An Analogue-digital Model of Computation: Turing Machines with Physical Oracles

Tânia Ambaram[1], Edwin Beggs[2], José Félix Costa*[1], Diogo Poças[3], and John V Tucker[2]

[1] Department of Mathematics, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
taniambaram@gmail.com,fgc@math.ist.utl.pt,diogopocas1991@gmail.com
[2] College of Science, Swansea University, Singleton Park, Swansea, SA3 2HN, Wales, United Kingdom
e.j.beggs@swansea.ac.uk, j.v.tucker@swansea.ac.uk
[3] Department of Mathematics and Statistics, McMaster University, Hamilton, Ontario, L8S 4K1, Canada
pocasd@math.mcmaster.ca

Abstract. We introduce an abstract analogue-digital model of computation that couples Turing machines to oracles that are physical processes. Since any oracle has the potential to boost the computational power of a Turing machine, the effect on the power of the Turing machine of adding a physical process raises interesting questions. Do physical processes add significantly to the power of Turing machines; can they break the Turing Barrier? Does the power of the Turing machine vary with different physical processes? Specifically, here, we take a physical oracle to be a physical experiment, controlled by the Turing machine, that measures some physical quantity. There are three protocols of communication between the Turing machine and the oracle that simulate the types of error propagation common to analogue-digital devices, namely: infinite precision, unbounded precision, and fixed precision. These three types of precision introduce three variants of the physical oracle model. On fixing one archetypal experiment, we show to classify the computational power of the three models by establishing the lower and upper bounds. Using new techniques and ideas about timing, we give a complete classification.

1 Introduction

Loosely speaking, by an analogue-digital system we mean a system that makes physical measurements in the course of a digital computation; equivalently, we the term hybrid system may be used. The digital computation can be of any complexity, though some embedded systems can be modelled by hybrid systems based upon finite automata. Actually, many processes perform an analogue measurement that is used in a digital computation of some kind. For example, models of analogue computation also involve digital elements: this is found in the construction of the analogue recurrent neural net model (ARNN) (see [1])[4]; in the optical computer model (see [2]); and in the mirror system (see [3]).

* Corresponding author

[4] In the ARNN case, a subsystem of about eleven neurones (variables) performs a measurement of a *real-valued* weight of the network up to some precision and resumes to a computation with advice simulated by a system of a thousand rational neurones interconnected with integer and a very few rational weights.

Computational dynamical systems that are able to read approximations to real numbers also behave as analogue-digital systems: they perform digital computations, occasionally accessing some external values. At the moment of access, the "computer" executes some task on the analogue device, such as a test for a given value of a quantity. In the perfect platonic world, this test cab be performed with infinite or arbitrary precision, in the sense that the machine can obtain as many bits of the real number as needed; or, less ideally, with a fixed finite precision provided by the equipment in use.

To analyse the computational capabilities of analogue-digital systems, we introduce an abstract analogue-digital model of computation that couples Turing machines to oracles that are physical processes. Thus, we consider Turing machines with the ability of making measurements. The Turing machines considered are deterministic, but in fact they can use the oracle both to get advice and to simulate the toss of a coin.

Interacting with physical processes takes time. There are cost functions of the form $T : \mathbb{N} \to \mathbb{N}$ that gives the number of time steps to perform the measurements that the Turing machine allows. In weighing using a balance scale – as, indeed, in most measurements – the pointer moves with an acceleration that depends on the difference of masses placed in the pans. It does so in a way such that the time needed to detect a mass difference increases exponentially with the precision of the measurement, no matter how small that difference can be made. This means that the measurement has an exponential cost that should be considered in the overall complexity of the analogue-digital computation.[5]

A possible objection to such a model is that it is not limited in precision; for even if it is sufficiently precise, it sooner or later finds the obstacle of the atomic structure of matter. However, measurement has its own theoretic domain and can only be conceived as a limiting procedure; see [8,9,5,4,6]. It means that measurement is like complexity and can only be conceived asymptotically. Once we limit, in absolute terms, space or time resources of a Turing machine, its complexity vanishes, since all (now finite) sets can be decided in linear time and constant space[6]

A measurement can be fundamental or derived. Measuring distance is fundamental, but measuring velocity is derived. Commonly, according to Hempel [8], fundamental measurement is based on a partial order of comparisons that, taken to the limit, generates a real number. Comparisons in the sense of Hempel are based on events in the experimental setup. The most common measurement of some concept y – position, mass, electric resistance, etc. – consists of performing the experiment with a test value z, for which we could test one or both of the comparisons "$z < y$" and "$y < z$"; experiments allowing such comparisons are called two-sided.

Experiments in physics provide intuitions about abstract measurements, namely (a) that they result from comparisons, (b) that they have a cost, (c) that they come with errors, and (d) that they are stochastic. Coupling a Turing machine with a physical experiment, we construct a specific type of analogue-digital machine.

Since any oracle has the potential to boost the computational power of a Turing machine, the effect on the power of the Turing machine of adding a physical process is an interesting area to investigate. Do experiments add significantly to power to the Turing machine, and break the Turing Barrier? Does the power of the Turing machine vary with different experiments?

[5] In contrast, in the ARNN model, the time of a measurement is linear due to the fact that the activation function of each neuron is piecewise linear instead of the common analytic sigmoid.

[6] In the same way, we could also say that tapes of Turing machines can have as many cells as the number of particles in the universe, but in such a case no interesting theory of computability can be developed.

We classify computational power. In Section 2, from a number of physical experiments, the so-called *smooth scatter machine*, first seen in [13], is selected as representative of the analogue-digital of machines of interest. In Section 3, we summarise the analogue-digital model. Section 4, we begin by summarising the theory of non-uniform complexity classes, which allows us to formalise a real number value for a parameter, by encoding an advice function of some class $\mathcal{F}\star$, and reading it into the memory of the machine. The section continues with lower bound (starting subsection 4.3) and upper bounds (starting subsection 4.7).

The role of precision of data has been explicated in our earlier papers, but the role of precision in time has been less clear. The notion is subtle: there is time in the physical theory, time that manages the oracle queries, and the runtime of the computation. In addition to explaining the model and its properties, we develop some new techniques and results to explore the nature of timing. In Section 5, we use a technique, based on knowing time exactly, to lower the upper bounds.

All the concepts will be defined in due course. Among the results proved in this paper are:

Theorem 1. *The classification of the computational power of the analogue-digital model in the absense and presence of errors is as follows:*

1. *If B is decidable by a smooth scatter machine with infinite precision and exponential protocol, clocked in polynomial time, then $B \in P/\log^2\star$.*
2. *B is decidable by a smooth scatter machine with infinite precision and exponential protocol $T(k) \in \Omega(2^k)$, clocked in polynomial time, if, and only if, $B \in P/\log\star$.*
3. *If B is decidable by a smooth scatter machine with unbounded or fixed precision and exponential protocol, clocked in polynomial time, then $B \in BPP//\log^2\star$.*
4. *B is decidable by a smooth scatter machine with unbounded or fixed precision and exponential protocol, clocked in polynomial time, and having access to exact physical time if, and only if, $B \in BPP//\log\star$.*

Moreover, we will argue that such bounds are, to a large extent, really independent of the analogue system considered.

This paper offers an overview of the development of the analogue-digital model with technical details and new results. Here we have the first complete analysis of the analog-digital machine having access to two-sided measurements, including the full proofs of the lower and upper computational bounds for the polynomial time case.

2 Physical oracles

The substrate of a real computation is a physical process executed by a machine; the input data is in many cases obtained by a physical measurement process. During a computation, the machine may have access to external quantities. For example,an analogue-digital device controlling the temperature in a building "solves" differential equations while "calling" thermometers to check the values of the temperature through time.

This idea that both computation and data are in some way physical processes motivates the theoretical quest for the limits of computational power of a physical process. We propose the coupling of a Turing machine with a physical oracle. The idea is to replace the standard oracle, which is just a set, by a physical experiment that allows the machine to perform a measurement.

A Turing machine coupled with such an oracle becomes a hybrid computation model, having an analogue component — the experiment — combined with a digital component — the Turing

machine. As the standard Turing machine with oracle, the Turing machine with physical oracle will use the information given by the physical oracle in its computations.

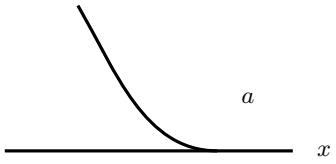


Fig. 1. Threshold Can only measure $a < x$. Can give a sequence of tests approximating x from below.

2.1 Types of physical oracles

The physical oracles that we will be considering are physical processes that enable the Turing machine to measure quantities. As far as we have investigated (see [7]), measurements can be classified in one of the three types: *one-sided* – also called *threshold measurement* – *two-sided measurement* and *vanishing measurement*.

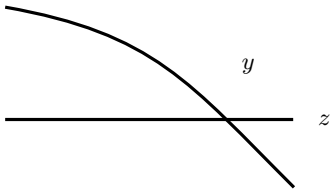


Fig. 2. Two-sided Measure both $y < z$ and $z < y$. A bisection method can be used to find y. Measure both $y < z$ and $z < y$. Assume continuity.

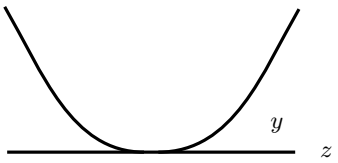


Fig. 3. Vanishing Can only measure ($y < z$ or $z < y$). A modified bisection method will work. Assume monotonicity on each side of y.

An one-sided experiment is an experiment that approximates the unknown value y just from one side, i.e., it approximates an unknown value y either with values z from above or with values z from below, checking either if $y < x$ or if $x < y$, but not both (see Figure 1).

A two-sided experiment is an experiment that approximates the unknown value y from two sides, i.e., it approximates the unknown value y with values z from above and with values z from below, checking if $y < z$ and $z < y$ (see Figure 2).

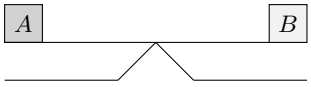
A vanishing experiment is an experiment that approximates the unknown value y from the physical time taken by the experiment (see Figure 3).

Note that this type of experimental classification is neither in Hempel's original work in [8], nor in the developed theory synthesised in [9], since, in a sense, these authors only consider the two-sided experiments.

In this paper we deal only with the two-sided measurement. Threshold experiments were considered in [10] and vanishing experiments in [11].

We now illustrate this category with a gallery of two-sided experiments of measurement, emphasising the experimental time.

