Çankaya University Journal of Science and Engineering
Volume 15, No. 1 (2018) 023-037
Date Received: 10/11/2016
Date Accepted: 09/01/2018

CUJ

Permanent Persistent Turing Machine: A new Model for Interactive Computation

Sepehr Ebrahimi Mood[1], Mohammad Masoud Javidi[2,*]

[1]*Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran.*

[2]*Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran.*

E-mail: *sepehr_ebrahimi@math.uk.ac.ir* , *javidi@uk.ac.ir*

Abstract: Since the computation of human life is in the progress rapidly and there is interaction in their computational process, we need to offer new concepts in computational theory, especially in the interactive computation to obviate these needs. In this paper, we introduce a new model for interactive computation. First, we briefly review interactive computation concept. Then, we provide and explain Persistent Turing Machine, as a model for interactive computation. There are some disadvantages in definitions of this model. For example, there is not functional property in the computational process of these models. As well as, the language which was accepted by this model, defined as a set of streams. Whereas, in other computational models, we use a set of strings, for indicating the language which was accepted by a model. As a result, we cannot compare the languages were accepted by these models, with the other models and their languages. Next, we propose Permanent Persistent Turing Machine (PPTM), as a model of interactive computation, so that, eliminate these disadvantages. After that, we define sets and relationships were accepted by the PPTMs and introduce their properties in details. Finally, we compare the computational power of this model with other classical models; and we prove that the computational power of the proposed model is more than Turing machine and it can compute some problems, which are not computed by Turing machine.

Keywords: Interactive Computation, Permanent Persistent Turing Machine, Persistent Turing Machine, Permanent Recursive Languages, Permanent Recursive Enumerable Languages.

1. Introduction

By concept, the word "computation" is synonymous with counting and calculus. Moreover the computer is a technical device that performs the computation process. After the widespread application of electronic computers in 1950 that could not only count but also perform tasks and computing values for functions, theoretical and physical concepts were also included in the definition of computers. Theoretical concepts are described based on computers' capability to do the counting or computing value for a function. The Church-Turing thesis is an example of these types of concepts [10].

According to traditional theory of computation, in which Turing machine is considered as a reference computational model related to the computability of problems, a process is called successful if its related computations halts within a finite time and the desirable output produced. Looking at today's computers based on this viewpoint, many of the computational processes are classified as unsuccessful. For example, consider a computer system responsible for monitoring a function of the respiratory system, that continuously performs its computational processes. This system halts when the patient is dead. The operating system is another example of computers that continuously perform computation processes; halt in such systems indicates that the computer has been unsuccessful in performing its related tasks. On the other hand, many of today's computers, do not calculate the value of a function; but rather, they have the duty of performing services and tasks required by users. Web servers are good examples of such computational models.

Nowadays, computational devices communicate with their environment to perform their tasks. By further development of science in the field of computer engineering, and widespread application of electronic computers for daily tasks, the need for introducing new concepts of computation theories, compatible with available developments, has been noticed [1]. In fact, having interaction with the surrounding environment is an undeniable component of today's computational systems [11]. As a result, we also need to define computational models that are capable of interacting with their surrounding environment. Turing machine, as a closed computational model, is devoid of such a capability [2].

In 1998, Wegner introduced the concept of interactive machine and showed that it has higher computational capability than Turing machine, because they can models interactive computations [3]. Moreover, a set of interactive grammars was described and a suitable position was considered for these grammars within Chomsky's hierarchy. The interactive machine is a type of Turing machine which is dynamically capable of reading and writing on its tape. This capability has transformed Turing machines from a closed and pre-defined system into a dynamic and active one that is capable of interaction with its external environment. This characteristic is imposed to the machine by providing one or multiple data streams as input or providing symmetrical or asymmetrical communication with the environment [4]. Persistent Turing machine (PTM) that was introduced by Goldin [5], is also capable of performing interactive computations. This model has a persistent tape which is called state of the machine. PTMs perform their task using their state and the input stream. Kosub in [12], investigate the persistent Turing machine which is a computational model for interactive computation. Some computability concepts such as essential and conditional computability are studied in ref. [12]. Some computational models for interactive computations such as interactive Turing machines (ITMs) and reactive Turing machines (RTMs) are investigated in [13] and their computational behavior is compared.

