Computation Beyond the Turing Limit

– A Summary of Siegelmann's Paper [Sie95] –

Dominik Schultes

20. May 2004

1 Introduction

Even Alan Turing himself thought about a machine that could break the barrier that had been set by the Church-Turing Thesis resting upon his concept of an universal Turing machine. His *O-machine* enhances a conventional Turing machine by adding an oracle in a black box. In addition to the normal operations of a Turing machine, this machine can ask the oracle questions (depending on the current state and letter under the head) and the oracle answers after one time step either 0 or 1. The Turing machine can use this advice for its computation. Turing showed that the O-machine is a super-Turing machine, i.e., it can compute functions that are uncomputable on a conventional Turing machine. However, he did not elaborate on the realization of such a black box. Hence, he did not disprove the Church-Turing Thesis because the proof that an O-machine is realizable is missing. [CP00]

A *Turing machine with advice* is similar to Turing's O-machine. The input of such a Turing machine is supplemented with an advice sequence w_n, which depends only on the length of the input n. The Turing machine can use this advice for its computation, like the O-machine used the answers of the oracle. This machine is more powerful than a conventional Turing machine. For example, the *unary halting problem*[1] can be solved using an advice sequence whose lengths is polynomial in n. The solution is trivial as there is only one input of length n so that the advice w_n can consist of one bit to decide if the program run on this input halts or not. If we loose the constraint that the advice has a polynomial length, we can compute *all* functions $f : \{0,1\}^* \to \{0,1\}$ because an advice sequence of length 2^n can encode the results for all possible inputs of length n. Of course, again, the problem is to realize such a machine as it is quite "difficult" to give the correct advice.

A common approach to break the Turing barrier bases on dealing with real numbers.[2] Such (not yet realized) machines are called *analog computers* or, in order to avoid confusion with already existing analog machines (such as differential analyzers), *real computers*. While a conventional computer deals only with discrete numbers (e.g., the natural numbers), a real computer can perform operations on real numbers in a single step.[3] Obviously, this is an enhancement of computability as the set of real numbers is uncountable (in contrast to the set of natural numbers).

In this context, one concrete possibility of breaking the Turing barrier is the measurement of a real-valued physical quantity up to an arbitrary precision [Cop00]. For instance, let us assume there is a physical quantity whose value corresponds to the halting probability Ω (for a given Turing machine and a given encoding scheme). In order to decide if a given program of size n halts, we just have to measure this quantity up to a precision of n bits, as the first n bits of Ω are sufficient to make this decision. [Cha90]

Another example for a computation with real numbers is the classical *analog recurrent neuronal network* (ARNN). However, the main focus in Siegelmann's paper is the model of *analog shift maps*.

2 Analog Shift Maps

The basic data structure is the *dotted sequence* $\dot{\mathrm{E}}$ over a finite alphabet E. It has the form $E^* \,.\, E^*$, where "." $\notin E$, i.e., there is exactly one dot (that does not belong to the alphabet E) and all other

[1]The unary halting problem is a special case of the halting problem, where a unary encoding is used for both the program and the input.

[2]Other common approaches base on the unlimited acceleration of the computation or on the addition of randomness.

[3]However, an important restriction still is that the input and output are finite, i.e., no irrational numbers are admitted as I/O, in order to comply with the modern theory of computability.

letters are in E. Both sequences on the left and the right hand side, can be finite or infinite so that the dotted sequence can be finite, (one-side) infinite, or bi-infinite.

A *shift map* $S^k : \dot{E} \to \dot{E} : (a)_i \to (a)_{i+k}, k \in \mathbb{Z}$, shifts the dot k places to the right.

An *analog shift map* Φ enhances this concept in the following way. First, the given dotted sequence a is replaced by the dotted sequence $G(a)$, which is defined by the function G. Second, each position i that is empty in the new sequence $G(a)$ is filled by the letter a_i of the original sequence. This is denoted by $a \oplus G(a)$ and ensures that the new sequence has the same length as the old sequence. Third, a conventional shift map is applied where the shift index k is defined by a function F depending on the original sequence a. To sum up, we have $\Phi : a \to S^{F(a)}(a \oplus G(a))$. Both functions F and G have a finite domain of dependence (DoD), i.e., only a finite dotted substring of the original sequence a determines the outcome. However, the domain of effect (DoE) of G may be (bi-)infinite. For instance, the mapping $G(0.0) = \bar{\pi}.\pi$ is valid.[4]

The program of such a machine is given by the analog shift map, i.e., by the functions F and G. The input is the initial dotted sequence. In each step, the analog shift map is applied on the current dotted sequence. If and only if a fixed point is reached, i.e., a dotted sequence a with $\Phi(a) = a$, the program terminates. The output is defined as the sequence on the right hand side of the dot. In order to comply with the constraint that the I/O has to be finite, only systems with a finite input sequence that stop either with a finite or a left infinite sequence – in both cases the output is finite – are regarded. Still, it can be shown that this is a super-Turing machine by proving that a Turing machine with advice can be simulated by an analog shift map in the following way.