Balance experiment The balance experiment is intended to measure the (gravitational) mass of some physical body. To perform the experiment we put the unknown mass m_A of body A in one of the pans and we approximate its value by placing another body, B, in the other pan. Body B can have a known mass, m_B, bigger or smaller than m_A. After placing the body B, the balance can display one of the behaviors explained in the right hand side of Figure 4.

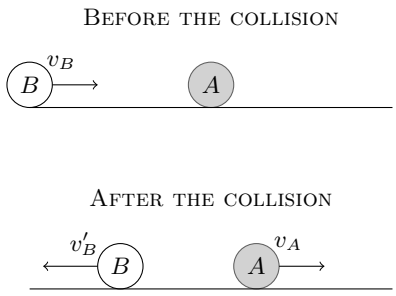


Input: The masses m_A (unknown) and m_B (test mass)
if $m_A > m_B$ **then** the pan containing A goes up;
if $m_A < m_B$ **then** the pan containing B goes down
else both pans will stand side by side;

Fig. 4. The balance experiment.

Accordingly to the behavior of the balance, we can change the mass of B in order to approximate m_A, using linear search. Repeating the experiment a number of times and considering bodies with masses each time closer to m_A we get the desired approximation of the unknown value. The time needed to detect a move in the pointer of the balance scale is exponential in the number of bits of precision of the measurement.

Elastic collisions The collision experiment can be used to measure the (inertial) mass of a body. This experiment was already studied as a physical oracle in [12]. To perform the experiment, we project a test body B with known mass m_B and velocity v_B along a line towards the body A at rest with unknown mass m_A. The mass of B, m_B, can be bigger or smaller than m_A. After the collision the bodies acquire new speeds, accordingly to their relative masses. After the collision we can have one of the behaviors explained in the right hand side of of Figure 5. According to the behavior of the particles, we can change the mass of B in order to approximate m_A, using linear search. Repeating the experiment a number of times and considering masses each time closer to m_A, we get an approximation of the unknown value. The time needed to detect a possible motion of body A is exponential in the number of bits of precision of the measurement.



Input: The masses m_A (unknown) and m_B (test mass)
if $m_A > m_B$ **then** the test body B moves backwards;
if $m_A < m_B$ **then** the test body B moves forwards;
else the test body B comes to rest and A is projected forward with speed v_B;

Fig. 5. The collision experiment.

The Foucault's pendulum The construction of this *gedankenexperiment* is based on the principles of classical mechanics (see Figure 6). The motion of the pendulum exhibits two coupled harmonic components:

1. A periodic motion of period
$$T = 2\pi\sqrt{\frac{\ell}{g}}\ ,$$
corresponding to the classical period of the pendulum of length ℓ for small amplitude oscillations.
2. Another periodic motion of period
$$\tau = \frac{24\ hours}{|\sin\lambda|}\ ,$$
corresponding to a complete rotation of the plane of oscillation at latitude λ. Moreover, the rotation of the plane of oscillation of the pendulum is *clockwise* to the north of equator and *counterclockwise* to the south of equator.

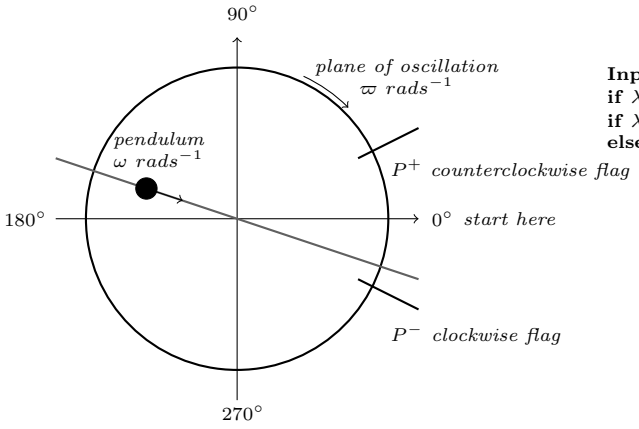
This two-sided experiment can be designed to locate the equator. The time needed for the pendulum to cross the angular distance of 1^s of arc is given by

$$t_\lambda = \frac{1}{15|\sin\lambda|}\ s\ ,$$

that, for small values of the angle λ, is of the order of

$$t_\lambda = \frac{1}{|15\lambda|}\ s\ ,$$

going to infinity as the angle λ approaches 0. In the case of a dyadic value λ, this time is exponential in the size of λ. The time needed to detect the equator is exponential in the number of bits of precision.



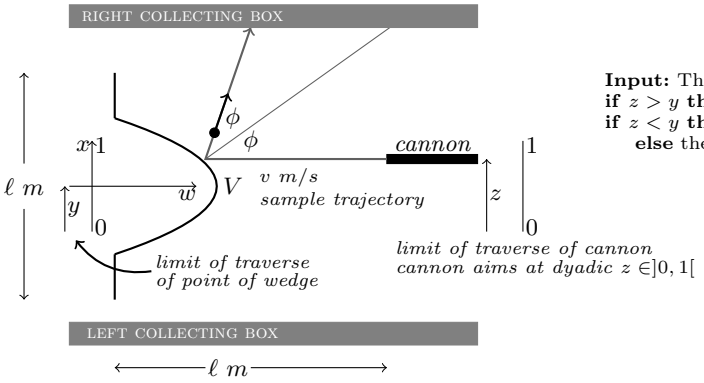
Input: The latitudes $\lambda_0 = 0^\circ$ (equator) and λ (test)
if $\lambda > 0$ **then** plane of oscillation crosses the flag P^+;
if $\lambda < 0$ **then** plane of oscillation crosses the flag P^-;
else the plane of oscillation does not rotate ;

Fig. 6. Foucault's machine experiment.

2.2 The smooth scatter experiment

We now introduce the measurement experiment that we take as a generic example throughout this paper.

The smooth scatter experiment (SmSE for short), considered in [13], is a variation of the sharp scatter experiment introduced in [14,15]. As the sharp scatter experiment, it measures the position of a vertex but it has a slightly different experimental apparatus. In the sharp case we considered a vertex in a sharp wedge and in the smooth case we are considering a vertex in a rounded wedge (see Figure 7), where the shape of the wedge is given by a smooth symmetric function.



Input: The position z
if $z > y$ **then** the particle is detected in the right box;
if $z < y$ **then** the particle is detected in the left box;
else the particle is not detected in any of box

Fig. 7. The smooth scatter experiment.

In order to measure the vertex position y, the smooth scatter experiment sets a cannon at some position z, shoots a particle from the cannon, and then waits some time until the particle is captured in a detection box. We consider that y can take any value in $]0,1[$. The fire and detection phases define one run of the experiment. After the shooting, we can have one of the behaviours explained in the algorithm of Figure 7 (right). By analyzing in which box the particle is detected, it is possible to conclude the relative position of y and z, i.e., whether the vertex is in the right or left side of the cannon. Then, repeating this procedure, by resetting the cannon position and executing one more run of the experiment, we can better approximate the vertex position.

The physical theory. The $SmSE$ is governed by a fragment of Newtonian mechanics, consisting in the following laws and assumptions:

1. Particles obey Newton's laws of motion in the two dimensional plane;
2. Collisions between barriers and particles are perfectly elastic, that is, kinetic energy is preserved;
3. Barriers are rigid and do not deform upon impact;
4. Cannon projects a particle with a given velocity and direction;
5. Detectors are capable of telling if a particle has crossed them; and
6. A clock measures the time.

The experimental clock that measures the physical time $\tau : [0,1] \to \mathbb{R}$ is very important in analysing the cost of accessing the oracle. When the Turing machine accesses the oracle it must wait until the end of the experiment in order to continue with its computation, so this means that accessing the oracle no longer takes one computation step but "t steps", where t is a function of the precision of the query.

The time The access to the $SmSE$ will cost more than one computation step as the physical experiment takes some intrinsic time t to be performed, designated as physical time. As explained in [13,16], the time required to run the $SmSE$ is given by the following proposition:

Proposition 1. *Consider that $g(x)$ is the function describing the shape of the wedge of a $SmSE$. Suppose that $g(x)$ is n times continuously differentiable near $x = 0$, all its derivatives up to $(n-1)$-th vanish at $x = 0$, and the n-th derivative is nonzero. Then, when the $SmSE$, with vertex position y, fires the cannon at position z, the time needed to detect the particle in one of the boxes is $t(z)$, where:*

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

for some $A, B > 0$ and for $|y - z|$ sufficiently small.

Looking at Equation 1, we conclude that we cannot bound the physical time, as it goes to infinity when z and y become close. Without loss of generality, we will assume from now on that $n = 2$, that is, $g(x) \in C^2$, since the results are essentially the same for values of $n > 2$.

3 The smooth scatter machine

We now fix the smooth scatter experiment as paradigmatic of two-sided measurement experiments and will proceed with the developing of our theory of analogue-digital computation.

A Turing machine coupled with the smooth scatter experiment as oracle is called a smooth scatter machine ($SmSM$). This machine was introduced in [13]. The $SmSM$ communicates with the $SmSE$ using the query tape. During a computation, the $SmSM$ performs its standard transitions and, whenever necessary, it consults the oracle. The oracle is an experiment that allows the Turing machine to measure an approximation to the vertex position of the scatter and use this value in the computation. In order to initialize the experiment, the Turing machine writes in the query tape the parameters for the experiment, e.g., the position of the cannon. After some time, the machine will be in other state according to the outcome of the experiment. To guarantee that our machine does not wait unbounded time for the answer, we will add to the Turing machine a *time constructible schedule* $T : \mathbb{N} \to \mathbb{N}$. The schedule depends on the size of the query.

Therefore, after some time not exceeding $T(n)$, where n is the size of the query, if the particle is detected in the right box, then the next state of the Turing machine is q_r; if the particle is detected in the left box, then the next state is q_l; and if the particle is not detected in any box, then the next state is q_t. The machine resumes the computation. The states q_r, q_l, and q_t replace the standard states *yes* and *no* of an oracle Turing machine. Note that, during the consultation of the oracle, the Turing machine waits for the oracle answer but keeps counting the running time in parallel.

3.1 Communicating with the smooth scatter experiment

The Turing machine communicates with the $SmSE$ when necessary. The communication is made through the query tape, where the Turing machine writes the parameters of the experiment, in this case the position of the cannon. We consider that the position provided by the Turing machine is written in binary and denotes a number of the form $n/2^k$, where n is a non-negative integer in $]0, 2^k[$ and k is a positive integer. The query z corresponds to the dyadic rational

$$z \;=\; \sum_{i=1}^{|z|} 2^{-i} z[i] \;, \tag{2}$$

where $z[i]$ denotes the i-th bit of z. After processing z, the apparatus should set the position of the cannon and execute the experiment. Note that by Equations 1 and 2, with the vertex at $y \in]0, 1[$, the experimental time grows exponentially with the size of the query, for values of z approaching y.

