$BPP//\log\star$
Random Walks as Oracles

Pedro Cortez Rodrigues

Instituto Superior Técnico - Universidade de Lisboa

March 2015

Abstract

Coupling Turing machines with oracles is one of the techniques used to boost the computational power, or only the efficiency, of Turing machines.

In this thesis, we study the computational power of analogue-digital Turing machines, i.e., Turing machines coupled with physical experiments, which are treated as oracles. We consider a generic and abstract physical oracle, satisfying certain axioms, we analyse its computational power and we use it to abstract some of the work already done for analogue-digital Turing machines. We also prove that these oracles allow us to generalize the probabilistic Turing machine, comparing the obtained analogue-digital machine with the probabilistic machine that has real numbers as probability of transition.

Then, we concretize our generic oracle with the random walk experiment, presenting an interesting review of the experiment. After that, we bound the number of calls to the oracle, in order to study the inner structure of $BPP//\log\star$. We analyse the computational power of random walk machines with certain bounds on the number of oracle calls. We also prove an important result about non-uniform complexity classes for the probabilistic case. Finally, we present a hierarchy of complexity classes in $BPP//\log\star$, corresponding to the analogue-digital Turing machines with the number of oracle calls bounded by certain functions on the size of the input.

Keywords: Turing machines. Physical oracles. Non-uniform complexity. Random walks. $BPP//\log\star$.

1 Introduction

Coupling oracles to Turing machines is one of the methods used to boost the computational power or only to speed up the computations of standard Turing machines. Conventionally, an oracle is a set. However, in [1] and [3], Beggs et al. proposed that the classical oracle could be replaced by an analogue device. The main idea is to couple a Turing machine with a physical experiment. The Turing machine interacts with the oracle by means of the query tape, providing the device with the initial conditions needed to initialize the experiment (if any). This communication between the machine and the oracle, i.e., the whole process of transferring data to, and from, the analogue device is ruled by some protocol, that consumes time and may introduce some error in the system. There are three types of protocols that have been considered: infinite precision, unbounded precision and fixed precision.

As one can expect, the main difference between these analogue-digital Turing machines, or AD machines, and classical oracle machines is that, unlike the classical oracles, it takes time to consult a physical oracle, because a physical experiment is executed, so, the oracle consultation is not any more a single step computation, but it lasts a number of time steps that depend on the size of the query. Usually, a time schedule is defined, i.e., a time constructible function in the size of the query that establishes the time that the machine shall wait for the oracle computation to produce a result. Then, the finite control of the Turing machine changes to other state, according to the outcome of the oracle, and the computation proceeds.

In this work, we analyse the random walk experiment and construct a machine that uses it as an oracle, launching particles and receiving as result if they were, or not, absorbed during the scheduled time. We chose this experience, not to be just another analogue-digital machine such as the ones already analysed, but as an attempt to abstract all previously studied experiences, for the case of fixed precision. Leaving aside the Physics, there is no need to consider concrete results of physical science and we get an experiment from the field of Probability.

2 Stochastic oracles: an abstract model

Consider an abstract generic oracle, with two possible outcomes, e.g. 'yes' or 'no', such that no parameters are needed to be given in order to initialize the experiment. We couple this oracle with a Turing machine, so that the query consists in a simple trigger to start the experiment and, if the query time is less than the scheduled time, then the finite control of the Turing machine changes to the 'yes' state, otherwise the finite control changes to the 'no' state. Let σ denote the unknown parameter associated with the physical experiment. We establish some conditions on the machine, like a constant time schedule and some analytical properties of the distribution $F(\sigma)$ that corresponds to the probability of getting the outcome 'yes': $F \in C^1([0,1])$, $F'(\sigma) \neq 0, \forall \sigma \in [0,1]$ and we can compute n bits of F in time $O(2^n)$. We prove lower and upper bounds on the computational power of these AD machines.