The behavior of interactive machines is determined according to the data stream. A data stream is a time-sensitive model composed of a sequence of strings .Wegner defined a recursive stream over the sets as follows [3]:

Definition 1.1. Suppose $A \subset \Sigma^*$ is a set; then, a stream over A is a recursive sequence $S =< a, S' >$, in which $a \epsilon A$ and S' are other streams over the set A.

Previous computational models for interactive computation have some disadvantages. Their input should be streams over the alphabet; but, other computational models defined on strings as their input. Moreover, these computational models have not functional behavior. It means, they can produce different output for the same input at different time step. In this paper, we proposed permanent persistent Turing machine (PPTM) a new model for interactive computation. The proposed model can compute some relations over strings. Furthermore, this model has the functional behavior. We defined permanent recursive and permanent recursive enumerable languages which are computable by PPTMs. Then, we prove that the computational power of the proposed model is more than Turing machine; and they can solve some problems which are not computable by TMs.

This paper is organized as follows. In section 2, the definition of the persistent Turing machine and some notions and preliminaries are described. The definition of the proposed model, permanent persistent Turing machine, and some properties of this model is investigated in section 3. In section 4, we study the computational behavior of the proposed model and compare the computational power of this model with the standard Turing machine, and finally, section 5 contains a brief conclusion.

2. Persistent Turing Machine

Persistent Turing Machine (PTM) which is proposed by Goldin [5], is a generalization of Turing machine that is capable of performing interactive computations. In fact, this model is utilized for performing interactive sequential computations as a dynamic stream of an ordered pair of inputs and outputs. A PTM is a special type of Turing machine that utilizes three distinct tapes (input, output, and work tapes) for its computations.

The contents of work tape are permanently stored even after the completion of computations and end of the computational process. This tape acts as a memory for the PTM and its contents show the state of the machine (which is different from that of Turing machine) before and after computation process. Therefore, state of a PTM is defined by the strings of infinite length. Formally, Goldin provided a definition of the persistent Turing machine in [5] as follows:

Definition 2.1. A Persistent Turing Machine (PTM) is a type of multi-tape Turing machine with a permanent (persistent) work tape whose contents are preserved between sequential computations by the Turing machine. Contents of this tape before and after the computations are called state of the PTM.

Computations of a PTM like P, is a mapping of $\varphi_P: I \times W \to O \times W$, where I, O and W are input-output streams and machine's work tape (state of the machine), respectively. In fact, the machine P transforms input stream $< i_1, i_2, i_3, \ldots >$ into output stream $< o_1, o_2, o_3, \ldots >$ and during this computation process, it uses contents of work tape, i.e. state of machine. At the beginning and end of the computation process, the machine is always in a null state; after the computational processes, machine's state is preserved to be used for the next steps. Intuitively, in each computational step, machine's output can be determined as a computable function of its input and state (i.e. contents of work tape) [5]. For further clarification about how these machines perform computational tasks, we use an example mentioned in [2].

Example 2.2. Consider a PTM, A, whose computational task is as follows:

$$\varphi_A(recordx, y) = (ok, yx)$$

$$\varphi_A(playback, x) = (x, x)$$

$$\varphi_A(erase, y) = (done, \varepsilon)$$

The task of A is to store, display, and delete strings. The key point is that the content of work tape is used as a part of providing a definition for φ_A and does not have any effect on the performance of A machine. Now, consider the following stream as an input stream for this machine:

$$< recordA, recordBC, erase, recordD, recordE, playback, \ldots >$$

This machine produces the following output stream on this input steam:

$$< ok, ok, done, ok, ok, DE, \ldots >$$

During this computational process, state of the machine is changed as:

$$< \varepsilon, A, ABC, \varepsilon, D, DE, DE, \ldots >$$

This computational model can compute interactive problems. Goldin presented interactive problems in interactive streams as follows [5]:

Definition 2.3. Suppose that I and O are input and output streams for a PTM, P, respectively. Interactive stream for P is a stream of ordered pairs in the form of (i, o), whose first and second components are chosen from the input and the corresponding output streams.

