The computational power of infinite precision measurements

Luis Filipe Barreiros Fonseca
Instituto Superior Técnico
September 10, 2020

In this dissertation we study the power of analogue-digital Turing Machines that use physical experiment as oracles, allowing them to making measurements of real numbers and thus breaking the Turing barrier. The physical experiment we consider is the smooth scatter experiment, which takes some time to return an answer depending on the query word. It is possible to limit the time the machine waits for the oracle by using a time schedule.

We analyze the influence of the nature of the time schedule when we set the vertex position to $1/2$, caracterizing these machines in terms of non-uniform complexity classes and completing the current proof of when the time schedule is a computable function. Furthermore, eventhough there are three types of communication protocols between the deterministic Turing machine and the experiment, we only study the infinite precision protocol and in particular, we introduce an alternative technique to proof the known lower bound for the computational power of such machines when using a time schedule $T(k) \in \Omega(k)$.

I. Introduction

The ideia of non computable sets is highly related to our notion of computers, afterall, they are defined by the lack of a computer programme that can decide them. But what if we could give a help to the computer and tell it that some set of words belongs or not to some set? By giving a non-computable set it becomes straight forward to show that these machines could decide more than an usual computer. We shall call this set of words an oracle and define an oracle Turing machine to be a standard Turing machine coupled with an oracle. These machines have an additional tape, the query tape, and an additional special state, the query state. When the machine enters the query state, it reads the word written in the query tape and performs a transition to either the state YES or the state NO, depending if the query word is an element in the oracle set or not. This consultation to a standard oracle is done in a single time step.

These oracles work as black boxes and in the first years after their idealization by Alan Turing, researchers questioned if there could be some computing model based in a current physical theory that could break the Turing barrier. In 1999 Hava Siegelmann introduced in the Analogue Recurrent Neural Network ($ARNN$) which can be considered as a first model of the brain and consists in a neural networks with real valued weights. Hava Siegelmann proved that if at least one non-rational weight is used, then this model can in fact decide non-computable sets*. This is of particular interest as

*If only rational weights are used, then the $ARNN$ is equivalent to an usual Turing machine.

every language over a finite alphabet can be codified into a real weight and we could use some mean of measurement to fetch that same real weight, bit by bit.

Since then the academic community introduced several theorys on whether this model could be in fact be implemented as the used activation function for the neurons has a discontinuous derivative. While some as Sielgelmann, Younger and Redd claim to have engineered an implementation of this model, others as Martin Davis in [1, 2] are very critic on any physical implementation of super-Turing systems, arguing that the reason why models such as the $ARNN$ are able to decide super-Turing sets is because they are provided with non-computable parameters in the first place (in the case of the $ARNN$ these parameters are the real weights). Furthermore, Davis claims that even if a computer could produce a non-computable sequence of natural numbers, its non-computability could not be verified, which is a problem related to the existence (or lack of) non-computable numbers in nature.

Beggs, Costa and Tucker offer a different computational paradigm in which Turing machines communicate with some physical device that aim to measure some unknown quantity (see [3–5]). In this model, the Turing machine uses as an oracle a physical experiment that is based in some theory of Physics and so we may call it a physical oracle, and similarly to oracle Turing machines, it is possible to execute queries over it. However, the fact that these physical experiments are ruled by the laws of Physics imply that these experiments take more than some fixed time to return an answer. Take for example a balance, if two weights are very different then the heaviest will go down almost immediately, but if the two weights are almost similar the heaviest will take a lot more time to go down the same distance. Thus, we must have some way of telling the machine how much time it will wait for the physical experiment before timing it out. This amount of time that the machine waits is given by a time constructible function that we will call the time schedule, which must strictly increase with the size of the query word. If the experiment does not halt before timing out, the machine cannot go to the YES state or the NO state, so we must add to the Turing machine a third state: the "timeout" state.

This computational model has therefore two main components: the Turing machine wich is the digital part, and the physical experiment which is the analogue part. This is why these machines are called analogue-digital Turing machines, or $ADTM$'s. Many experiments have been considered for the analogue part, and all of these experiments could be used to achieve the same results as they all are equivalent to the $ARNN$ model, however, in this thesis we consider the smooth scatter experiment. We call smooth scatter machine ($SmSM$) (see [6]) to an $ADTM$ equipped with this physical oracle.