Theorem 2.1. *Every set in $BPP//\log\star$ is decidable by an AD machine with 'yes' probability $F(\sigma)$ in polynomial time.*

To prove this lower bound theorem, as we are trying to establish as lower bound the non-uniform complexity class $BPP//\log\star$, we must find a way to encode the advice information into the parameter σ. We use an encoding based on the Cantor set $\mathcal{C}_3$, already used in [1] and [4].

$$\mathcal{C}_3 = \left\{ x \in \mathbb{R} : x = \sum_{k=1}^{\infty} x_k 2^{-3k}, \text{ where } x_k \in \{1,2,4\} \right\}$$

Let $A \in BPP//\log\star$ and let $\mathcal{M}$ be a probabilistic Turing machine with advice $f \in \log\star$ that decides A in polynomial time with error probability bounded by γ. Let $y(f)$ be the encoding of the advice function f. Then, we need to show that the AD machine with parameter $\sigma = y(f)$ can simulate, in polynomial

time, the behaviour of $\mathcal{M}$. So, our AD machine should be able to read the advice and to simulate a fair coin (for the probabilistic behaviour).

To read the advice, we make multiple calls to the oracle, doing a frequency analysis of the results, so that we can estimate the probability of getting the outcome 'yes'. Then, combining that result with the expression for that probability, $F(\sigma)$, we use some numerical methods to approximate σ. We obtain that the machine can read a number of bits of σ logarithmic on the size of the input, in polynomial time.

To prove that our machine can simulate a fair coin, we show that it can be seen as a biased coin and that, given a biased coin, we can produce a sequence of fair coin tosses in linear time on the size of the sequence.

As an upper bound for the classes computed by AD machines with 'yes' probability $F(\sigma)$, we also want a non-uniform complexity class. More, we want to make lower and upper bound coincide. So, we tried to set $BPP//\log\star$ as an upper bound.

Theorem 2.2. *Every set decided by an AD machine with unknown parameter σ and with 'yes' probability $F(\sigma)$ in polynomial time is in $BPP//\log\star$.*

To prove this result, we must use and explore some results of probabilistic trees. Our AD machines are composed by a deterministic Turing machine and a stochastic oracle. Thus, the computations are deterministic, except for oracle consultations. Therefore, the computations of an AD machine with 'yes' probability $F(\sigma)$ can be represented by a binary tree, where each node stands for one run of the experiment and its left and right sub-trees correspond to the two possible outcomes of the experiment. Since the oracle is stochastic, each branch has a probability associated with it (in this case, since both σ and the time schedule are fixed for the whole computation, all the left branches have equal probability, F(σ), while all the right branches have probability $1 - F(\sigma)$). We use probabilistic trees to show that a small variation in the probability of 'yes' does not affect the decision of the machine. Then, we conclude that a number of bits of the probability of 'yes' logarithmic in the size of the input suffices to preserve the decision of the machine (with a small variation of the error). So, we consider a probabilistic machine $\mathcal{M}$ and we set that number of bits as the advice function. A probabilistic machine can behave has a deterministic machine, thus, we only need to show that, using the advice, it can simulate the calls to the oracle. We can do this using the fair coin of the probabilistic machine, using fair coin tosses to simulate a biased coin with heads probability equal to the probability of 'yes'.

With these proposed lower and upper bounds, we get the following result:

Corollary 2.3. *The class of sets which are decidable in polynomial time by AD machines with unknown parameter σ and with 'yes' probability $F(\sigma)$ is exactly $BPP//\log\star$.*

3 Some concrete cases

The previous results abstract some of the previous work (in [1, 5, 4]) relative to Turing machines with physical experiments as oracles, both two-sided experiments and threshold experiments, in the case of fixed precision.

Here we present the broken balance experiment and the balance scale experiment, as examples of threshold and two-sided experiments, respectively.

The broken balance experiment (BBE) consists of a balance scale with two plates. In the right plate of the balance is placed a body with unknown mass, y, that we intend to measure. To do so, we place test masses z on the left plate of the balance. If $z < y$, then the plates will not move since the rigid block prevents the right plate from moving down; if $z > y$, then the left plate will move down; if $z = y$, the plates will not move. The scheme of this experiment is represented in Figure 1.

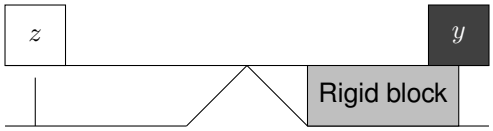

Figure 1: The broken balance experiment.

In the physical apparatus, a pressure-sensitive stick is placed below the left plate of the balance, such that, if the left plate moves down, it touches the pressure-sensitive stick and it reacts producing a signal.