For instance, in the above-mentioned example, the interactive stream is as follows:

$$< (recordA, ok), (recordBC, ok), (erase, done), (recordD, ok), \ldots >$$

The state of a PTM is not explicitly visible in the machine's interactive stream. However, it acts on the machine's output implicitly. Furthermore, in order to compare the computational behavior of two different PTMs, Goldin introduced the concept of $L(P)$, which is the language of a PTM, as follows:

Definition 2.4. Set of all possible interactive streams for every PTM like P constitutes the language of machine P.

According to this definition, two PTMs, P_1 and P_2, are equal if: $L(P_1) = L(P_2)$. Moreover, the behavior of a PTM is described by computation tasks over data streams. In fact, the language of a PTM is a set of interactive streams that cannot be compared with the language of a Turing machine which is composed of a set of strings. Furthermore, the behavior of a PTM has not functional property [5]. In other words, for a given input provided at different times, there is the possibility of producing different outputs. For instance, in example 2.2, the output of the PTM for input $playback$, given to the machine at different times, would be different according to the state of PTM.

3. Permanent Persistent Turing Machine

A Permanent Persistent Turing Machine (PPTM) is a type of PTM used for interactive computations and has the functional property. This computational model produces a unique output for a given input. However, this output can be different for different interactive streams.

Definition 3.1. A Persistent Turing Machine, P, is called a Permanent Persistent Turing Machine (PPTM), if during a given interactive stream, such as S, and for any desirable input string, it produces a unique output string at different repetitions and times.

Example 3.2. Due to the definition provided in example 2.2, suppose a computational model that computes the following output stream O, for a given input stream, I:

$$I =< recordA, recordBC, playback, erase, playback, recordE, playback, \ldots >$$

$$O =< ok, ok, ABC, done, ABC, ok, ABC, \ldots >$$

After the command playback is executed at the first time (the third command in the input stream) and output ABC is printed, this computer always returns the same output (ABC), by executing the $playback$ command as an input, irrespective of the tape content. On the other words, the behavior of this model has the functional property; hence, it always computes the same output for an input. The content of the work tape in this machine during the process is as follows:

$$< \varepsilon, A, ABC, ABC, \varepsilon, \varepsilon, E, E, \ldots >$$

According to the output stream, the fifth command result is ABC, while there is no symbol in the machine's work tape. As a result, this PTM has the property defined in 3.1; thus, this machine is a PPTM.

In order to be able to analyze the behavior of a PPTM and to discuss its computing power, the languages that are accepted by this computational model must be defined. But, some notations should be introduced before this definition. As mentioned in the definition 1.1 one data stream is a time-sensitive model composed of a sequence of strings. For instance, the stream S over the set A is denoted by $S =< s_1, s_2, s_3, \ldots >$, in which the order of elements $s_1, s_2, s_3, \ldots \in A$ is important. We define S^k as a notation (symbol) to refer to k first component of stream S. Hence, S^k is equal with the following stream:

$$S^k =< s_1, s_2, s_3, \ldots, s_k >$$

Obviously, an interactive stream in a computing machine is a stream whose elements are in the form of ordered pairs of input and output in each computational step and are represented as:

$$S =< (i_1, o_1), (i_2, o_2), (i_3, o_3), \ldots >$$

Where o_j is the output corresponding to input i_j at j^{th} step. Now, after introducing the required definitions, we provide the following definition for relation $R_{M,S}$ which is computable by the PPTM, M, with the interactive stream, S.

Definition 3.3. Relation $R_{M,S} \subseteq \Sigma^* \times \Sigma^*$ is computable by the PPTM M and the interactive stream S if, for every $(x, y) \in \Sigma^* \times \Sigma^* - R_{M,S}$ there would be a number like $k \in N$ such that, if $S_k = S'_k$ for every interactive stream S', then $(x, y) \notin S'$.

Intuitively, the above definition states that, if (x, y) does not satisfy relation $R_{M,S}$, then there is a number like k such that after k component of interactive stream S, it is not possible to reach output y by input x. This result may happen in two ways: first, for input x, the machine produced an output like y' such that $y \neq y'$; thus, it would produce output y' from input x for any other interactive streams. Therefore: $(x, y) \notin S$. The second case is that string x does not belong to the inputs of the interactive stream S. In the following theorem, the relationship between a computable relation by a PPTM and an interactive stream is reviewed.