This work has the following main contributions. First we conclude the proof of the lower bound for $AP(CI)$, which is the class of sets decided by a SmSM using unknown vertex position $y = 1/2$ and a total increasing computable function as time schedule. This is done by showing that when it comes to deciding recursive sets in polynomial time, recursive tally oracles have the same power as non-recursive tally oracles. This allow us to assume without loss of generality that the set used as oracle in the current proof is recursive. Afterwards we introduce a new technique to write advice functions which, although we still do not know if we can use it to prove new results, it is our hope that this

technique can be used to modify the current proofs in order to achieve $\mathtt{BPP}//\log\star$ as the upper bound for the power of Smooth Scatter machines working with an error-prone precision protocol but wihtout using explicitly the formula of the time schedule, which is an open problem to know whether or not its possible. Our motivation is that this new advice function is less dependent on the time schedule in the sense that it can be written given only the list of boundary numbers, unlike the former advice which needs information given in the formula of the time schedule to be written.

II. The Scatter Machine

The model of computation we use in this work is an $ADTM$ which is, by definition, composed by a digital component (the Turing machine), an analogue-component (the physical oracle) and the interface, responsible for the communication between the analogue and the digital components. The Turing machine must have a query tape to call the oracle and several special outcomes states, one for each possible outcome of the experiment we are considering as oracle.

A. The experiment

Several experiments have been consider, however, results regarding computation do not depend on the experiment so we choose to use the Scatter experiment as oracle, since it is very easy to understand. This experiment has two versions: the sharp and the smooth versions depicted in Figures 1 and 2 respectively.

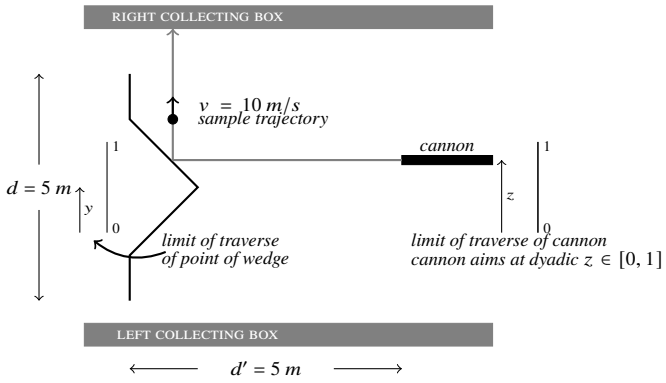

Fig. 1 Sharp Scatter Experiment

In both versions the vertex has a position $y \in]0,1[$ and a cannon placed at a position $z \in [0,1]$. The cannon shoots a particle which is reflected in the wedge and is collected by one of the boxes: if the particle hits the wedge above the vertex then it is collected by the right collecting box and the Turing machine enters state q_r, if the particle hits the wedge below the vertex then it is collected by the left collecting box and the Turing machine enters state q_l. The main difference between these two versions is that the sharp version takes constant intrinsic time which is explained by the non-analiticity of the wedge at the vertex position, while the intrinsic time of the smooth version increases as z approaches y. In fact, if the shape of the wedge is given by a function $g \in C^n$, the vertex is placed at y and the cannon

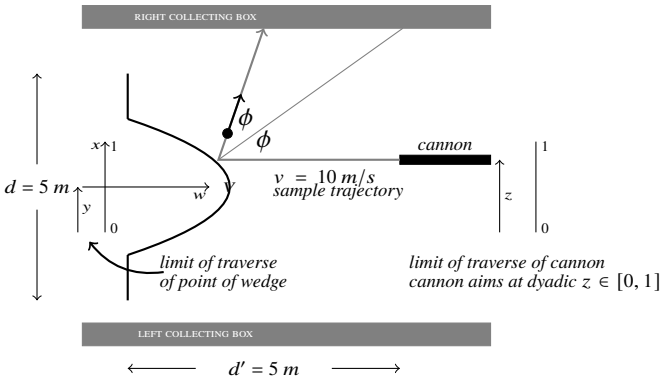

Fig. 2 Smooth Scatter Experiment

shoots at position z, it is shown in [6] that there exists constants A and B such that the time is bounded by

$$\frac{A}{|z-y|^{n-1}} \leq t_{exp}(z,y) \leq \frac{B}{|z-y|^{n-1}} \tag{1}$$

Furthermore, complexity wise, these constants have no influence on our results, so for simplicity we fix $A = B = 1$ and $n = 2$, defining completely the time as

$$t_{exp}(z,y) = \frac{1}{|z-y|}. \tag{2}$$

This result reinforces the BCT conjecture, which states that any feasible physical measurement experiment must take at least exponencial time on the desired precision.