A broken balance machine, or BB machine, is a Turing machine coupled with a BBE. The Turing machine interacts with the BBE using the query tape. To initialize the experiment, the Turing machine writes in the query tape the parameters for the experiment, namely the test mass. After the test mass is placed in the left plate (according to some precision protocol), the machine will wait $T(|z|)$ units of time, where T is the time schedule. If the pressure-sensitive stick produces a signal during the scheduled time, the oracle returns 'yes', otherwise, it returns 'timeout'.

Unlike the AD machines that we have studied in Section 2, in the case that we consider now, the Turing machine makes queries to the oracle, providing it with a dyadic rational, needed to initialize the experiment. This communication between the Turing machine and the physical oracle is ruled by one of the following protocols:

- infinite precision: when z is read in the query tape and a test mass $z' = z$ is placed in the left plate;
- unbounded precision: when z is read in the query tape and a test mass z' is placed in the left plate, where z' is randomly and uniformly chosen in $(z - 2^{-|z|}, z + 2^{-|z|})$;
- fixed precision ε ($\varepsilon > 0$): when z is read in the query tape and a test mass z' is placed in the left plate, where z' is randomly and uniformly chosen in $(z - \varepsilon, z + \varepsilon)$.

We present the lower bound theorem, for fixed precision case, that was already proved in [4, 10], but now its proof can be reconstructed, using the results obtained in Section 2.

Theorem 3.1. *If $A \in BPP//\log\star$ and $\varepsilon \in (0, 1/2)$, then A is decidable by a BB machine with fixed precision ε in polynomial time.*

The balance scale experiment (BSE) consists in trying to determine the mass y of a body placed in one plate of a balance, by means of approximations by another body, with dyadic rational mass z,

placed on the other plate. The test mass, z, can be bigger or smaller than y: if $z < y$, the plate that contains the unknown mass y will move down; if $z > y$, the plate that contains the test mass z will move down; if $z = y$, the plates will not move. The main difference from the broken balance experiment is that, instead of only observing a reaction if the test mass is bigger than the unknown mass, now we observe a reaction for every $z \neq y$. The scheme of the BSE is represented in Figure 2.

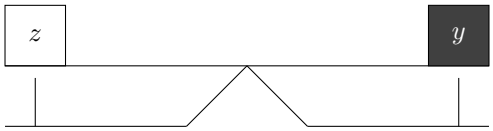

Figure 2: The balance scale experiment.

In the physical apparatus, a pressure-sensitive stick is placed below each plate of the balance, such that, if any plate moves down, it touches the corresponding pressure-sensitive stick and it reacts producing a signal.

A balance scale machine, or BS machine, is a Turing machine coupled with a BSE. The Turing machine interacts with the BSE using the query tape. To initialize the experiment, the Turing machine writes in the query tape the parameters for the experiment, namely the test mass. After the test mass is placed in the left plate (according to some precision protocol), the machine will wait $T(|z|)$ units of time, where T is the time schedule. If any pressure-sensitive stick produces a signal during the scheduled time, the oracle returns 'left' or 'right', according to the pressure-sensitive stick that had produced the signal, otherwise, it returns 'timeout'.

We present the lower bound theorem, for fixed precision case, which can be proved using the results obtained in Section 2.

Theorem 3.2. *If $A \in BPP//\log\star$ and $\varepsilon \in (0, 1/2)$, then A is decidable by a BS machine with fixed precision ε in polynomial time.*

4 Real numbers as probabilities of transition

After all the discussion about these AD machines, there is one question that arises: is it possible to remove the oracle and reduce everything to a probabilistic Turing machine with a probability of transition that is a real number? In this section , we will answer affirmatively to that question.

In [11], Hava Siegelmann presents the following definition of probabilistic Turing machines:

Definition 4.1 (Siegelmann [11])**.** A probabilistic Turing machine is a machine that computes as follows:

1. Every step of the computation can have two outcomes, one chosen with probability p and the other with probability $1 - p$;
2. All computations on the same input require the same number of steps;
3. Every computation ends with *reject* or *accept*.

Observe that, although every step of the computation can be made in exactly two possible ways, the machine can still take deterministic steps, if there is no difference in the corresponding actions of the two possible successor configurations.

In the classical definition of probabilistic Turing machine, p takes the value $1/2$.

To avoid confusion, we call generalized probabilistic Turing machine to the probabilistic Turing machine of Definition 4.1, i.e., the probabilistic machine such that p can take any real value in $(0, 1)$.