We consider that our experiment can set the cannon's position in three ways, inducing different computational powers. The three protocols are:

Protocol 1 *Given a dyadic rational z in the query tape, the experiment sets the position of the cannon to $z' = z$. In this case we are working with an error-free SmSE or with an infinite precision SmSE — the protocol is $Prot\_IP(z)$. A Turing machine coupled with this oracle is called an* error-free smooth scatter machine.

Protocol 2 *Given a dyadic rational z in the query tape, the experiment sets the position of the cannon to $z' \in [z - 2^{-|z|}, z + 2^{-|z|}]$, chosen uniformly. In this case we are working with an error-prone SmSE with unbounded precision — the protocol is $Prot\_UP(z)$. A Turing machine coupled with this oracle is called an* error-prone smooth scatter machine with unbounded precision.

Protocol 3 *Given a dyadic rational z in the query tape, the experiment sets the position of the cannon to $z' \in [z-\epsilon, z+\epsilon]$, chosen uniformly, for a fixed $\epsilon = 2^{-q}$, for some positive integer q. In this case we are working with an error-prone SmSE with fixed precision — the protocol is $Prot\_FP(z)$.*

A Turing machine coupled with this oracle is called an error-prone smooth scatter machine with fixed precision.

Algorithm 1: General communication protocol.

Data: Dyadic rational z (possibly padded with 0's)
Set cannon's position to $z' = Prot(z, mode)$;
Wait $T(|z|)$ units of time ;
if *The particle is detected in the right box* **then**
  A transition to the state q_r occurs ;
end
if *The particle is detected in the left box* **then**
  A transition to the state q_l occurs ;
end
if *The particle is not detected in any box* **then**
  A transition to the state q_t occurs ;
end

We can describe all the protocols in the Algorithm 1.

The procedure $Prot$ used in Algorithm 1 assigns a value to z' according with the protocol, i.e., using infinite precision ($Prot(z, inf) = z$), unbounded precision ($Prot(z, unb) \in [z-2^{-|z|}, z+2^{-|z|}]$), or fixed precision ($Prot(z, fix) \in [z - \epsilon, z + \epsilon]$).

Definition 1. *Let $A \subseteq \{0,1\}^\star$. We say that an error-free SmSM $\mathcal{M}$, clocked with runtime $\tau : \mathbb{N} \to \mathbb{N}$, decides A if, for every $w \in \{0,1\}^\star$, $\mathcal{M}$ accepts w in at most $\tau(|w|)$ steps if $w \in A$ and $\mathcal{M}$ rejects w in at most $\tau(|w|)$ steps if $w \notin A$.*

Definition 2. *Let $A \subseteq \{0,1\}^\star$. We say that an error-prone SmSM $\mathcal{M}$, with unbounded or fixed precision, clocked in runtime $\tau : \mathbb{N} \to \mathbb{N}$, decides A if there exists γ, $1/2 < \gamma \leq 1$, such that, for every $w \in \{0,1\}^\star$, $\mathcal{M}$ accepts w in at most $\tau(|w|)$ steps with probability γ if $w \in A$ and $\mathcal{M}$ rejects w in at most $\tau(|w|)$ steps with probability γ if $w \notin A$.*

3.2 Measurement algorithms

The measurement of the vertex position depends on the protocol, so that different protocols may originate different measurement algorithms. From now on we denote by $m\lfloor_\ell$ the first ℓ digits of m, if m has ℓ or more than ℓ digits, otherwise it represents m padded with k zeros, for some k such that $|m0^k| = \ell$. The pruning or the padding technique is used to control the time schedule during the measurement process. For the infinite precision we have the measurement Algorithm 2.

In Algorithm 2 we have no errors associated with the protocol since the cannon's position is always set to the desired position. Thus, doing this search around the vertex, we can approximate it up to any precision with no errors. The following result was already proved in [13] and [10].

Algorithm 2: Measurement algorithm for infinite precision.

Data: Positive integer ℓ representing the desired precision
$x_0 = 0$;
$x_1 = 1$;
$z = 0$;
while $x_1 - x_0 > 2^{-\ell}$ **do**
  $z = (x_0 + x_1)/2$;
  $s = Prot\_IP(z\downarrow_\ell)$;
  if $s ==$ "q_r" **then**
    $x_1 = z$;
  end
  if $s ==$ "q_l" **then**
    $x_0 = z$;
  else
    $x_0 = z$;
    $x_1 = z$;
  end
end
return *Dyadic rational denoted by* x_0

Algorithm 3: Measurement algorithm for unbounded precision.

Data: Positive integer ℓ representing the precision
$x_0 = 0$;
$x_1 = 1$;
$z = 0$;
while $x_1 - x_0 > 2^{-\ell}$ **do**
  $z = (x_0 + x_1)/2$;
  $s = Prot\_UP(z\downarrow_\ell)$;
  if $s ==$ "q_r" **then**
    $x_1 = z$;
  end
  if $s ==$ "q_l" **then**
    $x_0 = z$;
  else
    $x_0 = z$;
    $x_1 = z$;
  end
end
return *Dyadic rational denoted by* x_0

Proposition 2. *Let s be the result of $Prot\_IP(z\downarrow_\ell)$, y the vertex position of the SmSE, and T the time schedule. If $s =$ "q_l", then $z\downarrow_\ell < y$; if $s =$ "q_r", then $z\downarrow_\ell > y$; otherwise $|y - z\downarrow_\ell| < \frac{C}{T(\ell)}$, for some constant $C > 0$.*

Proof: The first two cases are obvious. In the third case we have

$$\frac{C}{|y - z\downarrow_\ell|} = t(z\downarrow_\ell) > T(\ell) \ ,$$

for some constant $C > 0$. Whence, $|y - z\downarrow_\ell| < \frac{C}{T(\ell)}$. □

Unbounded precision requires the measurement Algorithm 3.

In the Algorithm 3 we have errors associated with the protocol since the cannon's position is set to a value uniformly chosen in the closed interval $[z - 2^{-|z|}, z + 2^{-|z|}]$. But, according to the definition of $Prot\_UP$ (see Protocol 2), the interval that is considered could be increasingly smaller by increasing the precision, and so the error will be increasingly smaller too.

We proved in [10] and [14] the following result, where ℓ is such that $[z\downarrow_\ell -2^{-\ell}, z\downarrow_\ell +2^{-\ell}] \in]0, 1[$.

Proposition 3. *Let s be the result of $Prot\_UP(z\downarrow_\ell)$, y the vertex position of the $SmSE$, and T the time schedule. If $s =$ "q_l", then $z\downarrow_\ell < y + 2^{-\ell}$, if $s =$ "q_r", then $z\downarrow_\ell > y - 2^{-\ell}$, otherwise $|y - z\downarrow_\ell| < \frac{C}{T(\ell)} + 2^{-\ell}$, for some constant $C > 0$.*

Proof: Let z' be the position uniformly chosen by the $Prot\_UP(z\downarrow_\ell)$. Thus $z' \in [z\downarrow_\ell -2^{-\ell}, z\downarrow_\ell +2^{-\ell}]$ and $|z\downarrow_\ell - z'| \leq 2^{-\ell}$. From this it follows that, if $s =$ "q_l'", then $z' < y$ and therefore $z\downarrow_\ell < y + 2^{-\ell}$; if $s =$ "q_r'', then $z' > y$ and therefore $z\downarrow_\ell > y - 2^{-\ell}$; otherwise, $|y - z'| < \frac{C}{T(\ell)}$, for some constant $C > 0$. Therefore $|y - z\downarrow_\ell| < \frac{C}{T(\ell)} + 2^{-\ell}$. □

Proposition 4. *Given a $SmSM$ with unbounded or infinite precision, vertex position at y and time schedule $T(\ell) = C2^\ell$,*

1. *The time complexity for the measurement algorithm for input ℓ is $\mathcal{O}(\ell T(\ell))$;*
2. *The output of the algorithm, for input ℓ, is a dyadic rational z such that $|y - z| < 2^{-\ell+1}$.*

Proof: Statement (1) comes from the fact that the call to the protocol is repeated ℓ times, each taking $\mathcal{O}(T(\ell))$ steps. To prove statement (2) note that: (a) in the infinite precision case, the execution of the algorithm halts when $|y - z| < \frac{C}{T(\ell)}$; (b) in the worst case of the unbounded precision, the execution of the algorithm halts when $|y - z| < \frac{C}{T(\ell)} + 2^{-\ell}$, i.e., $|y - z| < 2^{-\ell+1}$; in both cases we have $|y - z| < 2^{-\ell+1}$. □

For the fixed precision protocol the measurement algorithm is different from the previous two.

Assume that initially we have a $SmSE$ with vertex at any real number $s \in]0, 1[$. We consider another $SmSE$ with vertex at position $y = 1/2 - \epsilon + 2s\epsilon$ and repeat ξ times the $Prot\_FP$ protocol with input $z' = 1$, i.e., with the vertex at position $1/2$. Consider a fixed precision $\epsilon = 2^{-q}$, for some $q \in \mathbb{N}$, and physical time given by the Equation 1. Consider also a Turing machine coupled with this $SmSE$ with a fixed schedule T. Then we will have three possible outcomes after running the experiment: q_r, q_l, or q_t. If η represents the position of the cannon where the time schedule exceeds the limit, for some precision ℓ, we have that the output q_r occurs for cannon positions in the interval $[y + \eta, 1/2 + \epsilon]$, the output q_t occurs for cannon positions in the interval $[y - \eta, y + \eta]$ and the output q_l occurs for cannon positions in the interval $[1/2 - \epsilon, y - \eta]$. Therefore we have that each outcome occurs with the following probability:

1. For the output q_r we have $p_r = (1/2 + \epsilon - y - \eta)/(2\epsilon)$.
2. For the output q_t we have $p_t = (y + \eta - y + \eta)/(2\epsilon)$.