B. The interface

The interface has two main components which are the time schedule and the communication protocol.

Definition 1 *A time schedule T is an increasing time-constructible function $T : \mathbb{N} \to \mathbb{N}$.*

Since the smooth version of the scatter experiment can take an arbitrarily large amount of time, we need the time schedule to make sure that the experiment does not run for more than polynomial time on the size of the input. We also constrain time schedules to be time-constructible functions so that the machine can count the time by itself. This is however a restriction that we will drop in Section III to see how the power of a SmSM changes according to the nature of the time schedule. Concerning the communication protocols, given a word q in the query tape and the real number $z = 0.q$, the cannon can be placed according to one three protocols.

Protocol 1 Error-free infinite precision protocol

The cannon position is set to the real number z (using an infinite precision).

Protocol 2 Error-prone arbitrary precision protocol

The cannon position is set to a real number in the interval $]z - 2^{-|q|}, z + 2^{-|q|}[$ *using a uniform distribution.*

Protocol 3 Error-prone fixed precision protocol

The cannon position is set to a real number in the interval $]z - \xi, z + \xi[$ *with uniform distribution, where* ξ *is a fixed positive real number.*

In the error-free protocol we take advantage that the cannon is placed exactly at z and so it is possible to perform a linear search to approximate the vertex position, as shown by algorithm 1, where we define by $z\downarrow_l$ as the l sized prefix of z, possibly by padding with 0's.

Algorithm 1

```
1:  procedure Linear Search Method
2:      l ← input                                   ▷ desired precision
3:      x_0 ← 0
4:      x_1 ← 1
5:      z ← 0
6:      while x_1 − x_0 > 2^{−l} do
7:          z ← (x_0 + x_1)/2
8:          s ← state after calling the oracle with z↓_l   ▷ It may be needed to pad z with 0's
9:          if s == "q_r" then
10:             x_1 ← z
11:         if s == "q_l" then
12:             x_0 ← z
13:         else
14:             x_0 ← z
15:             x_1 ← z
        return x_0
```

C. Simulating the experiment

In order for an ordinary Turing machine to simulate the scatter experiment, we need to give it the necessary information. We begin by defining the boundary numbers.

Definition 2 *Let* y *be the vertex position of a SmSM working with time schedule* T*. For a fixed query size* k*, we define the boundary numbers as* $0 < l_k, r_k < 1$*, such that* $l_k < y < r_k$ *and*

$$t_{exp}(l_k) = T(K) = t_{exp}(r_k). \tag{3}$$

Given $z = 0.q$ with size k and the prefixes of size k of l_k and r_k, which we define as $l_k\downarrow_k$ and $r_k\downarrow_k$ respectively, it is possible to decide in polynomial time in k whether or not $z \leq l_k$, but if $z = r_k\downarrow_k$, we cannon distinguish between $z = r_k$ and $z < r_k$ hence it is needed one more bit of information to separate these cases which we define as σ_k. This has

value 1 if $z = r_k$ and 0 otherwise. We can know formulate how much time the Turing Machine takes to simulate the experiment if given access to this information.

Proposition 1 *Take a SmSM with left boundary number l_k for some fixed k. If we have access to $l_k\downharpoonright_k$ then, for any query z with $|z| = k$ we can determine if the SmSM will perform a transition to state q_l with query z, in linear time on k.*

Proposition 2 *Take a SmSM with right boundary number r_k for some fixed k. If we have access to $r_k\downharpoonright_k$ and σ_k then, for any query z with $|z| = k$, we can determine if the SmSM will perform a transition to state q_r with query z, in linear time over k.*

III. Changing time schedules

We now study the influence of time schedules in the computational power of Scatter Machines working with infinite precision and vertex position $y = 1/2$ by not restricting time schedules to time-constructible functions, so we start by defining these classes of sets.