The Turing machine is represented by a dotted sequence. The letter 0 is encoded as 10, while 1 is encoded as 11. This allows us to use 01 as a special left-end marker. Everything to the left of this marker is ignored because it is regarded as garbage. The letters between the marker and the dot correspond with the tape of the Turing machine on the left hand side of the head. A unary encoding of the state of the Turing machine follows directly after the dot: the number of 0's represents the state. This part of the sequence is terminated by a single 1. Then, the sequence is continued by the letter under the head and the letters on the right hand side of the head. Finally, infinitely many 0's (denoted by $\bar{0}$) represent the empty part of the tape. To sum up, we have

$$[\text{garbage} \mid \text{left-end marker} \mid \text{tape left} \;.\; \text{state} \mid \text{head} \mid \text{tape right} \mid \bar{0}].$$

The initial sequence is $[\bar{0} \;.\; q_1 \; x \; \bar{0}]$, where x is the finite input string. The advice (in reversed order) for all possible input lengths, is $w = < \dots \; w_3 \; w_2 \; w_1 >$. In the first step, G replaces the 0's of the left side of the dot by the infinite string w so that we get $[w \;.\; q_2 \; x \; \bar{0}]$. In the next steps, the advices $w_1, w_2, \dots, w_{n-1}$ are deleted and the left-end marker is set to the left of w_n so that we obtain [garbage | left-end marker | $w_n \;.\; p_1 \; x \; \bar{0}$]. Now, the simulation of the Turing machine with advice is straightforward.

3 Realization

The power of the analog shift map hides in the function G because it generates the advice for the Turing machine. Hence, the crucial question is how to realize such a function. Up to this point, the difficulty of giving the correct advice has only been shifted to the difficulty of implementing G, but, so far, no difficulty has been eliminated.

Siegelmann claims that a highly chaotic dynamical system, e.g., the motion of a particle in a three-dimensional potential, can be implemented in order to realize an analog shift map. The left and the right halves of the dotted sequence can be represented by x- and y-coordinates, the operations are implemented by mirrors that reflect the particle. For more details, the reader may refer to the original paper.

Siegelmann emphasizes that an infinite precision of the operations and of the measurement of the particle's position is *not* required, but that a linear precision (depending on the running time) is sufficient. Unfortunately, she does not elaborate on the difficulties of obtaining an arbitrary, but finite precision. For example, thermal noise has to be regarded for every machine that operates above absolute zero. Furthermore, limitations of the obtainable precision arising by quantum physics effects such as the Heisenberg uncertainty principle could prevent an implementation of the analog shift map. [Rea04]

Such questions have to be addressed because the Church-Turing Thesis can only be disproved by a realizable machine, not by a fictional one.

[4]Here, π denotes the sequence 31415... in base 2. Similarly, $\bar{\pi}$ denotes the left infinite sequence ...51413 in base 2.

References

[Cha90] G. J. Chaitin. *Information, Randomness & Incompleteness*, chapter "Undecidability & Randomness in Pure Mathematics", pages 307–313. World Scientific, 2nd edition, 1990. http://www.cs.auckland.ac.nz/CDMTCS/chaitin/belgium2.html.

[Cop00] J. Copeland. Some notional hypercomputers. http://www.cs.usfca.edu/www.AlanTuring.net/turing_archive/pages/Reference Articles/hypercomputation/SomeNotionalHypercomputers.html, 2000.

[CP00] J. Copeland and D. Proudfoot. Introduction to hypercomputation: Computing the uncomputable. http://www.cs.usfca.edu/www.AlanTuring.net/turing_archive/pages/Reference Articles/hypercomputation/Intro to Hypercomputation.html, 2000.

[Rea04] Real computation. http://en.wikipedia.org/wiki/Real_computer, 2004.

[Sie95] H. T. Siegelmann. Computation beyond the Turing limit. *Science*, 268:545–548, 1995.