3. For the output q_l we have $p_l = (y - \eta - 1/2 + \epsilon)/(2\epsilon)$.

So, our experiment can be modeled as a multinomial distribution with three categories of success, each one with the stated probabilities. Let α, β, and γ be random variables used to count outcomes q_l, q_r, and q_t, respectively, and consider a new random variable $X = \frac{2\alpha+\delta}{2\xi}$. The expected value of X is

$$\bar{X} = \frac{2E[\alpha] + E[\delta]}{2\xi} = \frac{-1 + 2y + 2\epsilon}{4\epsilon} = \frac{4s\epsilon}{4\epsilon} = s \ .$$

The variance of X is

$$Var(X) = \Big(\frac{1}{\xi}\Big)^2 Var(\alpha) + \Big(\frac{1}{2\xi}\Big)^2 Var(\delta) + 2\Big(\frac{1}{\xi}\Big)\Big(\frac{1}{2\xi}\Big) Covar(\alpha, \delta) \ .$$

Simplifying we get

$$Var(X) = -\frac{-4s\epsilon + 4s^2\epsilon + \eta}{4\epsilon\xi} \leq \frac{4s\epsilon(1-s)}{4\epsilon\xi} \leq \frac{1}{\xi} \ .$$

Using the Chebyshev's Inequality, for $\Delta > 0$, we get:

$$P(|X - \bar{X}| > \Delta) \leq \frac{Var(X)}{\Delta^2}$$
$$P(|X - s| > \Delta) \leq \frac{1}{\xi\Delta^2}$$

For precision $\Delta > 1/2^\ell$, we get

$$P(|X - s| > 1/2^\ell) \leq \frac{2^{2\ell}}{\xi} \ .$$

If we consider

$$\frac{2^{2\ell}}{\xi} < 2^{-h} \ ,$$

for $h \in \mathbb{N} - \{0\}$, then we get $\xi > 2^{2l+h}$.

Finally, we get the Algorithm 4 for the fixed precision case, where h is a positive integer chosen in order to get the error less than 2^{-h}.

Algorithm 4: Measurement algorithm for fixed precision.

Data: Integer ℓ representing the precision
$c = 0$;
$i = 0$;
$\xi = 2^{2\ell+h}$;
while $i < \xi$ **do**
  $s = Prot\_FP(1\lfloor_\ell)$;
  if $s ==$ "q_l" **then**
    $c = c + 2$;
  end
  if $s ==$ "q_t" **then**
    $c = c + 1$;
  end
  i++ ;
end
return $c/(2\xi)$

The following statements have been proved in [10].

Proposition 5. *For any real number $s \in]0, 1[$, fixed precision $\epsilon = 2^{-q}$ for $q \in \mathbb{N}$, error probability 2^{-h} for $h \in \mathbb{N} - \{0\}$, vertex position at $y = 1/2 - \epsilon + 2s\epsilon$, and time schedule T,*

1. *The time complexity of the measurement algorithm for fixed precision with input ℓ is $\mathcal{O}(2^{2\ell}T(\ell))$;*
2. *The output of the algorithm is a dyadic rational m such that $|y - m| < 2^{-\ell}$.*

Proof: Statement (1) derives from the fact that the procedure calls the oracle $2^{2\ell+h}$ times. Statement (2) is justified by observing that an approximation to $y \pm 2^{-\ell}$ with error probability less than 2^{-h} requires $2^{2\ell}/2^{-h}$ calls to the oracle. □

We are now ready to investigate the computational power of analogue-digital machines clocked in polynomial time, using (without loss of generality) the smooth scatter machine. Note that, on input ℓ, all the measurement algorithms output a value with ℓ bits of precision, or a value with the maximum number of bits of precision ($< \ell$) allowed by the schedule.

4 Computational power of the smooth scatter machine

The three types of $SmSM$ obtained in the Section 3 express different computational powers. In this section, we first recall some concepts of nonuniform complexity classes (see [17] and [18]).

4.1 Nonuniform complexity classes

An *advice function* is a total map $f : \mathbb{N} \to \{0, 1\}^\star$ and a *prefix advice function* is an advice function with the extra condition that $f(n)$ is a prefix of $f(n + 1)$. These functions have an important role in nonuniform complexity as they provide external information to the machines that depend only in the input size. We recall the definition of nonuniform complexity class.

Definition 3. *Let $\mathbb{C}$ be a class of sets and $\mathbb{F}$ be a class of advice functions. We define $\mathbb{C}/\mathbb{F}*$ as the class of sets B for which there exists a set $A \in \mathbb{C}$ and a prefix advice function $f \in \mathbb{F}$ such that, for every word $w \in \Sigma^*$ with size less or equal to n, $w \in B$ iff $\langle w, f(n)\rangle \in A$.*

We consider deterministic Turing machines clocked in polynomial time and logarithmic prefix advice functions (see [18]), obtaining the nonuniform complexity classes

$$P/\log\star \text{ and } P/\log^2\star.$$

We also consider probabilistic Turing machines clocked in polynomial time. Thus, based on the Definition 3, we get the following:

Definition 4. *$BPP/\log\star$ is the class of sets B for which a probabilistic Turing machine $\mathcal{M}$, a prefix advice function $f \in \log$ and a constant $\gamma < 1/2$ exist such that, for every word w with size less or equal to n, $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most γ if $w \in B$ and accepts $\langle w, f(|w|)\rangle$ with probability at most γ if $w \notin B$.*

The above definition is too restrictive forcing the advice function to be chosen after the Turing machine (see Appendix A). Thus we choose to use a more relaxed definition where the Turing machine is chosen after the advice function.

Definition 5. *$BPP//\log\star$ is the class of sets B for which, given a prefix advice function $f \in \log$, a probabilistic Turing machine $\mathcal{M}$ and a constant $\gamma < 1/2$ exist such that, for every word w with size less or equal to n, $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most γ if $w \in B$ and accepts $\langle w, f(|w|)\rangle$ with probability at most γ if $w \notin B$.*

Similarly we define the nonuniform class $BPP//\log^2\star$, just changing the class of advice functions.

At this point we know that the Turing machine has an oracle that measures approximations of the vertex position. Since we are talking about nonuniform complexity classes, the information given by the advice is codified in the position of the vertex of the $SmSE$.

4.2 The Cantor set $\mathcal{C}_3$

The Cantor set $\mathcal{C}_3$ is the set of ω-sequences x of the form

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

for $x_k \in \{1, 2, 4\}$. This means that $\mathcal{C}_3$ corresponds to the set of elements composed by the triples 001, 010, or 100. This set is often used to codify real numbers. For example, in [19], the Cantor codification with base 4 and 9 is used to codify the real weights of neural net.

This type of codification is required in order to be able to distinguish close values. For example, in order to describe the first bit of $011\cdots1$ and $100\cdots0$, we must read the whole number. To enforce gaps between close values, we encode the binary representations of the values in elements of the cantor set $\mathcal{C}_3$.

We will work with prefix advice $f : \mathbb{N} \to \{0,1\}^\star$, such that $f \in log$. We denote by $c(w)$, with $w \in \{0,1\}^\star$, the binary sequence obtained from the binary representation of w where each 0 is replaced by 100 and each 1 is replaced by 010. Given f, we denote its encoding as a real number by $y(f) = \lim y(f)(n)$ recursively defined as follows:

1. $y(f)(0) = 0.c(f(0))$;
2. If $f(n+1) = f(n)s$, then $y(f)(n+1) = \begin{cases} y(f)(n)c(s) & \text{if } n+1 \text{ is not power of 2} \\ y(f)(n)c(s)001 & \text{if } n+1 \text{ is power of 2} \end{cases}$.

By means of this definition, $y(f)(n)$ is logarithmic in n and the encoding function returns a word of size $\mathcal{O}(f(n))$. Note that separators are added only at positions that are a power of 2. If we want to decode $f(|w|)$, such that $2^{m-1} < |w| \leq 2^m$, we need to read $y(f)$ in triplets until we reach the $(m+1)$-th separator. To reconstruct $f(2^m)$, we eliminate the separators and replace each triplet for the corresponding value. Since $|c(f(2^m))| = am + b$, for some constants a and b, the number of binary digits needed to reconstruct $f(2^m)$ is linear in m.

Note that we considered a prefix advice function instead of an advice function, otherwise the encoding would not be logarithmic in the size of the input.[7]

As first proved in [10] and [14], these Cantor sets have the following property (see Appendix B):

Proposition 6. *For every $y \in \mathcal{C}_3$ and for every dyadic rational $z \in]0,1[$, such that $|z| = m$, if $|y - z| \leq 1/2^{i+5}$, then the binary expansion of y and z coincide in the first i bits and $|x - z| > 1/2^{-(m+10)}$.*

4.3 Lower bound for the infinite precision

We specify smooth scatter machines that decide the sets of some suitable nonuniform complexity class.

Theorem 2. *If $B \in P/\log\star$, then there exists an error-free $SmSM$, clocked in polynomial time, that decides B.*

Proof: If $B \in P/\log\star$, then there exists a set $A \in P$ (i.e.,A is decided by a deterministic Turing machine $\mathcal{M}_A$ clocked in polynomial time p_A) and a prefix advice function $f \in log$, such that $w \in B$ iff $\langle w, f(|\ w\ |)\rangle \in A$. We can thus compute the pairing of w and $f(|w|)$ in polynomial time p, and check in polynomial time p_A, using Turing machine $\mathcal{M}_A$, if such a pair belongs to A.

To get $f(|w|)$, $\mathcal{M}$ reads some binary places of the vertex position $y(f)$ using the $SmSE$ (Algorithm 2) with an exponential schedule $T(\ell) = C2^\ell$. Since $|w|$ may not be a power of 2 the $SmSM$ reads $f(n)$, where $n \geq |w|$ and $n = 2^{\lceil \log |w| \rceil}$. We have that $|f(|w|)| \leq a \log(|w|) + b$, for some constants $a, b \in \mathbb{N}$. Consequently $|f(n)| \leq a\lceil \log(|w|) \rceil + b$, so that $\mathcal{M}$ reads

$$\ell = 3(a\lceil \log(|w|) \rceil + b) + 3(\lceil \log(|w|) \rceil + 1)$$

binary places of the vertex, where $3(\lceil \log(|w|) \rceil + 1)$ denotes the number of bits used in the separators.

Using Algorithm 2 on $\ell + 5 - 1$, by Proposition 4, the algorithm returns a dyadic rational m such that $|y(f) - m| < 2^{-\ell-5}$. Hence, by Proposition 6, $y(f)$ and m coincide in the first ℓ bits.

Again, by Proposition 4, we know that the time complexity of the Algorithm 2 is $\mathcal{O}(\ell T(\ell))$. Since ℓ is logarithmic in the size of the input word and T is exponential in the size of the query word we find that the Algorithm 2 takes polynomial time p_a in the size of the input word. We conclude that the total time for the whole computation is $\mathcal{O}(p_a + p + p_A)$, that is polynomial in the size of the input. □

[7] We would get $y(f) = 0.c(f(0))001c(f(1))001\cdots$ and since each $c(f(i))$ has a logarithmic size in i, the sequence $y(f)(n)$ would have a size $\mathcal{O}(n \times log(n))$.