Definition 3 *Let f be an increasing total function. We denote by $AP(f)$ the class of sets decidable in polynomial time by an analogue-digital scatter machine using the physical oracle with time schedule f, infinite precision and unknown $y = 1/2$. If $\mathcal{F}$ is a class of increasing functions then we define $AP(\mathcal{F}) = \bigcup_{f\in\mathcal{F}} AP(f)$.*

Furthermore, we intend to relate the power of these Scatter Machines with non-uniform complexity classes, hence it is also necessary to introduce advice functions and non-uniform complexity.

Definition 4 *An advice function $f : \mathbb{N} \to \Sigma^\star$ is a total function which assigns a word for each (input size) $n \in \mathbb{N}$. A prefix advice function is an advice function f such that, for any $n, m \in \mathbb{N}$, if $n < m$, then $f(n)$ is a prefix of $f(m)$.*

Definition 5 *Let C be a class of sets and F a class of advice functions. We denote by $C/F\star$ the class of sets A for which there exist a set $B \in C$ and a prefix advice function $f \in F$ such that, for every $w \in \Sigma^*$, $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$.*

We now consider three classes of time schedules: total increasing functions (IN), total increasing computable functions (CI) and total increasing time-constructible functions (TC). The results are as follows:

Theorem III.1 $AP(IN) = \mathtt{P/poly}$.

Theorem III.2 $AP(CI) = \mathtt{P/poly} \cap REC$.

Theorem III.3 $AP(TC) = \mathtt{P}$.

It is worth noticing that by considering $y = 1/2$ we are taking some possibly non-computational variables out of the machine. In the case of Theorem III.1, it still might not be possible to compute the boundary numbers in polynomial time using Equation 3, for example, if we use a non-computable time-schedule. Hence the advice function still needs to include all the necessary information and so it is used the prefix advice function $f \in \texttt{poly}$:

$$f(n) = l_1\downarrow_1\#r_1\downarrow_1\#\sigma_1\#l_2\downarrow_2\#r_2\downarrow_2\#\sigma_2\#...\#l_n\downarrow_n\#r_n\downarrow_n\#\sigma_n. \tag{4}$$

However, in Theorem III.3 the time schedule must be a time-constructible function which implies that the running time of the Machine is bounded by some polynomial, hence the boundary numbers can be computed in polynomial time using Equation 3. This shows that the experiment itself does not add any power to the machine which justifies P as its computational power.

Regarding Theorem III.2, it was shown in [7] that we have $\texttt{P/poly} \cap REC$ as the upper bound, but we could only have this class as a lower bound on the condition that a set could be decided by a Turing machine running in polynomial time on help by a recursive tally oracle, remaining an open problem to know whether or not this lower bound remains without this assumption. In this work we show that indeed we can drop this constrain on the tally oracle by showing Proposition 3, thus finishing the proof of Theorem III.2.

Proposition 3 *If $A \in \texttt{P/poly} \cap REC$, then $A \in \texttt{P}(T')$ for some recursive tally set T'.*

IV. The infinite precision Smooth Scatter Machine

In this section we present the upper and lower bounds for the SmSM, where unlike the previous section we allow y to be any value in $]0, 1[$. Altough we only consider time-constructible time schedules, we will see that the upper bound for the power of such machines heavily depends on how fast the time schedule grows.

A. The Cantor set C_3

We define the Cantor set of base 3 (C_3) as the set of $\omega-$sequences x of the form

$$x = \sum_{k=1}^{+\infty} x_k 2^{-3k} \tag{5}$$

for $x_k \in \{1, 2, 4\}$. This can be seen as the set of numbers $0.z$ where z is obtained by concatenating infinitely many triples of the form $100, 010$ or 001. The fact that any number in the Cantor set can only be composed by these triples allows a nice property that is very useful for proving the lower bound.

