On Interactively Computable Functions

Jérémie Cabessa[1,2] and Alessandro E.P. Villa[2]

[1] Université Paris 2 – Panthéon-Assas
Laboratory of Mathematical Economics (LEMMA)
75006 Paris, France
jcabessa[at]nhrg.org

[2] University of Lausanne
Department of Informations Systems
Neuroheuristic Research Group
Ch-1015 Lausanne, Switzerland
alessandro.villa[at]unil.ch

Abstract. Interactive computation refers to the computational framework where systems may react or interact with each other as well as with their environment during the computation. This paradigm was theorised in contrast to classical computation which has been argued to no longer fully correspond to the current notions of computing in modern systems. In this context, we provide a complete characterisation of the so-called ω-translations performed by deterministic interactive systems of any possible kind. Firstly, we show that the class of interactively computable ω-translations corresponds precisely to the class of continuous ω-translations. Secondly, we prove that the interactively computable ω-translations coincide with the ω-translations computable by interactive Turing machines with advices. These results extend previous characterisations of interactively computable functions to the case of arbitrary deterministic interactive systems. They support the interactive extension of the Church-Turing Thesis which states that any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice.

Keywords: interactive computation, interactive Turing machine, interactive Turing machine with advice, ω-translation, computational power.

1 Introduction

Interactive computation refers to the computational framework where systems may react or interact with each other as well as with their environment during the computation [17, 5]. This paradigm was theorised in contrast to classical computation [9] which rather proceeds in a function-based transformation of a given input to a corresponding output (closed-box and amnesic fashion), and has been argued to "no longer fully correspond to the current notions of computing in modern systems" [15]. Interactive computation also provides a particularly

appropriate framework for the consideration of natural and bio-inspired complex information processing systems [11, 15, 2].

Wegner first propose a foundational approach to interactive computation [17]. He claimed that "interaction is more powerful than algorithms", in the sense that computations performed in an interactive way are capable of handling a wider range of problems than those performed in a classical way, namely by standard algorithms and Turing machines [16, 17].

In this context, Goldin et al. introduced the concept of a *persistent Turing machine* (PTM) as a relevant extension of the classical Turing machine model to the framework of interactive computation [3, 4]. A persistent Turing machine consists of a multi-tape machines whose inputs and outputs are given as streams of tokens generated in a dynamical and sequential manner, and whose work tape is kept preserved during the transition from one interactive step to the next. In this sense, a PTM computation is sequentially interactive and history dependent. Goldin et al. further provided a transfinite hierarchical classification of PTMs according to their expressive power, and established that PTMs are more expressive (in a precise sense) than amnesic PTMs (an extension of classical Turing machines in their context of interactive computation), and hence also than classical Turing machines [3, 4].

All these consideration led Goldin and Wegner to formulate the so-called *Sequential Interaction Thesis*, a generalisation of the Church-Turing Thesis in the realm of interactive computation, claiming that "any sequential interactive computation can be performed by a persistent Turing machine" [4, 8, 6, 7]. They argue that this hypothesis, when combined with their result that PTMs are more expressive than classical TMs, provides a formal proof of Wegner's conjecture that "interaction is more powerful than algorithms" [4, 8, 6, 7], and hence refutes what they call the Strong Church-Turing Thesis – different from the original Church-Turing Thesis –, stating any possible computation can be captured by some Turing machine, or in other words, that "models of computation more expressive than TMs are impossible" [8, 7].

Van Leeuwen and Wiedermann proposed a slightly different interactive framework where a general component interacts with its environment by translating an incoming input stream of bits into a corresponding output stream of bit in a sequential manner [10, 14]. In their study, they restrict themselves to deterministic components, and provide mathematical characterisations of interactively computable relations, interactively recognisable sets of inputs streams, interactively generated sets of output streams, and interactively computable translations.