Theorem 3.4. If $R_{M,S} \subseteq \Sigma^* \times \Sigma^*$is a relation which is computable by the PPTM, M, with interactive stream S, then: $S \subset R_{M,S}$.

Proof. Suppose that (x, y) is one of the components of interactive stream S. Hence, there is not a number like k such that, for any other interactive streams follow, after the k component of S with the input x, y is not computed as the output. That is because the interactive stream S itself is one of these streams (computations are carried out by a PPTM) and therefore (x, y) cannot be eliminated from all the streams. Hence, $(x, y) \in R_{M,S}$.

To complete the proof, we must show that there is an (x, y) so that $(x, y) \in R_{M,S}$, but: $(x, y) \notin S$. To do so, the computing machine M and the interactive stream S are defined as:

M: A PPTM that computes output 1 for the prime numbers regarded as the input.

S: An interactive stream whose first elements are odd natural numbers (consider the interactive stream produced from odd input stream of number 2 as the input);

According to the definition, no components can be found in S that starts with 2. This means: $(2,1) \notin S$. However, on the other hand, since the Permanent persistent Turing machine M is defied to compute output 1 for input 2, then necessarily: $(2,1) \in R_{M,S}$.

Now the concepts of permanent recursive (p.recursive) and permanent recursive enumerable (p.r.e) will be introduced using these computable relations so that the computational behavior of these hyper-computers can be analyzed.

Definition 3.5. Set $L \subseteq \Sigma^*$ is called a permanent recursive (p.recursive) language if characteristics function χL, shown below, can be computed as a relation using a PTM and an interactive stream.

$$\chi L(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases} \tag{3.1}$$

Based on this definition, a language like L is p.recursive if there is such a computability relation like $R_{M,S}$ so that, for every desired $x \in \Sigma^*$ if elements x are members of language L, then $(x, 1) \in R_{M,S}$; otherwise, $(x, 0) \in R_{M,S}$.

Corollary 3.6. The domain for defining the computable relation in a p.recursive languages is Σ^*.

This conclusion can be derived directly from the above-mentioned definition, because in characteristic function for every considered string (all the possible strings over language alphabet), the function output is determined according to being/not being a member of the language.

Definition 3.7. Language $L \subseteq \Sigma^*$, is a permanent recursive enumerable (p.r.e) language if and only if there is a computable relation $R_{M,S} \subseteq \Sigma^* \times \Sigma^*$such that:

$$x \in L \Leftrightarrow x \in Dom(R_{M,S}).$$

Similar to the previous definition, the equivalent form of this definition is described by the following theorem.

Theorem 3.8. Language L is a p.r.e language if and only if there is a computable relation like $R_{M,S} \subseteq \Sigma^* \times \Sigma^*$ so that:

$$x \in L \Leftrightarrow (x, 1) \in R'_{M',S'}.$$

Proof. Suppose L is a p.r.e language. Hence, a relation $R_{M,S}$ which is computable by the PPTM M, and interactive stream S, exists so that:

$$x \in L \Leftrightarrow x \in Dom(R_{M,S})$$

Now, we choose the interactive stream S' as previous interactive stream S. PPTM, M', is also defined the same as M, with the difference that, for an input that M halts and produces different outputs, M' also halts, but gives output 1:

$$M'(x) = \begin{cases} 1 & M \downarrow x \\ \uparrow & o.w \end{cases} \tag{3.2}$$

In this case, the following conclusion holds for any considered string like $x \in \Sigma^*$:

$$x \in Dom(R_{M,S}) \Rightarrow M'(x) = 1 \Rightarrow (x,1) \in R'_{M',S'}.$$

Now, the opposite of this theorem will be proved. There is a computable relation $R'_{M'S'}$ so that:

$$x \in L \Leftrightarrow (x,1) \in R'_{M',S'}.$$

We must show that L is a p.r.e language. In other words, a computable relation $R_{M,S}$ should be defined such that:

$$x \in L \Leftrightarrow x \in Dom(R_{M,S}).$$

The interactive stream S is defined to be the same as the interactive stream S'. The PPTM, M, is also defined to be the same as machine M', with the only difference that, when M' halts at an input with an output other than 1, the computing machine M does not halt on that input and computes the infinite loop.

$$M(x) = \begin{cases} 1 & M'(x) = 1 \\ \uparrow & o.w \end{cases} \qquad (3.3)$$

In this case, the following relation is satisfied:

$$x \in L \Leftrightarrow (x,1) \in Dom\big(R'_{M',S'}\big) \Leftrightarrow (x,1) \in R_{M,S} \Leftrightarrow x \in Dom(R_{M,S}).$$

Therefore, L is a p.r.e language.