Proposition 4 ***(Adapted from [8])*** *For every $x \in C_3$ and for every dyadic rational $z \in [0, 1]$ with size $|z| = m$, if $|x - z| \leq 2^{-(i+5)}$, then x and z coincide in the first i bits and $|x - z| > 2^{-(m+10)}$.*

The idea for proving the lower bound is to codify the advice function in an element $y \in C_3$ and consider a SmSM where the vertex position is y. In order to do this, we shall use the coding c where given a word w, $c(w)$ consists on replacing every 1 by 010 and every 0 by 100. Furthermore, we denote the encoding of a prefix advice function f by the real number $y(f) = \lim y(f)(n)$ where $y(f)(n)$ is defined recursively as follows[†] : $y(f)(0) = 0.c(f(0))$ and

$$y(f)(n+1) = \begin{cases} y(f)(n)c(s), & \text{if } n+1 \text{ is not a power of 2} \\ y(f)(n)c(s)001, & \text{if } n+1 \text{ is a power of 2} \end{cases} \tag{6}$$

where s is the word such that $(f)(n+1) = f(n)s$. It is important to notice that given a prefix of $y(f)$, it is easy to recover a prefix of the function f.

B. Bounds

A very important aspect of the SmSM is that the physical experiment takes exponential time in the level of desired precision (see Equation 2) but we require that the machine runs in polynomial, in other words, the experiment must run in polynomial time which is achieved by having query words at most of logarithmic size. Since the Turing machine only has to simulate the experiment for logarithmic sized queries, it is enough for the prefix advice functions to also have logarithmic size.

Theorem IV.1 *(**Lower Bound**) Let $A \in \mathtt{P/log}\star$, then there exists a SmSM $\mathcal{M}$ with infinite precision and time schedule $T(k) \in \Omega(2^k)$ that decides A when clocked in polynomial time.*

As mentioned, the lower bound is obtained by coding the advice function into a real number y in the Cantor set, considering a SmSM equipped with a physical oracle with y as vertex position and finally having the Turing machine querying the oracle to fetch the coding of the advice function, which will then be decoded.

Theorem IV.2 *(**Upper Bound**) If $A \subseteq \{0,1\}^*$ is decidable by an infinite precision SmSM with time schedule $T(n) \in O(2^n)$ when clocked in polynomial time, then $A \in \mathtt{P/log^2}\star$.*

In this case, we can simply include every prefix of the boundary numbers in the advice function which justifies the fact that there is some advice function $f \in \mathtt{P/log^2}\star$ in the conditions of the statement. If however the time schedule grows fast enough, we don't need the entire prefixes of the boundary numbers and so it is possible to get a smaller prefix advice function, and hence improving the upper bound.

Theorem IV.3 *(**Upper Bound**) Is A is decidable by a SmSM with infinite precision and exponential protocol $T(k) \in \Theta(2^k)$, clocked in polynomial time, then $A \in \mathtt{P/log}\star$.*

[†]This limit is defined not as function but as a sequence: $\forall n$, $y(f)(n)$ is a prefix of $y(f)$.

The advice function used in the proof of Theorem IV.3 (see [9]), needs information given by the formula of the time schedule to be written. We introduce a new type of prefix advice function which can be written even if we only have access to a list of the prefixes of the boundary numbers.

Let $T(k)$ be a time-constructible time schedule and r_k and l_k its right and left boundary numbers respectively for some vertex position y. For each $k \in \mathbb{N}$, we rewrite $l_k\downarrow_k = 0.x_k \cdot u_k$ and $r_k\downarrow_k = 0.v_k \cdot w_k$ where

- $x_1 = v_1 = \varepsilon$;[‡]
- for any $k > 1$, x_k is the biggest common prefix of $l_k\downarrow_k$ and $l_{k-1}\downarrow_{k-1}$;
- for any $k > 1$, v_k is the biggest common prefix of $r_k\downarrow_k$ and $r_{k-1}\downarrow_{k-1}$.

The prefix advice function is given by

$$f(n) = u_1\#w_1\#\sigma_1\#u_2\#w_2\#\sigma_2\#...\#u_n\#w_n\#\sigma_n\#. \tag{7}$$

It is shown in our work that this function has linear size (logarithmic if we only allow logarithmic sized query words) if and only if the SmSM operates with a time schedule $T(k) \in \Omega(2^k)$. Hence we can write Theorems IV.4 and IV.5.

Theorem IV.4 *(**Upper Bound**) If A is decidable by an infinite precision SmSM with vertex at position $y \in]0, 1[$ clocked in polynomial time and time schedule $T(k) \in \Omega(2^k)$, then $A \in \mathtt{P/log}\star$.*

Theorem IV.5 *There exists a set decided by a SmSM working with infinite precision and time schedule $T(k) \notin \Omega(2^k)$ bounded in polynomial time that cannot be decided by a deterministic Turing machine bounded in polynomial time receiving $\langle w, f(|w|)\rangle$ as input where f is a prefix advice function such that $|f(n)| \in O(n)$.*

Theorems IV.1 and IV.4 completely characterize the power of SmSM working with infinite precision and a time schedule $T(k) \in \Omega(2^k)$.