In this context, they further introduced the concepts of an *interactive Turing machines* (ITM) [11] and an *interactive Turing machine with advice* (ITM/A) (described in Section 4) as a relevant TM extension and non-uniform computational model in the context of interactive computation, respectively [11, 12]. The computational power of ITMs and ITM/As, as well as the computational equivalence between ITM/As and several other models of computation has been studied in [11, 12]. These considerations led van Leeuwen and Wiedermann to formulate an *Interactive Extension of the Church-Turing Thesis* which states

that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [12].

As opposed to Goldin and Wegner, van Leeuwen and Wiedermann consider that interactivity alone is not sufficient to break the Turing barrier [11, 13]. They write [15]:

> "From the viewpoint of computability theory, interactive computing e.g. with ITMs does not lead to super-Turing computing power. Interactive computing merely extends our view of classically computable functions over finite domains to computable functions (translations) defined over infinite domains. Interactive computers simply compute something different from non-interactive ones because they follow a different scenario."

In this paper, we follow a similar approach to interactive computation as presented in [10, 14]. We also restrict ourselves to the study of deterministic interactive systems, but make no further assumption about them. In particular, we do not require for the systems to be driven by a Turing program nor to contain any computable component of whatever kind. In this context, we provide a complete characterisation of the ω-translations that can be performed by such interactive systems. Firstly, we show that the class of interactively computable ω-translations corresponds precisely to the class of continuous ω-translations. Secondly, we prove that the interactively computable ω-translations coincide with the ω-translations computable by interactive Turing machines with advices. These results extend those mentioned in [10, 14] to the class of arbitrary deterministic interactive systems. They support the Interactive Extension of the Church-Turing Thesis mentioned above.

2 Preliminaries

Given some finite alphabet Σ, we let Σ^*, Σ^+, Σ^n, and Σ^ω denote respectively the sets of finite words, non-empty finite words, finite words of length n, and infinite words, all of them over alphabet Σ. We also let $\Sigma^{\leq\omega} = \Sigma^* \cup \Sigma^\omega$ be the set of all possible words (finite or infinite) over Σ. The empty word is denoted λ.

For any $x \in \Sigma^{\leq\omega}$, the *length* of x is denoted by $|x|$ and corresponds to the number of letters contained in x. If x is non-empty, we let $x(i)$ denote the $(i+1)$-th letter of x, for any $0 \leq i < |x|$. The prefix $x(0)\cdots x(i)$ of x is denoted by $x[0{:}i]$, for any $0 \leq i < |x|$. For any $x \in \Sigma^*$ and $y \in \Sigma^{\leq\omega}$, the fact that x is a *prefix* (resp. *strict prefix*) of y is denoted by $x \subseteq y$ (resp. $x \subsetneq y$). If $x \subseteq y$, we let $y - x = y(|x|)\cdots y(|y|-1)$ be the *suffix* of y that is not common to x (if $x = y$, then $y - x = \lambda$). Moreover, the *concatenation* of x and y is denoted by $x \cdot y$.

Given some sequence of finite words $\{x_i : i \in \mathbb{N}\}$ such that $x_i \subseteq x_{i+1}$ for all $i \geq 0$, one defines the *limit* of the x_i's, denoted by $\lim_{i\geq 0} x_i$, as the unique finite or infinite word which is ultimately approached by the sequence of growing prefixes $\{x_i : i \geq 0\}$. Formally, if the sequence $\{x_i : i \in \mathbb{N}\}$ is eventually constant, i.e. there exists an index $i_0 \in \mathbb{N}$ such that $x_j = x_{i_0}$ for all $j \geq i_0$,

then $\lim_{i\geq 0} x_i = x_{i_0}$, meaning that $\lim_{i\geq 0} x_i$ corresponds to the smallest finite word containing each word of $\{x_i : i \in \mathbb{N}\}$ as a finite prefix; if the sequence $\{x_i : i \in \mathbb{N}\}$ is not eventually constant, then $\lim_{i\geq 0} x_i$ corresponds to the unique infinite word containing each word of $\{x_i : i \in \mathbb{N}\}$ as a finite prefix.