4.4 Smooth scatter machine as a biased coin

To prove the lower bounds for an error-prone $SmSM$ with unbounded or fixed precision we need some preliminary work. The first two statements – proved in [15], [12] and [13] – explain how the $SmSE$ can be seen as a biased coin. The third statement – already proved in [14] and [16] – state that, given a biased coin it is possible to simulate a sequence of fair coin tosses.

Proposition 7. *Given an error-prone smooth scatter machine with unbounded precision, vertex position at y, experimental time t, and time schedule T, there is a dyadic rational z and a real number $\delta \in]0,1[$ such that the outcome of $Prot\_UP$ on z is a random variable that produces q_l with probability δ.*

Proof: Consider an error-prone $SmSM$ with unbounded precision, vertex position y, experimental time t and time schedule T. Fix a positive integer ℓ such that $t(0) < T(\ell)$ and $t(1) < T(\ell)$, this means that if we run the experiment in the position $0\rfloor_\ell$ or if we run the experiment in the position $1\rfloor_\ell$, we will get an answer within $T(\ell)$ units of time.[8]

Given y and $T(\ell)$ we know that there exists $y' < y$ such that $t(y') = T(\ell)$. Consider now a dyadic rational z' such that $|z'| = \ell$ and $0 < z' - 2^{-\ell} < y' \leq z'$. Observe that $0 < y' < y < 1$ and so $0 < z' < 1$ (See Figure 8). The value of ℓ is fixed once and for all; however, it is supposed that ℓ can be fixed to a value large enough to get $z' - 2^{-\ell} > 0$. This restriction is easy to satisfy since we only require $t(0) < T(\ell)$ and $t(1) < T(\ell)$, which is obviously true for a large ℓ since the schedule is exponential.

If we run the $Prot\_UP$ on z', it will choose a value $z \in [z' - 2^{-\ell}, z' + 2^{-\ell}]$ uniformly. Thus, the probability that the protocol returns q_l is

$$\delta = \frac{y' - z' + 2^{-\ell}}{z' + 2^{-\ell} - (z' - 2^{-\ell})} = \frac{1}{2} - \frac{z' - y'}{2 \times 2^{-\ell}} \ .$$

Therefore $0 < \delta < 1$ and the probability that the protocol returns q_t or q_r is $1 - \delta$. □

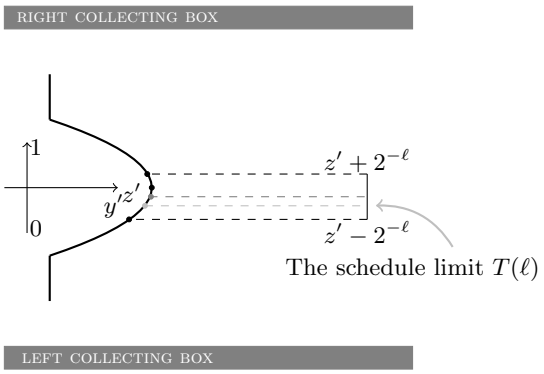


Fig. 8. The $SmSE$ with unbounded precision as a coin.

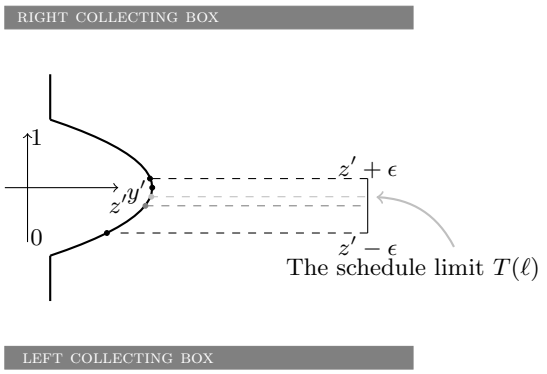


Fig. 9. The $SmSE$ with fixed precision as a coin.

[8] In this case the position $0\rfloor_\ell$ or $1\rfloor_\ell$ means $0.\underbrace{0\cdots0}_{\ell}$ and $1.\underbrace{0\cdots0}_{\ell}$, respectively, not the dyadic position.

Proposition 8. *Given an error-prone smooth scatter machine with fixed precision, vertex position at y, experimental time t, and time schedule T, there is a dyadic rational z and a real number $\delta \in]0,1[$ such that the outcome of Prot_FP on z is a random variable that produces q_l with probability δ.*

Proof: Consider an error-prone $SmSM$ with fixed precision ϵ, vertex position at y, experimental time t and time schedule T. Fix a positive integer ℓ that verifies the following conditions: $2^{-\ell} \leq \epsilon$, $t(0) < T(\ell)$, and $t(1) < T(\ell)$. The last restriction means that if we run the experiment with the cannon either at $0\rfloor_\ell$ or at $1\rfloor_\ell$, we will get an answer within $T(\ell)$ units of time.[9]

Given y and T we know that there exists $y' < y$ such that $t(y') = T(\ell)$. Consider now a dyadic rational z' such that $|z'| = \ell$ and $0 < z' \leq y' \leq z' + 2^{-\ell} < 1$ (Figure 8). The value of ϵ is fixed once and for all as is the value of y. However, it is supposed that ϵ can be fixed to a value that although fixed is very small, such that, $z' - \epsilon > 0$.

If we run the $Prot\_FP$ on z' it will choose a value $z \in [z' - \epsilon, z' + \epsilon]$ uniformly. Thus the probability of the protocol return q_l is

$$\delta = \frac{y' - z' + \epsilon}{z' + \epsilon - (z' - \epsilon)} = \frac{1}{2} - \frac{z' - y'}{2\epsilon} \ .$$

Therefore, $0 < \delta < 1$ and the probability of outcome q_t or q_r is $1 - \delta$. □

Proposition 9. *Given a biased coin with probability of heads $q \in]\delta, 1-\delta[$, for some $0 < \delta < 1/2$, and $\gamma \in]0,1[$, we can simulate, up to probability $\geq \gamma$, a sequence of independent fair coin tosses of length n by doing a linear number of biased coin tosses.*

See the proof in Appendix C.

4.5 Lower bound for the unbounded and fixed precisions

Theorem 3. *If $B \in BPP//\log\star$, then there exists an error-prone smooth scatter machine with unbounded precision, clocked in polynomial time, that decides B.*

Proof: If $B \in BPP//\log\star$, then there exists a prefix advice function $f \in \log$, a probabilistic Turing machine $\mathcal{M}$ working in polynomial time p_1, and a constant $\gamma_1 < 1/2$, such that, for every word w with size less or equal to $|w|$, $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most γ_1 if $w \in B$ and $\mathcal{M}$ accepts $\langle w, f(|w|)\rangle$ with probability at most γ_1 if $w \notin B$.

We compute $f(|w|)$ from $y(f)$ (the vertex of our $SmSM$) and use $\mathcal{M}$ to decide B. For this purpose consider γ_2 such that $\gamma_1 + \gamma_1\gamma_2 < 1/2$ and an exponential time schedule T.

By Proposition 7 there is a dyadic rational z, depending on y and T, that can be used to produce independent coin tosses with probability of heads $\delta \in]0,1[$. By Proposition 9, we can use the biased coin to simulate a sequence of fair coin tosses of size $p_1(n)$, with probability of failure γ_2.

Similarly to the proof of the Theorem 2, if $|w| = n$, using Algorithm 3 on input $\ell + 5 - 1$, for $\ell = 3(a\lceil\log(n)\rceil + b) + 3(\lceil\log(n)\rceil + 1)$, we can extract, by Proposition 4, a dyadic rational m such that $|y(f) - m| < 2^{-\ell-5}$. Hence, by Proposition 6, $y(f)$ and m coincide in the first ℓ bits, so we can use $\mathcal{M}$ to compute $f(2^{\lceil\log(n)\rceil})$.

[9] id.

Thus, after using the $SmSE$ as oracle in order to compute $f(2^{\lceil \log(n) \rceil})$, it is used as a generator of a biased coin and thus as a generator of a sequence of size $p_1(n)$ of random coin tosses. Then we just need to simulate $\mathcal{M}$ on $\langle w, f(2^{\lceil \log(n) \rceil}) \rangle$.

The behavior of the smooth scatter machine is the following:

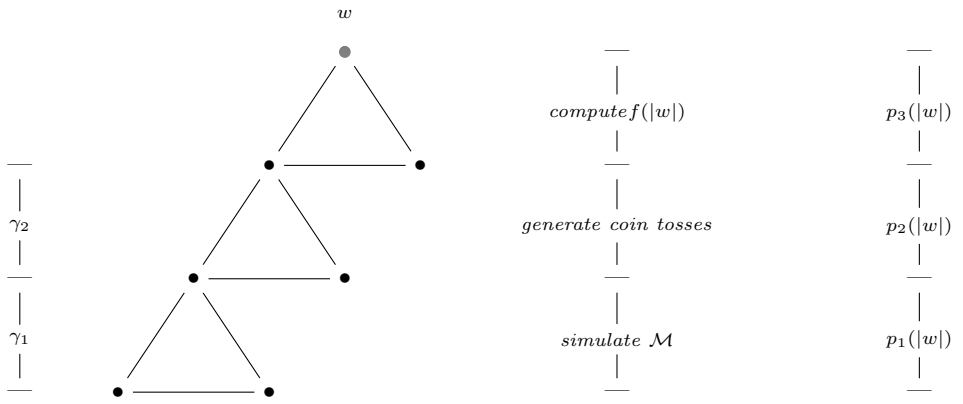


Fig. 10. Schematic description of the behavior of the $SmSM$ with unbounded precision.

Firstly the $SmSM$ uses the $SmSE$ to compute $f(2^{\lceil \log(n) \rceil})$; then it uses the $SmSE$ as a generator of a fair sequence of coin tosses (with error probability γ_2); finally the $SmSM$ uses this sequence to guide its computation on $\langle w, f(2^{\lceil \log(n) \rceil}) \rangle$.

Therefore, if $w \in A$, then the $SmSM$ rejects w if $\mathcal{M}$, guided by the sequence of coin tosses, rejects $\langle w, f(2^{\lceil \log(n) \rceil}) \rangle$, situation that happens with probability at most $\gamma_2\gamma_1 + \gamma_1 < 1/2$. Similarly, the probability that $\mathcal{M}$ accepts w, for $w \notin A$, is less than $1/2$.

