computations*, Theoretische Informatik Tech. Report, U. Wurzburg, 1998.

[15] K.D. Leeuw, E.F. Morre, C. E. Shannon, and N. Shapiro. *Computability by probabilistic machines*, 183-212 in C.E. Shannon and J. McCarthy, editors, Automata Studies, Princeton University Press, Princeton, N.J., 1956.

[16] T. Ord, *The many forms of hypercomputation*, Journal of Applied Mathematics and Computation 178, 143–153, 2006.
[17] T. Ord. *Hypercomputation: Computing more than the Turing machine*, Technical Report [arXiv:math.LO/0209332](https://arxiv.org/abs/math.LO/0209332), University of Melbourne, Melbourne, Australia, September 2002.

[18] B.A.W. Russell. *The Limits of Empiricism*, Proceedings of the Aristotelian Society, 36: 131-150, 1936.

[19] R. D. Schindler, *P \neq NP for Infinite Time Turing Machines*, Monatshefte fur Mathematik 139, 4. 335-340, 2003.

[20] Ian Stewart. *Deciding the Undecidable*, Nature, 352:664-665, 1991.

[21] A. Syropoulos, *Hypercomputation: computing beyond the Church-Turing barrier*, Springer, 2008.

[22] A. S. Troelstra, D. van Dalen, *Constructivism in Mathematics, An introduction*, Vol. 1, North-Holland, 1988.

[23] A.M. Turing, *Systems of Logic Based on the Ordinals*, Proceeding of the London Mathematical Society, 45:161-228, 1939.

[24] H. Weyl. *Philosophie der Mathematik und Naturwissenschaft*, R. Oldenburg, Munich, 1927.