A function $f : \Sigma^* \to \Sigma^*$ is called *monotone* if the relation $x \subseteq y$ implies $f(x) \subseteq f(y)$, for all $x, y \in \Sigma^*$. It is called *recursive* if it can be computed by some Turing machine. Throughout this paper, any function $\varphi : \Sigma^\omega \to \Sigma^{\leq\omega}$ mapping infinite words to finite or infinite words will be referred to as an *ω-translation*.

Note that any monotone function $f : \{0,1\}^* \to \{0,1\}^*$ induces "in the limit" an ω-translation $f_\omega : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by

$$f_\omega(x) = \lim_{i\geq 0} f(x[0{:}i])$$

for all $x \in \{0,1\}^\omega$. The monotonicity of f ensures that the value $f_\omega(x)$ is well-defined for all $x \in \{0,1\}^\omega$. In words, the value $f_\omega(x)$ corresponds to the finite or infinite word that is ultimately approached by the sequence of growing prefixes $\{f(x[0{:}i]) : i \geq 0\}$.

According to these definitions, in this paper, an ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ will be called *continuous*[1] if there exists a monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$; it will be called *recursive continuous* if there exists a monotone and recursive (i.e. Turing computable) function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$.

Note that our notion of a recursive continuous ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ is a direct transposition to the present context of the notion of a limit-continuous function $\varphi : \{0,1\}^\omega \to \{0,1\}^\omega$ defined in [10, Definition 12] and [14, Definition 13].

3 Interactive Computation

The general interactive computational paradigm consists of a step by step exchange of information between a system and its environment. In order to capture the unpredictability of next inputs at any time step, the dynamically generated input streams need to be modelled by potentially infinite sequences of symbols (indeed, any interactive computation over a finite input stream can a posteriori be replayed in a non-interactive way producing the same output) [17, 6, 15].

Throughout this paper, we consider a basic interactive computational scenario similar to that described for instance in [14]. At every time step, the environment first sends a non-empty input bit to the system (full environment activity condition), the system next updates its current state accordingly, and then answers by either producing a corresponding output bit or remaining silent.

[1] The choice of this name comes from the fact that continuous functions over the Cantor space $\mathcal{C} = \{0,1\}^\omega$ can be precisely characterised as limits of monotone functions. We then chose to extend this definition in the present broader context of functions from $\{0,1\}^\omega$ to $\{0,1\}^{\leq\omega}$ that can also be expressed as limits of monotone functions.

In other words, the system is not obliged to provide corresponding output bits at every time step, but might instead stay silent for a while (to express the need of some internal computational phase before producing a new output bit), or even staying silent forever (to express the case that it has died). Consequently, after infinitely many time steps, the system will have received an infinite sequence of consecutive input bits and translated it into a corresponding finite or infinite sequence of not necessarily consecutive output bits. Throughout this paper, we assume that every interactive system is deterministic.

Formally, given some interactive deterministic system $\mathcal{S}$, for any infinite input stream $s \in \{0,1\}^\omega$, we define the corresponding output stream $o_s \in \{0,1\}^{\leq\omega}$ of $\mathcal{S}$ as the finite or infinite subsequence of (non-λ) output bits produced by $\mathcal{S}$ after having processed input s. The deterministic nature of $\mathcal{S}$ ensures that the output stream o_s is unique. In this way, any interactive system $\mathcal{S}$ realises an ω-translation $\varphi_\mathcal{S} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by $\varphi_\mathcal{S}(s) = o_s$, for each $s \in \{0,1\}^\omega$.

An ω-translation ψ is then called *interactively deterministically computable*, or simply *interactively computable* in this paper, iff there exists an interactive deterministic system $\mathcal{S}$ such that $\varphi_\mathcal{S} = \psi$. Note that in this definition, we do absolutely not require for the system $\mathcal{S}$ to be driven by a Turing program nor to contain any computable component of whatever kind. We simply require that $\mathcal{S}$ is deterministic and performs ω-translations in conformity with our interactive paradigm, namely in a sequential interactive manner, as precisely described above.

4 Interactive Turing Machines and Interactive Turing Machines with Advices

Goldin et al. introduced the concept of a *persistent Turing machine* (PTM) as a relevant extension of the classical Turing machine model in the context of interactive computation [3, 4]. Driven by similar motivations, van Leeuwen and Wiedermann introduced a related concept of *interactive Turing machines* (ITM) [11].