Our goal in this section is to prove that generalized probabilistic Turing machines are equivalent to the AD machines presented in Section 2.

Theorem 4.2. *Every set decided by an AD machine with unknown parameter σ and with 'yes' probability $F(\sigma)$ in polynomial time is decided by a generalized probabilistic Turing machine with bounded error probability in polynomial time.*

To prove the reverse statement, we have to restrict a little our universe of functions F. We consider only the functions that take all the values in $[0, 1]$, i.e., the functions F such that $F([0, 1]) = [0, 1]$.

Theorem 4.3. *Every set decided by a generalized probabilistic Turing machine with bounded error probability in polynomial time is decided by an AD machine with 'yes' probability $F(\sigma)$ in polynomial time.*

Using the results, we obtain the following important result about generalized probabilistic machines.

Theorem 4.4. *The class of sets which are decidable in polynomial time by generalized probabilistic Turing machines with bounded error probability is exactly $BPP//\log\star$.*

5 Random walks

Consider the experiment of having a particle moving along an axis. The particle is sent from position $x = 0$ to position $x = 1$. Then, at each positive integer coordinate, the particle moves right, with probability σ, or left, with probability $1 - \sigma$. If the particle ever returns to its initial position, $x = 0$, it is absorbed (it remains there).

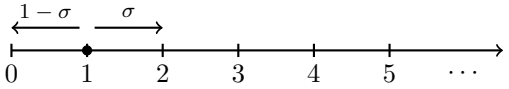

Figure 3: Random walk on the line with absorption at $x = 0$.

In this process, the particle takes steps of one unit, at time intervals also of one unit, thus, our experiment is a discrete unidimensional random walk with one absorbing barrier (see [12]).

We are interested in the probability that the particle is absorbed. We do as in [9]. Let p_i be the probability of absorption (at $x = 0$) when the particle starts at $x = i$. In our model, the particle is launched from $x = 0$ but it only starts its random walk at $x = 1$, so, we want to know the value of p_1. It is easy to see that

$$p_1 = (1 - \sigma) + \sigma p_2.$$

From $x = 2$, to be absorbed, the particle must initially move from $x = 2$ to $x = 1$ (not necessarily in one step), and then from $x = 1$ to $x = 0$ (again, not necessarily in one step). Both movements are made, independently, with probability p_1, thus, p_2 is just p_1^2. More generally, we have $p_k = p_1^k$. Therefore, the equation for the unidimensional random walk with absorption at $x = 0$ is given by

$$p_1 = (1 - \sigma) + \sigma p_1^2,$$

and we have a quadratic equation in p_1, with solutions $p_1 = 1$ and $p_1 = \frac{1-\sigma}{\sigma}$. For $\sigma = \frac{1}{2}$, the solutions coincide and $p_1 = 1$. For $\sigma < \frac{1}{2}$, the second solution is impossible, because $\frac{1-\sigma}{\sigma} > 1$, so, we must have $p_1 = 1$. For $\sigma = 1$, the particle always moves to the right, so $p_1 = 0$. Thus, for the sake of continuity of p_1, for $\sigma > \frac{1}{2}$, we must choose $p_1 = \frac{1-\sigma}{\sigma}$. Consequently, we get

$$p_1 = \begin{cases} 1 & \text{if } \sigma \leq \frac{1}{2} \\ \frac{1-\sigma}{\sigma} & \text{if } \sigma > \frac{1}{2} \end{cases}.$$

So, if $\sigma \leq \frac{1}{2}$, with probability 1 the particle always returns, but the number of steps is unbounded.

Now, let us consider a Turing machine coupled with a random walk experiment. This machine was introduced in [8]. To use the RWE as an oracle, we admit that the probability that the particle moves forward, σ, encodes some advice.

Unlike previously studied experiments, such as the balance experiments, both threshold and two-sided cases (see [4] and Section 3), the RWE does not need any parameters to be initialized, so, the Turing machine does not provide the oracle with any dyadic rational, it just "pulls the trigger" to start the experiment, i.e., the query corresponds to the impulse given to the particle at position $x = 0$ so it moves to $x = 1$. As we saw before, for every $\sigma \in (0, 1)$ the time that the particle takes to return is unbounded, thus, we need to provide a time schedule. If the particle is absorbed during the time schedule, the finite control of the Turing machine changes to the 'yes' state, otherwise, the finite control changes to the 'no' state. Defining the time schedule is an important and complex matter. For a fixed $\sigma \in (0, \frac{1}{2})$ (in this interval we know that the particle always returns), depending on the time schedule being too small or too long, we can have always the same outcome ('yes' or 'no'). For now, let us consider that the time schedule is constant.