We can conclude that the $SmSM$ decides A with an error probability less than $1/2$. To see that it decides A in polynomial time note that Proposition 4 states that the running time of the measurement algorithm is $\mathcal{O}(\ell T(\ell))$. Since ℓ is logarithmic in input size and T is exponential in ℓ, the result is polynomial in the size of the input. Let such time be denoted by p_3.

Let p_4 denote the time needed to encode the pair w with $f(|w|)$, that is $\mathcal{O}(|w| + |f(|w|)|)$, and that p_2 denote the polynomial time needed to generate the sequence of coin tosses. Then we can conclude that the total time for the whole computation is $\mathcal{O}(p_1 + p_2 + p_3 + p_4)$, which is a polynomial in the size of the input. □

Theorem 4. *If $B \in BPP//\log\star$, then there exists an error-prone smooth scatter machine with fixed precision ϵ, clocked in polynomial time, that decides B.*

Proof: The proof is similar to that of Theorem 3 but instead of Proposition 7 it uses Proposition 8; instead of using Proposition 4 it uses the Proposition 5; and instead of using measurement Algorithm 3 it uses Algorithm 4.

Note that in the fixed precision case the Algorithm 3 has an error γ_3. Thus, in this case, we consider γ_2 and γ_3 such that $\gamma_1 + \gamma_2 + \gamma_3 < 1/2$. In these conditions, if $w \in A$, then the $SmSM$

rejects w with probability at most $\gamma_3(\gamma_2\gamma_1 + \gamma_1) + \gamma_2\gamma_1 + \gamma_1 < 1/2$. Similarly, the probability that $w \notin A$ is accepted by $\mathcal{M}$ is less than $1/2$. □

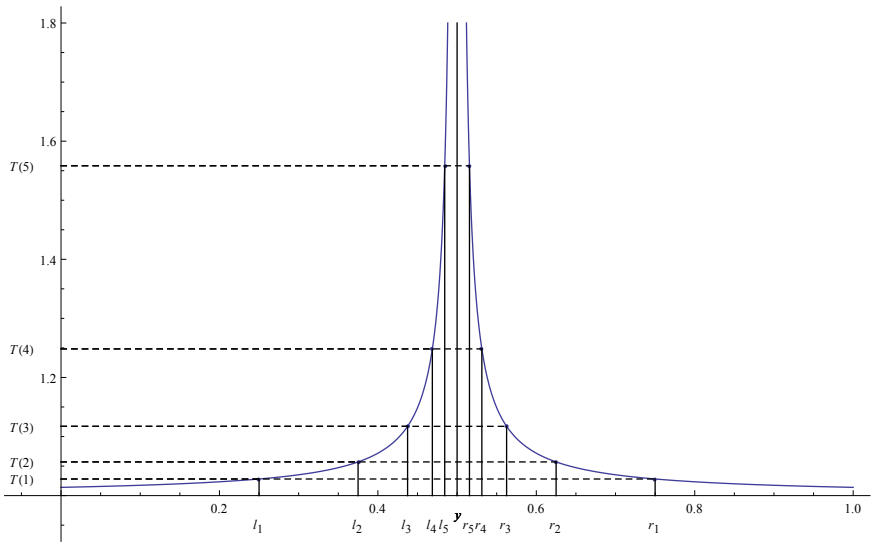


Fig. 11. The boundary numbers.

4.6 Boundary numbers

For the purpose of proving upper bounds, we introduce the *boundary numbers*.

Definition 6. *Let $y \in]0,1[$ be the vertex position and T the time schedule for a smooth scatter machine. For every $z \in \{0,1\}^\star$, we define $l_{|z|}$ and $r_{|z|}$ as the two real numbers in $]0,1[$ that satisfy the equation $t(l_{|z|}) = t(r_{|z|}) = T(|z|)$, with $l_{|z|} < y < r_{|z|}$.* [10]

Suppose that we want to query the oracle with z. If we query the $SmSE$, with vertex at y, we have three possible answers: q_l, if $z < y$ and $t(z) \leq T(|z|)$; q_r, if $z > y$ and $t(z) \leq T(|z|)$; q_t otherwise. We thus arrive to the Algorithm 5.

Therefore we may replace oracle consultation by a comparison of the query word (dyadic rational) with both $l_{|z|}$ and $r_{|z|}$. As we are going to see, in a sense to be precise later on, knowing enough bits of the boundary numbers is such like querying the oracle.

[10] There are always two boundary numbers satisfying this equation, as Figure 11 shows.

Algorithm 5: Oracle simulation.

Data: Dyadic rational z representing the query, boundary numbers $l_{|z|}$ and $r_{|z|}$
if $z \leq l_{|z|}$ **then**
  A transition to the state q_l occurs ;
end
if $z \geq r_{|z|}$ **then**
  A transition to the state q_r occurs ;
end
if $l_{|z|} < z < r_{|z|}$ **then**
  A transition to the state q_t occurs ;
end

4.7 Upper bound for the infinite precision

The precision we can get on the measurement of the vertex position determines the computational bounds of the smooth scatter machine. So far, the time schedule did not interfere in establishing the computational bounds. Now we apply the simulation technique, replacing the oracle by an advice function and prove upper bounds that, this time, seem to be sensible to the time schedule.

Theorem 5. *If B is decidable by a smooth scatter machine with infinite precision and exponential protocol, clocked in polynomial time, then $B \in P/\log^2\star$.*

Proof: Suppose that B is decidable by a $SmSM$ $\mathcal{M}$ with infinite precision and exponential time schedule T, clocked in polynomial time. Since T is exponential and $\mathcal{M}$ is clocked in polynomial time, we conclude that the size of the oracle queries grows at most logarithmically in the input size. This means that for any word w with size n, there exist constants a and b such that, during the computation, $\mathcal{M}$ only queries the oracle with words of size less or equal to $\ell = a\lceil\log(n)\rceil + b$. We consider a prefix advice function f such that $f(n)$ encodes the concatenation of boundary numbers needed to answer to all the queries of size ℓ:

$$l_1\rfloor_1 \# r_1\rfloor_1 \# l_2\rfloor_2 \# r_2\rfloor_2 \# \cdots \# l_\ell\rfloor_\ell \# r_\ell\rfloor_\ell \#.$$

The prefix advice function f is such that $|f(n)| \in \mathcal{O}(2\ell + 2\sum_{i=1}^{\ell} i) = \mathcal{O}(\ell^2) = \mathcal{O}(\log^2(n))$.

Therefore, to decide B in polynomial time, with prefix advice $f \in \log^2$, we simulate $\mathcal{M}$ on the input word but whenever a transition to the query state occurs and z is written in the query tape, we compare the query with the boundary numbers relative to $|z|$), i.e., with $l_{|z|}$ and $r_{|z|}$. The comparison, as explained in Section 4.6, gives us the same answer as $Prot\_IP(z)$, and thus the machine uses a similar measurement algorithm to approximate the vertex, replacing the call to the physical oracle by a comparison of the query word with $f(n)$. Since the comparison explained in Algorithm 5 can be done in polynomial time and $\mathcal{M}$ runs in polynomial time too, we can decide B in polynomial time given the advice in $P/\log^2\star$. □

If we analyze better the binary expansion of each boundary number, we can change the advice function considered in the previous proof, which is quadratic in the size of the query word, in order to obtain an advice function linear in the size of the query word. The main idea of this result is based on the fact that the boundary number $l_{|z|+1}$ ($r_{|z|+1}$) can be obtained from $l_{|z|}$ ($r_{|z|}$, respectively), by adding a few more bits of information.

Proposition 10. *Given the boundary numbers for a smooth scatter machine with time schedule $T(k) \in \Omega(2^k)$ it is possible to define a prefix advice function f such that $f(n)$ encodes all the boundary numbers with size up to n and $|f(n)| \in \mathcal{O}(n)$.*

Proof: Consider a $SmSM$ with vertex at y. If the time schedule associated with the $SmSM$ is $T(k) \in \Omega(2^k)$, then there exist α and k_0 in $\mathbb{N}$ such that for all $k \geq k_0$, $T(k) \geq \alpha 2^k$.

The value of the boundary number r_k is such that $y < r_k < y + 2^{-k+c}$, for some constant $c \in \mathbb{N}$ and for $k > k_0$. This means that, when we increase the size of k by one bit, we also increase the precision on y by one bit. Let us write the dyadic rational $r_k\downarrow_k$ as the concatenation of two strings, $r_k\downarrow_k = v_k \cdot w_k$, where w_k has size c and v_k has size $k-c$. Note that $r_k - 2^{-k+c} < v_k < r_k$, i.e.,$y - 2^{-k+c} < r_k - 2^{-k+c} < v_k < r_k < y + 2^{-k+c}$, i.e., $|v_k - y| < 2^{-k+c}$.

The same reasoning applies to $l_k\downarrow_k = x_k \cdot y_k$, i.e., $|x_k - y| < 2^{-k+c}$, where y_k has size c and x_k has size $k-c$.

We show that we can obtain v_{k+1} from v_k with just two more bits. Suppose that v_k ends with the sequence $v_k = \cdots 10^\ell$. The only two possibilities for the first $k-c$ bits of y are $\cdots 10^\ell$ or $\cdots 01^\ell$. Thus, v_{k+1} must end in one of the following: $v_{k+1} = \cdots 10^\ell 1$ or $v_{k+1} = \cdots 10^\ell 0$ or $v_{k+1} = \cdots 01^\ell 1$ or $v_{k+1} = \cdots 01^\ell 0$. That is, *even though v_k is not necessarily a prefix of v_{k+1}*, the latter can be obtained from v_k by appending some information that determines which of the four possibilities is the case.[11] Suppose now that v_k ends with the sequence $v_k = \cdots 01^\ell$. The only two possibilities for the first $k-c$ bits of y are $\cdots 01^{\ell-1}1$ or $\cdots 01^{\ell-1}0$. Thus, v_{k+1} must end in one of the following: $v_{k+1} = \cdots 01^\ell 0$ or $v_{k+1} = \cdots 01^\ell 1$ or $v_{k+1} = \cdots 01^{\ell-1}00$ or $v_{k+1} = \cdots 01^{\ell-1}01$. In the same way, v_{k+1} can still be obtained from v_k by appending some information that determines which of the four possibilities is the case.

Similarly we obtain x_{k+1} from x_k.