An *interactive Turing machine* (ITM) consists of an interactive abstract device driven by a standard Turing machine program. In other words, the machine receives an infinite stream of bits as input step by step, performs read-write operations on a semi-infinite work tape – exactly like in the case of classical Turing machines – and produces a corresponding finite or infinite stream of bits as output, step by step. The input and output bits are processed via corresponding input and output ports rather than tapes. Consequently, at every time step, the machine cannot operate anymore on the output bits that have already been processed.[2] According to our interactive scenario, it is assumed that at every time step, the environment sends a non-silent input bit to the machine, and the machine answers by either producing some corresponding output bit or remaining silent.

[2] In fact, allowing the machine to erase or modify its previous output bits would lead to the consideration of much more complicated ω-translations.

Van Leeuwen and Wiedermann also introduced the concept of *interactive Turing machine with advice* (ITM/A) as a relevant non-uniform computational model in the context of interactive computation [11, 12]. An *interactive Turing machine with advice* (ITM/A) consists of an interactive Turing machine provided with an advice mechanism, which takes the form of an advice function $\alpha : \mathbb{N} \to \{0,1\}^*$. The machine uses two auxiliary special tapes, an advice input tape and an advice output tape, as well as a designated advice state. During its computation, the machine can write the binary representation of an integer m on its advice input tape, one bit at a time. Yet at time step n, the number m is not allowed to exceed n. Then, at any chosen time, the machine can enter its designated advice state, and then have the string $\alpha(m)$ be written on the advice output tape in one time step, replacing the previous content of the tape. The machine can repeat this process as often as necessary during its infinite computation.

Interactive Turing machines with advice are strictly more powerful than their classical counterpart (i.e., interactive Turing machines without advice) [12, Proposition 5] and [11, Lemma 1], and they were shown to be computationally equivalent to several other non-uniform models of interactive computation, like sequences of interactive finite automata, site machines, web Turing machines [11], and more recently to interactive analog neural networks and interactive evolving neural networks [1, 2]. These considerations led van Leeuwen and Wiedermann to propose the following extension of the Church-Turing Thesis in the context of interactive computation [12]: "Any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice."

According to these considerations, an ω-translation ψ is said to be *ITM-computable* iff there exists some deterministic ITM $\mathcal{M}$ such that $\varphi_{\mathcal{M}} = \psi$. It is called *ITM/A-computable* iff there exists some deterministic ITM/A $\mathcal{M}$ such that $\varphi_{\mathcal{M}} = \psi$.

The following result provide a complete mathematical characterisation of the ω-translations performed by ITMs and ITM/As. The first point of this result is a translation in the present computational context of [10, theorems 4 and 5] and [14, theorems 7 and 8]. The following result in its current form is proven in detail in [1, theorems 3 and 4].

Theorem 1. *1. An ω-translation ψ is ITM-computable iff it is recursive continuous.*
2. An ω-translation ψ is ITM/A-computable iff it is continuous.

Proof (sketch). 1: Let ψ be ITM-computable. Then, there exists an ITM $\mathcal{M}$ such that $\varphi_{\mathcal{M}} = \psi$. Consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word produced by $\mathcal{M}$ after exactly $|u|$ steps of computation over input stream u provided bit by bit. We can show that f is recursive. Moreover, by an exact transposition of the proof of Theorem 2 below, we can prove that f is monotone and satisfies $f_\omega = \varphi_{\mathcal{M}} = \psi$. This means precisely that ψ is recursive continuous.

Conversely, let ψ be recursive continuous. Then there exists a monotone recursive function f such that $f_\omega = \psi$. Now, consider the ITM $\mathcal{M}$ which proceeds

as follows: on every new input bit b_{t+1} received at time step $t+1$, $\mathcal{M}$ computes the value $f(b_0 \cdots b_{t+1})$, looks if this word extends $f(b_0 \cdots b_t)$, and if this is the case, output the extension $f(b_0 \cdots b_{t+1}) - f(b_0 \cdots b_t)$ bit by bit. We can show that $\varphi_{\mathcal{M}} = f_\omega = \psi$, meaning that ψ is ITM-computable.