Paths of the random walk along the positive x-axis with absorption at $x = 0$ are isomorphic to a specific set of well-formed sequences of parentheses. For instance, in a random walk of length 6, the particle could well behave as $((()))$ or $(()())$, where a movement to the right is represented by "$($" and a movement to the left is represented by "$)$". The first opening parenthesis corresponds to the first move of the particle from $x = 0$ to $x = 1$. The probability of answer in 6 steps is the sum of two probabilities corresponding to the two possible paths.

In [6], a generalization of Catalan numbers is used to derive the number of well-formed sequences of parentheses having the property that every nonempty, proper, initial segment has strictly more left than right parentheses. We use that result to obtain that the probability that the particle is absorbed during

the scheduled time T is given by

$$\sum_{\substack{t=1 \\ t\ odd}}^{T-1} \frac{1}{t} \binom{t}{\frac{t+1}{2}} (1-\sigma)^{\frac{t+1}{2}} \sigma^{\frac{t+1}{2}-1} .$$

This is the probability of getting the outcome 'yes' from the oracle. For analytical reasons, we will consider only $\sigma \in [\frac{1}{2}, 1]$, corresponding to a variation of p_1 from 1 to 0. Note that we could consider any interval contained in $[0, 1]$. For every T, this probability is a function of σ that satisfies the desired conditions: $F \in C^1([\frac{1}{2}, 1])$, $F'(\sigma) \neq 0, \forall \sigma \in [\frac{1}{2}, 1]$ and we can compute n bits of F in time $O(2^n)$. Therefore, RWE is an experiment with two possible outcomes, that does not need any initial information, with a constant time schedule and with the desired analytical properties of the probability function of the outcome 'yes'. Therefore, RW machines are in the conditions of our AD machines from Section 2. Applying to RW machines the results of Section 2, we obtain the following result:

Corollary 5.1. *The class of sets which are decidable in polynomial time by deterministic random walk machines is exactly $BPP//\log\star$.*

6 Counting the calls to the oracle

We bound the number of calls to the oracle, trying to study the inner structure of $BPP//\log\star$ and find a hierarchy of complexity classes in $BPP//\log\star$. With that goal in mind, we define the iterated logarithmic functions $\log^{(k)}(n)$:

- $\log^{(0)}(n) = n$;
- $\log^{(k+1)}(n) = \log(\log^{(k)}(n))$.

Denoting by $\log^{(k)}$ the class of advice functions f such that $|f(n)| \in O(\log^{(k)}(n))$, we use techniques similar to the ones used in Section 2 to prove the following:

Theorem 6.1. *The class of sets which are decidable in polynomial time by deterministic random walk machines that can make up to $O(\log^{(k)}(n))$ calls to the oracle, where n is the input size, is exactly $BPP//\log^{(k+1)}\star$.*

Thus, we see that random walk machines with a bounded number of oracle calls can induce a fine structure of $BPP//\log\star$.

Then, we present various definitions of concepts related to non-uniform complexity classes, such as the concept of reasonable advice functions, the relation $\prec$ between advice classes or the characteristic function of a tally set, in order to prove the important result:

Theorem 6.2. *If $\mathcal{F}$ and $\mathcal{G}$ are two classes of reasonable sublinear advice functions such that $\mathcal{F} \prec \mathcal{G}$, then $BPP//\mathcal{F} \subsetneq BPP//\mathcal{G}$.*

As we are considering prefix advice classes, it is useful to derive the following corollary:

Corollary 6.3. *If $\mathcal{F}$ and $\mathcal{G}$ are two classes of reasonable sublinear advice functions such that $\mathcal{F} \prec \mathcal{G}$, then $BPP//\mathcal{F}\star \subsetneq BPP//\mathcal{G}\star$.*

Since for all $k \geq 0$, $\log^{(k+1)} \prec \log^{(k)}$, we have the following infinite descending chain

$$\cdots \prec \log^{(4)} \prec \log^{(3)} \prec \log^{(2)} \prec \log .$$

Therefore, by Corollary 6.3, we have the descending chain of classes

$$\cdots \subsetneq BPP//\log^{(4)}\star \subsetneq BPP//\log^{(3)}\star \subsetneq BPP//\log^{(2)}\star \subsetneq BPP//\log\star,$$

that coincide with the sets decided by RW machines that that can make up to

$$\cdots \subsetneq O(\log^{(3)}(n)) \subsetneq O(\log^{(2)}(n)) \subsetneq O(\log(n)) \subsetneq O(n)$$

calls to the oracle, respectively, where n is the input size.