We define the advice function inductively as follows: if $n < k_0$, then $f(n) = l_1\downarrow_1 \# r_1\downarrow_1 \# l_2\downarrow_2 \# r_2\downarrow_2 \# \cdots \# l_n\downarrow_n \# r_n\downarrow_n$; if $n = k_0$, then $f(k_0) = f(k_0 - 1) \# \# x_{k_0} \# y_{k_0} \# v_{k_0} \# w_{k_0}$; and if $n > k_0$, then $f(n) = f(n-1) \# b_{11} b_{12} \# y_n \# b_{21} b_{22} \# w_n$, where the b_{ij}'s denote the bits that distinguish between x_{n-1} and x_n and between v_{n-1} and v_n.

Considering $f(n)$, we can always recover $r_k\downarrow_k$ or $l_k\downarrow_k$, for $k \leq n$, because, if $k \leq n < k_0$, then the values of $r_k\downarrow_k$ and $l_k\downarrow_k$ are explicitly in $f(n)$; if $k = n = k_0$, then these values are explicitly in $f(n)$ and, moreover, the machine knows it is the last fully given boundary numbers $r_k\downarrow_k$ and $l_k\downarrow_k$ (with the two ##); and finally, if $n \geq k > k_0$, to obtain l_k and r_k after knowing l_{k-1} and r_{k-1} we only need to recalculate the two final bits of x_k and v_k and concatenate the result with either y_k or w_k, respectively.

To conclude, since y_n and w_n have constant size c, the value of $|f(n)|$ is asymptotically linear in n. □

With this different encodings we obtain a different upper bound for the infinite case.

Theorem 6. *If B is decidable by a smooth scatter machine with infinite precision and exponential protocol $T(k) \in \Omega(2^k)$, clocked in polynomial time, then $B \in P/\log\star$.*

Proof: Suppose that B is decidable by a $SmSM$ $\mathcal{M}$ with infinite precision and exponential time schedule $T(k) \in \Omega(2^k)$, clocked in polynomial time. Since T is exponential and $\mathcal{M}$ is clocked in polynomial time, we conclude that the size of the oracle queries can grow at most logarithmically

[11] The following example helps to clarify the argument. Suppose that $y = 0.1100011000\ldots$ The sequence v_k can be taken as follows: $v_1 = 1$, $v_2 = 11$, $v_3 = 111$, $v_4 = 1101$, $v_5 = 11001$, $v_6 = 110010$, $v_7 = 1100100$, $v_8 = 11000111$, $v_9 = 110001100$, ...

in the size of the input. This means that for any word w with size n, there exist constants a and b such that, during the computation, $\mathcal{M}$ only queries the oracle with words of size less or equal to $\ell = a\lceil\log(n)\rceil + b$.

By Proposition 10 we can now define a prefix advice function f, encoding the boundary numbers, such that $|f(|z|)|$ is linear in size of the query $|z| \in \mathcal{O}(\ell)$, i.e., $|f(|z|)| \in \mathcal{O}(\log(n))$, where n is the size of the input.

Therefore, to decide B in polynomial time with prefix advice $f \in \log$, we can simulate $\mathcal{M}$ on the input word but whenever a transition to the query state occurs and z is written in the query tape we compare the query with the corresponding boundary numbers, i.e., with $l_{|z|}$ and $r_{|z|}$. The comparison, as explained in Section 4.6, provides the same answer as $Prot\_IP(z)$, and thus the machine uses a similar measurement algorithm to approximate the vertex, replacing the call to the physical oracle by a comparison with $f(n)$.

Since the comparison can be done in polynomial time and $\mathcal{M}$ runs in polynomial time too, we can decide B in polynomial time given the advice. □

The Theorems 2 and 6 allow us to prove the following corollary:

Corollary 1. *B is decidable by a smooth scatter machine with infinite precision and exponential protocol $T(k) \in \Omega(2^k)$, clocked in polynomial time, if and only if $B \in P/\log\star$.*

It is an open problem to know if in the above corollary holds if we remove the schedule restriction.

4.8 Probabilistic query trees

The error-prone smooth scatter machine can obtain approximations to the vertex position through the measurement algorithm. After each run of the $SmSE$, the Turing machine is in one of the three possible states: q_r, q_t or q_l. The oracle consultations of a $SmSM$ can then be seen as a ternary query tree since its (deterministic) computations are interspersed with calls to the oracle; after each call the machine is in one of the three above mentioned states.

Definition 7. *A* query tree *is a rooted tree (V, E, ν) where each node in V is a configuration of the Turing machine in the query state (the root ν is the configuration of the first call to the oracle) or in a halting state, and each edge in E is a deterministic computation of the Turing machine between consecutive oracle calls or between oracle calls and halting configurations. The only nodes with zero children are the corresponding accepting and rejecting configurations.*

Definition 8. *A* m-ary query tree *is a query tree where each node except the leaves has m children.*

Since we are not considering now the infinite precision case, we know that the behavior of the $SmSE$ is stochastic and thus, after each call to the oracle, the machine is in one of the three states with some probability. With this idea in mind, we can see all the oracle consultations by a $SmSM$ on an input w as a probabilistic ternary query tree, i.e., a ternary query tree where each edge is labeled by a probability. A single computation on w corresponds to a path in the tree, beginning in the root and ending in a leaf. The leaves of the tree are labeled with an A, if the computation on input w halts in an accepting configuration and are labeled with an R, if the computation on input w halts in a rejecting configuration (see Figure 12).

Let $T_{n,m} = (V_{n,m}, E_{n,m}, \nu_{n,m})$ denotes a n-ary probabilistic query tree with depth m.

Definition 9. *We define the set of all assignments of probabilities to the edges of* $T_{n,m}$ *as*

$$\rho(T_{n,m}) = \{\sigma : E_{n,m} \to [0,1] : \textit{the sum of } \sigma\textit{'s over the } n \textit{ outcomes of every node is } 1\} \ .$$

Denote by $T^{\sigma}_{n,m}$ the n-ary probabilistic query tree with depth m and assignment σ.

Definition 10. *The probability of one single path* π *of a* n*-ary probabilistic query tree* $T^{\sigma}_{n,m}$ *with depth* m *and assignment* σ *is*

$$\prod_{i=1}^{m} \sigma(\pi[i]) \ ,$$

where $\pi[i]$ *stands for the* i*-th edge of the path from the root. The* acceptance probability $P(T^{\sigma}_{n,m})$ *is the sum of the probabilities of all* accepting paths.

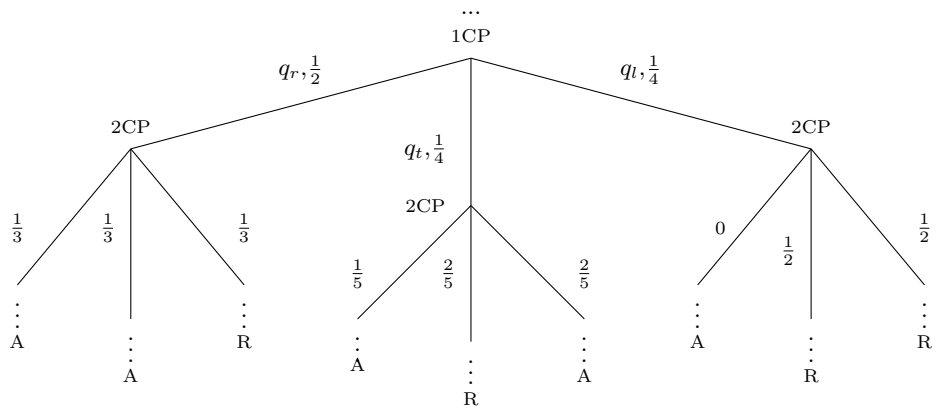


Fig. 12. The oracle calls by a smooth scatter machine as a ternary query tree, where iCP means the i-th cannon position.

We define $D(\sigma_1, \sigma_2)$, $\sigma_1, \sigma_2 \in \rho(T_{n,m})$ as the maximum distance between the two probabilistic query trees $T^{\sigma_1}_{n,m}$ and $T^{\sigma_2}_{n,m}$.

Definition 11. *For every* $\sigma_1, \sigma_2 \in \rho(T_{n,m})$*, we define*

$$D(\sigma_1, \sigma_2) = \max\{\mid \sigma_1(e) - \sigma_2(e) \mid : e \in E_{n,m}\} \ .$$

Now we define the largest possible difference in the acceptance probability for two different assignments, where the distance between the probabilities is less or equal to a value s.

Definition 12. *For any* $m \in \mathbb{N}$, $s \in [0,1]$ *and number of outcomes* $out \in \mathbb{N}$*, we define a function* $\mathcal{A}_{out} : \mathbb{N} \times [0,1] \to [0,1]$ *as*

$$\mathcal{A}_{out}(m,s) = \sup\{|P(T^{\sigma'}_{out,m}) - P(T^{\sigma}_{out,m})| : \sigma, \sigma' \in \rho(T_{out,m}) \textit{ and } D(\sigma,\sigma') \leq s\} \ .$$

This function satisfies the following relevant property:

Proposition 11. *For any $m \in \mathbb{N}$, $s \in [0,1]$, and any number $out \in \mathbb{N}$ of children in the tree, $\mathcal{A}_{out}(m,s) \leq (out-1)ms$.*

Proof: The proof follows by induction in m. The result is straightforward for $m = 0$: we take $P(T^{\sigma}_{out,0}) = P(T^{\sigma'}_{out,0}) = 0$ for each rejecting leaf, and $P(T^{\sigma}_{out,0}) = P(T^{\sigma'}_{out,0}) = 1$ for each accepting leaf. Let the statement be true for m and consider the probabilistic query tree $T_{out,m+1}$ with *out* outgoing edges, $e_1, e_2, ..., e_{out}$, and depth $m+1$. Each edge e_i is incident in a node $T_{out,m}(i)$, for $i = 1, \ldots, out$, respectively. Consider probability assignments $\sigma, \sigma' \in \rho(T_{out,m+1})$ such that $D(\sigma, \sigma') \leq s$. We have then

$$\begin{aligned} P(T^{\sigma}_{out,m+1}) &= \sigma(e_1)P(T^{\sigma}_{out,m}(1)) + \sigma(e_2)P(T^{\sigma}_{out,m}(2)) + \cdots + \sigma(e_{out})P(T^{\sigma}_{out,m}(out)) \\ P(T^{\sigma'}_{out,m+1}) &= \sigma'(e_1)P(T^{\sigma'}_{out,m}(1)) + \sigma'(e_2)P(T^{\sigma'}_{out,m}(2)) + \cdots + \sigma'(e_{out})P(T^{\sigma'}_{out,m}(out)) \ . \end{aligned}$$