4. Computational Power of a Permanent Persistent Turing Machine

In this section, first, the properties of a PPTM is described. Then, the computational power and behavior of the proposed method are compared with other computational models.

Theorem 4.1. Every recursive language like $L \subseteq \Sigma^*$ is a p.recursive language.

Proof. Supposing that $L \subseteq \Sigma^*$ is a recursive language. We will show that the characteristic function χL can be computed using a computable relation such as $R_{M,S}$. Since language L is a recursive language, there would be Turing machine M that is capable of computing function χL. [7]. That is:

$$M(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases} \tag{4.1}$$

Input stream I is so defined that contains all available strings in Σ^*. So, the interactive stream S over input stream $I = < i_1, i_2, i_3, \ldots >$ is defined as: $S = < \big(i_1, M(i_1)\big), \big(i_2, M(i_2)\big), \big(i_3, M(i_3)\big), \ldots >$Now, consider relation $R_{M,S}$. For every considered string like $x \in \Sigma^*$, if x is a member of language L, then: $(x, 1) \in S \Rightarrow (x, 1) \in R_{M,S}$. But, if x is not a member of language L, then: $(x, 0) \in S \Rightarrow (x, 0) \in R_{M,S}$. So, the computable relation $R_{M,S}$ has the capability of implementing function χL. Thus, that is a p.recursive language.

Theorem 4.2. Every recursive enumerable (r.e) language like $L \subseteq \Sigma^*$ is a p.r.e language.

Proof. Supposing $L \subseteq \Sigma^*$ is an r.e language. The computable relation $R_{M,S} \subseteq \Sigma^* \times \Sigma^*$is defined so that: $x \in L \Leftrightarrow x \in Dom(R_{M,S})$. Since L is an r.e language, it is the domain of computable function like φ_M [8]. The input and interactive streams I and S the same as the proof for theorem 4.1 in such a way that all strings Σ^* are covered and the output is corresponding to the computations of Turing machine M. Since relation $R_{M,S}$ responsible for performing computations regarding function, φM has the same domain, therefore:

$$x \in L \Leftrightarrow x \in Dom(\varphi M) \Leftrightarrow x \in Dom(R_{M,S}).$$

On the other hand, using reasoning similar to the previous theorem, $R_{M,S}$ is a computable relation. Therefore, L is a p.r.e language.

Now, by presenting several theorems, we show that opposite cases for the above theorems do not necessarily hold true. But, before presenting these theorems, we refer to the following definition, which is non-recursive language according to ref. [9]:

Definition 4.3. Number τ is an incomputable real number whose n^{th} number is one, if n indicates an ordered pair for Turing machine and its related input such that the Turing computations halt on this input. However, if computations do not halt on this input, n^{th} digit of the number τ would be zero.

Theorem 4.4. The following set is a p.recursive set.

$$H = \{(1, < T, \omega >) | T \downarrow \omega\}.$$

where $< T, \omega >$ is the code for the considered Turing machine T with its input ω; this language constitutes all the pairs whose computations are halted.

Proof. In order to prove that the above language is p.recursive, we must define a computable relation $R_{M,S}$ that is capable of computing the characteristic function for this set: i.e. χH. To do so, we would apply a technique that combines two PPTMs. In other words, first, the initial input stream is given to the PPTM M'; then, the interactive stream computed by this computational model is considered as the input stream for machine M and the computable relation $R_{M,S}$ is defined. Consider the PPTM, M', in which for computing number τ, a unique code $n =< T, \omega >$ can be computed for any given number n via computational processes related to the map used for coding pairs of Turing machine and its related input. The task of the PPTM M' is to code the natural numbers for any given input in an order mentioned above and to show this code as the output.