With the descendent chain of advice classes presented in [2] and [7], we can continue our descendent chain of non-uniform classes, although without knowing if there is a correspondence with the classes decided by RW machines:

$$BPP//\log^{(2\omega)}\star \subsetneq \cdots \subsetneq BPP//\log^{(\omega+1)}\star \subsetneq BPP//\log^{(\omega)}\star \subsetneq \cdots \subsetneq BPP//\log^{(2)}\star \subsetneq BPP//\log\star.$$

7 Future work

There are a lot of issues that we left behind, some more relevant than other.

First, once we used the random walk experiment to illustrate the AD machine that we presented, one can think of continuing to explore that experiment as an oracle to Turing machines, modifying some details. We can think about the behaviour of the oracle. Instead of returning only 'yes' or 'no' if the particle returns, or not, during the scheduled time, the Turing machine could receive from the analogue device the time that the particle took to arrive. Thus, we could not only do frequency analysis, but also other statistical and probabilistic reasoning. We can also consider modifying the experiment itself, and use high-dimensional random walks, as well as continuous random walks, and study if those experiments origin different results from the ones we obtained.

Thinking about the hierarchy that we presented, we said that the descent chain of non-uniform classes can be continued, considering $BPP//\log^{(\omega)}\star$, where $\log^{(\omega)} = \bigcap_{k\in\mathbb{N}} \log^{(k)}$ and that it can it be continued even further. However, we do not know if there is a correspondence between these complexity classes and the classes decided by RW machines with bounded number of oracle calls, since we only proved the correspondence for advice classes of the form $\log^{(k)}$, with $k \in \mathbb{N}$. One of the questions that we left for future thinking is how we could encode a function $f \in \log^{(\omega)}\star$ into a real number.

References

[1] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Computational complexity with experiments as oracles. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 464(2098):2777–2801, 2008.

[2] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Oracles and advice as measurements. In Cristian S. Calude, José Félix Costa, Rudolf Freund, Marion Oswald, and Grzegorz Rozenberg, editors, *Unconventional Computation (UC 2008)*, volume 5204 of *Lecture Notes in Computer Science*, pages 33–50. Springer-Verlag, 2008.

[3] Edwin Beggs, José Félix Costa, Bruno Loff, and John V. Tucker. Computational complexity with experiments as oracles II. Upper bounds. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 465(2105):1453–1465, 2009.

[4] Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. Oracles that measure thresholds: the Turing machine and the broken balance. *Journal of Logic and Computation*, 23(6):1155–1181, 2013. Special issue on The Incomputable, Editors S. Barry Cooper and Mariya I. Soskova.

[5] Edwin Beggs, José Félix Costa, and John V. Tucker. The impact of models of a physical oracle on computational power. *Mathematical Structures in Computer Science*, 22(5):853–879, 2012. Special issue on Computability of the Physical, Editors Cristian S. Calude and S. Barry Cooper.

[6] Andreas Blass and Gábor Braun. Random orders and gambler's ruin. *Electr. J. Comb.*, 12:R23, 2005.

[7] José Félix Costa. Incomputability at the foundations of physics (A study in the philosophy of science). *J. Log. Comput.*, 23(6):1225–1248, 2013.

[8] José Félix Costa. Uncertainty in time. *Parallel Processing Letters*, 25(1), 2015.

[9] Frederick Mosteller. *Fifty Challenging Problems in Probability with Solutions*. Dover Publications, 1987.

[10] Diogo Miguel Ferreira Poças. Complexity with costing and stochastic oracles. Master's thesis, Instituto Superior Técnico, July 2013.

[11] Hava T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, 1999.

[12] Salvador Elias Venegas-Andraca. *Quantum Walks for Computer Scientists*. Morgan and Claypool Publishers, 2008.