2: If ψ is ITM/A-computable, then it is interactively computable, and by Theorem 2 below, it is continuous.

Conversely, suppose that ψ is continuous, then there exists a monotone function f such that $f_\omega = \psi$. Now, consider the ITM/A $\mathcal{M}$ which contains some encoding of f is its advice function, and which proceeds as follows: on every new input bit b_{t+1} received at time step $t+1$, $\mathcal{M}$ calls its advice to determine the value $f(b_0 \cdots b_{t+1})$, looks if this word extends $f(b_0 \cdots b_t)$, and if this is the case, output the extension $f(b_0 \cdots b_{t+1}) - f(b_0 \cdots b_t)$ bit by bit. We can show that $\varphi_{\mathcal{M}} = f_\omega = \psi$, meaning that ψ is ITM/A-computable. □

5 Interactively computable ω-translations

The sequential interactive nature of our computational paradigm provides strong restrictions on the ω-translations that can be performed by deterministic systems working in this interactive framework. The following result provides a complete mathematical characterisation of these interactively computable ω-translations. It is a generalisation of [10, Theorem 5] and [14, Theorem 8] to the case of arbitrary interactive deterministic systems, rather than those driven by a Turing program.

Theorem 2. *An ω-translation ψ is interactively computable iff it is continuous.*

Proof. Let ψ be an interactively computable ω-translation. Then by definition, there exists a deterministic interactive system $\mathcal{S}$ such that $\varphi_{\mathcal{S}} = \psi$. Now, consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word produced by $\mathcal{S}$ after exactly $|u|$ steps of computation over input stream u provided bit by bit. Note that the deterministic nature of $\mathcal{S}$ ensures that the finite word $f(u)$ is indeed unique, and thus that the function f is well-defined.

We show that f is monotone. Suppose that $u \subseteq v$. It follow that $v = u \cdot (v - u)$. Hence, according to our interactive paradigm, the output strings produced by $\mathcal{S}$ after $|v|$ time steps of computation over input stream v, namely $f(v)$, simply consists of the output strings produced after $|u|$ time steps of computation over input u, namely $f(u)$, followed by the output strings produced after $|v - u|$ time steps of computation over input $v - u$. Consequently, $f(u) \subseteq f(v)$, and therefore f is monotone.

We now prove that the ω-translation $\varphi_{\mathcal{S}}$ performed by the interactive system $\mathcal{S}$ corresponds to the the "limit" (in the sense of Section 2) of the monotone function f, i.e. we show that $\varphi_{\mathcal{S}} = f_\omega$. Towards this purpose, given some infinite input stream $s \in \{0,1\}^\omega$, we consider in turn the two possible cases where either $\varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$ or $\varphi_{\mathcal{S}}(s) \in \{0,1\}^*$.

Firstly, suppose that $\varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$. According to our interactive scenario, $f(s[0{:}i])$ is a prefix of $\varphi_{\mathcal{S}}(s)$, for all $i \geq 0$ (indeed, once again, what has been produced by $\mathcal{S}$ on s after infinitely many time steps, namely $\varphi_{\mathcal{S}}(s)$, consists of what has been produced by $\mathcal{S}$ on $s[0{:}i]$ after $i+1$ time steps, namely $f(s[0{:}i])$, followed by what has been produced by $\mathcal{S}$ on $s - s[0{:}i]$ after infinitely many time steps). Moreover, since $\varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$, it means that the sequence of partial output strings produced by $\mathcal{S}$ on input s after i time steps of computation cannot be eventually constant, i.e. $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$. Hence, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$ for all $i \geq 0$ and $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$ ensure that $\varphi_{\mathcal{S}}(s)$ is the unique infinite word containing each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix, which is to say by definition that $\varphi_{\mathcal{S}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_\omega(s)$.