On the other hand, an input stream to Permanent persistent Turing machine M' is also the ordered digits of τ. Therefore, the interactive stream S' which will be obtained from the above computations is in the form of pairs like: $(i, < T, \omega >)$. Where i is equal to one if computations by Turing machine T halt on input ω; otherwise it is zero. Now we define the computable relation $R_{M,S}$ as:

Suppose that M is a PPTM whose given input stream is the same as the interactive stream computed by machine M. This machine returns element i, as an output. As a result, the interactive stream S computed by the M is as follows: $((i, < T, \omega >), i)$.

In this interactive stream, if number i is equal to one, then Turing machine T would halt at input string ω; if the machine does not halt on that input, the value for i would be zero. Now, relation $R_{M,S}$ which is computed by the PPTM M and interactive stream S as above is equivalent to language characteristic function H: i.e. χH.

The following figures intuitively demonstrate a computation process by PPTMs M' and M along with their interactive stream.

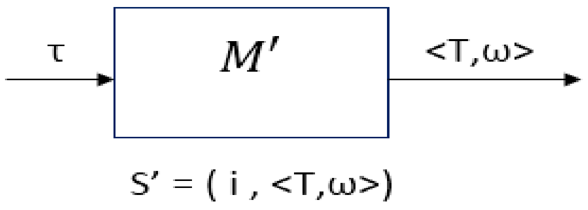

Fig.1 computational process of Permanent persistent Turing machine M'

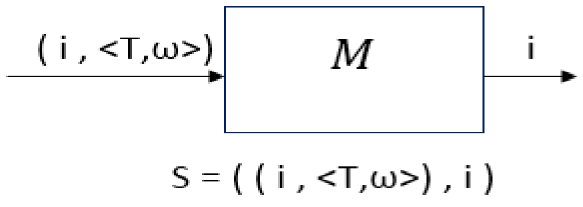

Fig.2 computational process of Permanent persistent Turing machine M

To complete this proof, we need to show that the relation $R_{M,S}$ is a computable relation. To do so, we proof by contradiction and suppose that the relation $R_{M,S}$ is not a computable relation. As a result, there would be an ordered pair like $((i, < T, \omega >), i)$ that does not satisfy the relation $R_{M,S}$; but, there is an interactive stream S' so that for any given k such that $S'^k = S^k$, the above ordered-pair is a member of stream S': $((i, < T, \omega >), i) \in S'$.

According to the structure of this ordered pair and performance of PPTM M, we know that the second element of this pair is also present in the structure of its first element and the machine M is independent of inputs arrangement and computes the output solely based on input structure. In other words, computations over any considered interactive stream are similar to the computations for the interactive stream S. Hence, no interactive stream like S with such properties can be presented. Therefore, our assumption is not correct and thus relation $R_{M,S}$ is a computable relation; consequently, this language is also a p.recursive language.

Now, a similar theorem for r.e languages is described. The following theorem shows that there is a p.r.e language that is not r.e.

Theorem 4.5. The following set is a p.r.e language: $L = \{(0, < T, \omega >) | T \uparrow \omega\}$.

Where $< T, \omega >$ is the code related to a considered Turing machine T with its related input string ω. This language contains the code of Turing machine and its related inputs whose computations are not halted.

Proof. The proof of this theorem is the same as the previous theorem. First, we present a PPTM M'; then, by combining its computations with those related to the PPTM M, the computable relation $R_{M,S}$ will be introduced in such a way that according to the theorem: 3.8 we would have:

$$(0, < T, \omega >) \in L \Leftrightarrow ((0, < T, \omega >), 1) \in R_{M,S}.$$

As a result, we show that the above language is a p.r.e language.

Input for the M' is an ordered stream composed of the digits of number τ. The output is also the code $< T, \omega >$, proportional to its input based on the map used for the number τ. As a result, the interactive stream S produced by the above machine using the given input is an ordered pair in the form of $(i, < T, \omega >)$, in which i is the digit related to the computations of the Turing machine T over input ω for number τ (Figure 1).