Theorem IV.6 *A set A is decidable by a SmSM clocked in polynomial time with infinite precision and exponential time schedule $T(k) \in \Omega(2^k)$ if and only if $A \in \mathtt{P/log}\star$.*

V. Conclusions

The main goal of this work is to see how can we break the Turing barrier, in this case, by coupling a physical oracle to the Turing machine as proposed by Alan Turing. Indeed, we can see that we can surpass this barrier by introducing some non-computable parameter in the Scatter Experiment. We study two techniques for doing so.

When we relax the definition of time schedules (see Section III), Theorems III.1 and III.2 show that if we use any total increasing function as time schedule, then it possible to decide $\mathtt{P/poly}$, and moreover, if we restrict the time

[‡]In this case we have $l_1\downarrow_1 = 0.u_1$ and $r_1\downarrow_1 = 0.w_1$.

schedules to computable functions, then it is possible to decide the recursive sets in P/poly. In both cases we have $ADTM$'s running in polynomial time deciding a superset of P.

In the case of having a time-contructible time schedule, we can place the vertex on a non-computable position (see Section IV) thus allowing the $ADTM$ to measure a non-computable value. Theorem IV.1 shows that the lower bound of these type of machines already has sets that are not in P. But perhaps the two most interesting results are the ones regarding the size of the prefix advice function according to the used time schedule: theorem IV.4 implies that if $T(k) \in \Omega(2^k)$, then it is always possible to write the necessary information in a linear sized advice function. Conversely, if $T(k) \notin \Omega(2^k)$, then Theorem IV.5 states that in some cases, the best we can do is to write an advice function with superlinear size.

References

[1] Davis, M., "The myth of hypercomputation," *Alan Turing: the life and legacy of a great thinker*, edited by C. Teuscher, Springer, 2006, pp. 195–212.

[2] Davis, M., "Why there is no such discipline as hypercomputation," *Applied Mathematics and Computation*, Vol. 178, No. 1, 2006, pp. 4–7.

[3] Beggs, E., Costa, J. F., Loff, B., and Tucker, J. V., "Computational complexity with experiments as oracles," *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, Vol. 464, No. 2098, 2008, pp. 2777–2801.

[4] Beggs, E., Costa, J. F., Loff, B., and Tucker, J. V., "Oracles and Advice as Measurements," *Unconventional Computation (UC 2008)*, Lecture Notes in Computer Science, Vol. 5204, edited by C. S. Calude, J. F. Costa, R. Freund, M. Oswald, and G. Rozenberg, Springer-Verlag, 2008, pp. 33–50.

[5] Beggs, E., Costa, J. F., and Tucker, J. V., "Physical experiments as oracles," *Bulletin of the European Association for Theoretical Computer Science*, Vol. 97, 2009, pp. 137–151. An invited paper for the "Natural Computing Column".

[6] Beggs, E., Costa, J. F., and Tucker, J. V., "The impact of models of a physical oracle on computational power," *Mathematical Structures in Computer Science*, Vol. 22, No. 5, 2012, pp. 853–879. Special issue on Computability of the Physical, Editors Cristian S. Calude and S. Barry Cooper.

[7] Beggs, E., Costa, J. F., Poças, D., and Tucker, J. V., "An Analogue-Digital Church-Turing Thesis," *International Journal of Foundations of Computer Science*, Vol. 25, No. 4, 2014, pp. 373–389.

[8] Beggs, E., Costa, J. F., Poças, D., and Tucker, J. V., "Oracles that measure thresholds: The Turing machine and the broken balance," *Journal of Logic and Computation*, Vol. 23, No. 6, 2013, pp. 1155–1181.

[9] Ambaram, T., Beggs, E., Costa, J. F., Poças, D., and Tucker, J. V., "An analogue-digital model of computation: Turing machines with physical oracles," *Advances in Unconventional Computing, Volume 1 (Theory)*, Emergence, Complexity and Computation, Vol. 22, edited by A. Adamatzky, Springer, 2016, pp. 73–115.