Secondly, suppose that $\varphi_{\mathcal{S}}(s) \in \{0,1\}^*$. By the very same argument as in the previous case, $f(s[0{:}i])$ is a prefix of $\varphi_{\mathcal{S}}(s)$, for all $i \geq 0$. Moreover, since $\varphi_{\mathcal{S}}(s) \in \{0,1\}^*$, the sequence of partial output strings produced by $\mathcal{S}$ on input s after i time steps of computation must become stationary from some time step j onwards, i.e. $\lim_{i\to\infty} |f(s[0{:}i])| < \infty$. Hence, the entire finite output stream $\varphi_{\mathcal{S}}(s)$ must necessarily have been produced after a finite amount of time, and thus $\varphi_{\mathcal{S}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$. Consequently, the three properties $f(s[0{:}i]) \subseteq \varphi_{\mathcal{S}}(s) \in \{0,1\}^*$ for all $i \geq 0$, $\lim_{i\to\infty} |f(s[0{:}i])| < \infty$, and $\varphi_{\mathcal{S}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$ ensure that $\varphi_{\mathcal{S}}(s)$ is the smallest finite word that contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix, which is to say by definition that $\varphi_{\mathcal{S}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_\omega(s)$. Consequently, $\varphi_{\mathcal{S}}(s) = f_\omega(s)$ for any $s \in \{0,1\}^\omega$, meaning that $\varphi_{\mathcal{S}} = f_\omega$.

We proved in turn that f is a recursive and monotone function satisfying $\varphi_{\mathcal{S}} = f_\omega$. This means by definition that $\varphi_{\mathcal{S}}$ is continuous. Since $\varphi_{\mathcal{S}} = \psi$, it follows that ψ is also continuous.

Conversely, let ψ be a continuous ω-translation. By Theorem 1 point 2 below, ψ is ITM/A-computable, and thus is interactively computable. □

The following theorem further shows that any possible ω-translation performed by some deterministic interactive system (according to the interactive paradigm described above) can actually be described in terms of interactive Turing machines with advice. This result supports the Church-Turing Thesis of Interactive Computation which states that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [12].

Theorem 3. *An ω-translation ψ is interactively computable iff it is ITM/A-computable.*

Proof. A direct consequence of Theorem 1 point 2 and Theorem 2. □

Finally, by putting together theorems 1 and 3, we obtain the following mathematical and machine-based characterisations of interactively computable ω-translations.

Theorem 4. *Let ψ be some ω-translation. The following conditions are equivalent:*

1. *ψ is interactively computable;*
2. *ψ is continuous;*
3. *ψ is ITM/A-computable*

6 Conclusion

In this paper, we provide a mathematical as well as a machine-based characterisation of the ω-translations performed by arbitrary deterministic interactive systems (theorems 2 and 3, summarised in Theorem 4).

Theorem 2 shows that the class of interactively computable ω-translations coincides with that of continuous ω-translations. It provides a generalisation of the results [10, Theorem 5] and [14, Theorem 8] to the case of arbitrary deterministic interactive systems, rather than those driven by a Turing program, or more precisely, as said in [14], those whose "behaviour can be effectively simulated, in the context of a simulation of any behaviour of their environment".

Note that [10, 14] also provides mathematical characterisations of interactively recognisable sets of inputs streams [14, Theorem 5 and Corollary 2] as well as interactively generated sets of output streams [14, Theorem 6], still in the context of deterministic interactive systems driven by a Turing program. For future work, we expect to study the generalisation of these results to our case of arbitrary deterministic interactive systems.

Theorem 3 shows that any computable ω-translation can be performed by some ITM/A. This results supports the Church-Turing Thesis of Interactive Computation which states that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [12].

Theorem 3 can be understood as follows: similarly to the classical context, where every possible function from integers to integers can be computed by some Turing machine with oracle [9], in the interactive context, every possible ω-translation performed in an interactive way can be computed by some interactive Turing machine with advice. Alternatively put, in the interactive context, the model of an interactive machine with advice exhausts the class of all possible ω-translations performed in an interactive way, exactly like in the classical context, the model of a Turing machine with oracle exhausts the class of all possible functions from integers to integers.