Consider the PPTM M whose input is the interactive stream obtained from the computations related to the M' and its output is the opposite of digit i present in its input. Thus, the computations for the Permanent persistent Turing machine M with the related interactive stream are as follows:

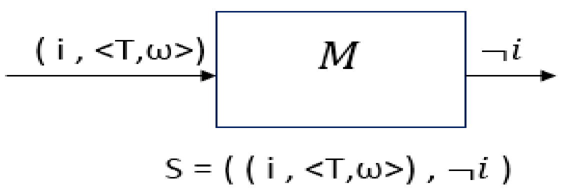

Fig.3 computational process of Permanent persistent Turing machine M

In the interactive stream S which is computed by the PPTM M, if the digit i is zero, i.e. computations by Turing machine T do not halt over input ω, then the output would be zero. In other words, this pair is a member of language L. Equivalently, we have:

$$T \uparrow \omega \Leftrightarrow (0, < T, \omega >) \epsilon L \Leftrightarrow \big((0, < T, \omega >), 1\big) \epsilon R_{M,S}.$$

On the other hand, using similar reasoning to the previous theorem, it can be shown that relation $R_{M,S}$ is a computable relation. As a result, language L is also a p.r.e language.

According to theorems 4.1 and 4.2 as well as the two above-mentioned theorems, the following corollary can be extracted:

Corollary 4.6. Recursive languages are a proper subset of p.recursive languages; and, r.e languages are a proper subset of p.r.e languages.

According to this conclusion, it can be intuitively claimed that the proposed computational model has higher computing power than Turing machine and is also capable of computing problems that cannot be solved by Turing machine, such as problems related to interactive computations.

5. Conclusion

Nowadays, interaction has a major role in the human computations. So, the computation models should interact with their surrounding environments. Persistent Turing machines and interactive machines are the models which can interact with the environment during their procedures and computations. The behavior of these models has not functional property. Besides, the languages admissible by these computational models are also defined in the form of data streams, while other models used the set of strings when defining their admissible languages. This difference in definition did not let us compare the languages admissible by interactive computational models with that of other models. In this paper, a permanent persistent Turing machine, which was the generalization of the persistent Turing machine, is introduced, and languages and functions computable by this computing machine are defined so that all the disadvantages of previous models would be covered. Then, the computational power of the proposed model is investigated and is concluded to be more than that of Turing machine.

References

[1] Goldin, Dina, Scott A. Smolka, and Peter Wegner, eds. Interactive computation: The new paradigm. Springer Science & Business Media, 2006.

[2] Goldin, Dina Q., Scott A. Smolka, Paul C. Attie, and Elaine L. Sonderegger. "Turing machines, transition systems, and interaction." Information and Computation 194, no. 2 (2004): 101-128.

[3] Wegner, Peter. "Interactive foundations of computing." Theoretical computer science 192, no. 2 (1998): 315-351.

[4] Syropoulos, Apostolos. Hypercomputation: computing beyond the Church-Turing barrier. Springer Science & Business Media, 2008.

[5] Goldin, Dina Q. "Persistent Turing machines as a model of interactive computation." In International Symposium on Foundations of Information and Knowledge Systems, pp. 116-135. Springer, Berlin, Heidelberg, 2000.

[6] Wegner, Peter. "Interactive foundations of computing." Theoretical computer science 192, no. 2 (1998): 315-351.

[7] Cooper, B. S. "Computability Theory. Chapman Hall/Crc Mathematics Series." (2003).

[8] Davis, Martin, Ron Sigal, and Elaine J. Weyuker. Computability, complexity, and languages: fundamentals of theoretical computer science. Newnes, 1994.

[9] Ord, Toby. "Hypercomputation: computing more than the Turing machine." arXiv preprint math/0209332 (2002).

[10] Copeland, B. Jack. "The church-turing thesis." Stanford encyclopedia of philosophy (2002).

[11] Kolan, Amy J. "Interactive Computation for Undergraduates: The Next Generation." Journal of Statistical Physics 167, no. 3-4 (2017): 997-1006.

[12] Kosub, Sven. Persistent computations. Inst. für Informatik, 1998.

[13] Luttik, Bas, and Fei Yang. "On the Executability of Interactive Computation." In Conference on Computability in Europe, pp. 312-322. Springer International Publishing, 2016.