For future work, we expect to extend the present study to the case of non-deterministic interactive systems of any kind. In this context, the relation between an input stream and its corresponding output stream is no more functional, but relational, since a given input stream might be translated into different output streams. It would be of interest to provide a complete mathematical and machine-based characterisation of the so-called ω-relations (subsets of $\{0,1\}^{\leq\omega} \times \{0,1\}^{\omega}$) computed by non-deterministic interactive systems of any possible kind. We conjecture that the class of interactively ω-relations corresponds precisely to those performed by non-deterministic ITM/As.

Conjecture 1. An ω-relation is interactively computable iff it is non-deterministically ITM/A-computable.

Finally, we believe that interactive computation provides a suitable framework to describe many current notions of computation. We also think that the super-Turing model of a ITM/A introduced by van Leeuwen and Wiedermann captures in a particularly relevant way the behaviour of several moderns, natural, and bio-inspired computing systems. The present paper aims to make a step forward in the theoretical approach to interactive computation.

References

1. Jérémie Cabessa and Hava T. Siegelmann. The computational power of interactive recurrent neural networks. *Neural Computation*, 24(4):996–1019, 2012.
2. Jérémie Cabessa and Alessandro E. P. Villa. The super-turing computational power of interactive evolving recurrent neural networks. *Lecture Notes in Computer Science*, volume 8131, pp. 58–65. Springer-Verlag, 2013.
3. Dina Goldin. Persistent turing machines as a model of interactive computation. *Lecture Notes in Computer Science*, volume 1762, pp. 116–135. Springer-Verlag, 2000.
4. Dina Goldin, Scott A. Smolka, Paul C. Attie, and Elaine L. Sonderegger. Turing machines, transition systems, and interaction. *Inf. Comput.*, 194:101–128, 2004.
5. Dina Goldin, Scott A. Smolka, and Peter Wegner. *Interactive Computation: The New Paradigm*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
6. Dina Goldin and Peter Wegner. Principles of interactive computation. In , *Interactive Computation*, Dina Goldin, Scott A. Smolka, and Peter Wegner, Eds., pp. 25–37. Springer-Verlag, 2006.
7. Dina Goldin and Peter Wegner. The interactive nature of computing: Refuting the strong church–turing thesis. *Minds Mach.*, 18:17–38, 2008.
8. Dina Q. Goldin and Peter Wegner. The church-turing thesis: Breaking the myth. *Lecture Notes in Computer Science*, volume 3526, pp. 152–168. Springer-Verlag, 2005.
9. Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.
10. Jan van Leeuwen and Jiřì Wiedermann. On algorithms and interaction. *Lecture Notes in Computer Science*, volume 1893, pp. 99–113. Springer-Verlag, 2000.
11. Jan van Leeuwen and Jiřì Wiedermann. Beyond the Turing limit: Evolving interactive systems. *Lecture Notes in Computer Science*, volume 2234, pp. 90–109. Springer-Verlag, 2001.
12. Jan van Leeuwen and Jiřì Wiedermann. The Turing machine paradigm in contemporary computing. In *Mathematics Unlimited - 2001 and Beyond*, Björn Engquist and Wilfried Schmid, Eds., pp. 1139–1155. Springer-Verlag, 2001.
13. Jan van Leeuwen and Jiřì Wiedermann. The emergent computational potential of evolving artificial living systems. *AI Commun.*, 15:205–215, 2002.
14. Jan van Leeuwen and Jiřì Wiedermann. A theory of interactive computation. In *Interactive Computation*, Dina Goldin, Scott A. Smolka, and Peter Wegner, Eds., pp. 119–142. Springer-Verlag, 2006.
15. Jan van Leeuwen and Jiřì Wiedermann. How we think of computing today. *Lecture Notes in Computer Science*, volume 5028, pp. 579–593. Springer-Verlag, 2008.
16. Peter Wegner. Why interaction is more powerful than algorithms. *Commun. ACM*, 40:80–91, 1997.
17. Peter Wegner. Interactive foundations of computing. *Theor. Comput. Sci.*, 192:315–351, 1998.