As $\sigma(e_{out}) = 1-\sigma(e_1)-\sigma(e_2)-\cdots-\sigma(e_{out-1})$ and $\sigma'(e_{out}) = 1-\sigma'(e_1)-\sigma'(e_2)-\cdots-\sigma'(e_{out-1})$, we have that

$$\begin{aligned} \left|P(T^{\sigma}_{out,m+1}) - P(T^{\sigma'}_{out,m+1})\right| &= \big|(\sigma(e_1)-\sigma'(e_1))(P(T^{\sigma}_{out,m}(1)) - P(T^{\sigma}_{out,m}(out))) \\ &\quad +(\sigma(e_2)-\sigma'(e_2))(P(T^{\sigma}_{out,m}(2)) - P(T^{\sigma}_{out,m}(out))) \\ &\quad +\cdots \\ &\quad +(\sigma(e_{out-1})-\sigma'(e_{out-1}))(P(T^{\sigma}_{out,m}(out-1)) - P(T^{\sigma}_{out,m}(out-1)) \\ &\quad +\sigma'(e_1)(P(T^{\sigma}_{out,m}(1)) - P(T^{\sigma'}_{out,m}(1))) \\ &\quad +\sigma'(e_2)(P(T^{\sigma}_{out,m}(2)) - P(T^{\sigma'}_{out,m}(2))) \\ &\quad +\cdots \\ &\quad +\sigma'(e_{out})(P(T^{\sigma}_{out,m}(out)) - P(T^{\sigma'}_{out,m}(out))\big| \ . \end{aligned}$$

Since the difference of any two real numbers in $[0,1]$ lies in $[-1,1]$, we conclude that

$$\begin{aligned} \left|P(T^{\sigma}_{out,m+1}) - P(T^{\sigma'}_{out,m+1})\right| &\leq |\sigma(e_1)-\sigma'(e_1)| + |\ \sigma(e_2)-\sigma'(e_2)\ | + \cdots + |\ \sigma(e_{out-1})-\sigma'(e_{out-1})| \\ &\quad +\sigma'(e_1)\mathcal{A}_{out}(m,s) + \sigma'(e_2)\mathcal{A}_{out}(m,s) + \cdots + \sigma'(e_{out})\mathcal{A}_{out}(m,s) \\ &\leq (out-1)s + \mathcal{A}_{out}(m,s) \ . \end{aligned}$$

Therefore, using the induction hypothesis,

$$\begin{aligned} \mathcal{A}_{out}(m+1,s) &\leq \mathcal{A}_{out}(m,s) + (out-1)s \\ &\leq (out-1)ms + (out-1)s \\ &= (out-1)(m+1)s \ . \end{aligned}$$

□

4.9 Upper bound for the unbounded precision

Consider a $SmSM$ with vertex at position y and physical time t, and suppose that the $SmSM$ writes a query z with $|z| = k$, for $k \in \mathbb{N}$. Then consider the two boundary numbers l_k and r_k (see Sections 4.6 and 4.7) for the schedule $T(k)$. Since the transition to one of the states q_r, q_t and q_l is probabilistic whenever the $SmSM$ uses the $SmSE$ with the protocol $Prot\_UP$, we conclude that approximations to the probabilities involved in these possible transitions are needed in order *to simulate the oracle calls*.

Protocol $Prot\_UP(z)$ chooses uniformly some position $z' \in [z - 2^{-k}, z + 2^{-k}]$ from where to shoot, originating eight possible shooting cases represented in Figure 13. Assuming that we always have k large enough to obtain the shooting interval inside $]0, 1[$, we exclude the cases 6 and 8. Assuming that the protocol is exponential, we can discard also case 7, since the interval $]l_k, r_k[$ shrinks faster than the shooting interval. For each one of these cases, from 1 to 5, we will have the following probabilities:

1. $0 < z - 2^{-k} < l_k < r_k < z + 2^{-k} < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= \frac{l_k - z + 2^{-k}}{z + 2^{-k} - z + 2^{-k}} = \frac{1}{2} - \frac{z - l_k}{2 \times 2^{-k}} \\ P(\text{“}q_t\text{”}) &= \frac{r_k - l_k}{z + 2^{-k} - z + 2^{-k}} = \frac{r_k - l_k}{2 \times 2^{-k}} \\ P(\text{“}q_r\text{”}) &= \frac{z + 2^{-k} - r_k}{z + 2^{-k} - z + 2^{-k}} = \frac{1}{2} - \frac{r_k - z}{2 \times 2^{-k}} \ ; \end{aligned}$$

2. $0 < z - 2^{-k} < l_k < z + 2^{-k} < r_k < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= \frac{l_k - z + 2^{-k}}{z + 2^{-k} - z + 2^{-k}} = \frac{1}{2} - \frac{z - l_k}{2 \times 2^{-k}} \\ P(\text{“}q_t\text{”}) &= \frac{z + 2^{-k} - l_k}{z + 2^{-k} - z + 2^{-k}} = \frac{1}{2} - \frac{l_k - z}{2 \times 2^{-k}} \\ P(\text{“}q_r\text{”}) &= 0 \ ; \end{aligned}$$

3. $0 < l_k < z - 2^{-k} < r_k < z + 2^{-k} < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= 0 \\ P(\text{“}q_t\text{”}) &= \frac{r_k - z + 2^{-k}}{z + 2^{-k} - z + 2^{-k}} = \frac{1}{2} - \frac{z - r_k}{2 \times 2^{-k}} \\ P(\text{“}q_r\text{”}) &= \frac{z + 2^{-k} - r_k}{z + 2^{-k} - z + 2^{-k}} = \frac{1}{2} - \frac{r_k - z}{2 \times 2^{-k}} \ ; \end{aligned}$$

4. $0 < z - 2^{-k} < z + 2^{-k} < l_k$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= 1 \\ P(\text{“}q_t\text{”}) &= 0 \\ P(\text{“}q_r\text{”}) &= 0 \ ; \end{aligned}$$

5. $r_k < z - 2^{-k} < z + 2^{-k} < 1$

$$P(\text{“}q_l\text{”}) = 0$$
$$P(\text{“}q_t\text{”}) = 0$$
$$P(\text{“}q_r\text{”}) = 1\ .$$

Looking at the expressions, and considering error propagation, we can conclude that if we know $k + d$ bits of l_k and r_k we can approximate the probabilities within an error less than 2^{-d}.

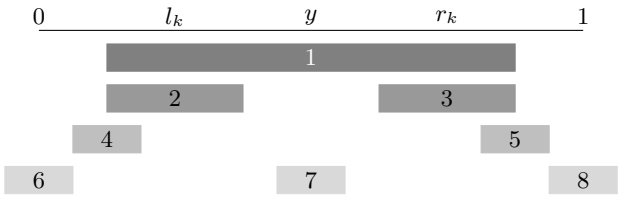


Fig. 13. Shooting cases.

Theorem 7. *If B is decidable by a smooth scatter machine with unbounded precision and exponential protocol T, clocked in polynomial time, then $B \in BPP//\log^2\star$.*

Proof: Suppose that B is decidable by a $SmSM$ $\mathcal{M}$ with unbounded precision and exponential protocol T, in polynomial time $\mathcal{O}(n^a)$. Since $\mathcal{M}$ on w runs in polynomial time in $n = |w|$ and it has an exponential time schedule $T(k)$, we can conclude that the size of the oracle queries must be at most logarithmic in the size of the input, that is, the size of the oracle queries must be less or equal to $b\lceil\log(n)\rceil + c$. The number of queries to the oracle cannot exceed the running time of the machine, so that the probabilistic query trees of the $SmSM$ have a depth of at most αn^a, for some constant α.

Let γ be the error probability of machine $\mathcal{M}$ and $d \in \mathbb{N}$ such that $2^d > 2\alpha/(1/2 - \gamma)$. The probabilities of each outcome of all oracle queries will be truncated up to the precision $2^{-d-a\lceil\log(n)\rceil}$, according with Proposition 11, in order to obtain error probability of acceptance less than $1/2 - \gamma$:

$$\begin{aligned}\mathcal{A}(\alpha n^a, 2^{-d-a\lceil\log(n)\rceil}) &\leq 2 \times \alpha n^a \times 2^{-d-a\lceil\log(n)\rceil} \\ &= \frac{2\alpha n^a}{2^d \times 2^{a\lceil\log(n)\rceil}} \\ &= \frac{2\alpha}{2^d} < (1/2 - \gamma)\ .\end{aligned}$$

Hence, as explained before the statement, to approximate the probabilities of all queries with precision $2^{-d-a\lceil\log(n)\rceil}$ we have to know $i + d + a\lceil\log(n)\rceil$ bits of l_i and r_i, for $1 \leq i \leq b\lceil\log(n)\rceil + c$. Consider now $\beta = max\{a, b\}$ and a prefix advice function f defined recursively as follows:

1. $f(0) = l_1\lfloor_{d+c}\#r_1\lfloor_{d+c}\#l_2\lfloor_{d+c}\#r_2\lfloor_{d+c}\#\cdots\#l_c\lfloor_{d+c}\#r_c\lfloor_{d+c}$;
2. $f(x+1)$ is obtained by concatenating $f(x)$ with the bits $d + c + 2\beta x + 1$ to $d + c + 2\beta x + 2\beta$ of l_i and r_i, for $1 \leq i \leq \beta x + c$; and then by adding the first $d + c + 2\beta x + 2\beta$ bits of l_i and r_i for $\beta x + c + 1 \leq i \leq \beta x + c + \beta$. (All the blocks of bits separated by #.)

Advice f encodes approximations to the boundary numbers l_i and r_i, for $1 \leq i \leq \beta x + c$: a Turing machine can read $2(\beta x + c)$ nonsequential blocks of size 2β from $f(x)$, updating at the same time the approximations of l_i and r_i for $1 \leq i \leq \beta x + c$ and 2β nonsequential blocks of size $d + c + 2\beta x + 2\beta$, to get approximations of l_i and r_i, for $\beta x + c + 1 \leq i \leq \beta x + c + \beta$. Thus, a Turing machine can have access to approximations of l_i and r_i with $d + c + 2\beta x$ bits of precision. Analysing the advice function we can conclude that:

$$|f(x)| = 2 \times (d + c + 2\beta x) \times (c + \beta x) + \sum_{i=0}^{x} 2(\beta x + c) = \mathcal{O}(x^2).$$

Since we only consider x at most logarithmic in the input size, n, we have that $|f(\lceil \log(n) \rceil)| = \mathcal{O}(\log^2(n))$. Thus, the advice function $g(n) = f(\lceil \log(n) \rceil)$ provide approximations of l_i and r_i, for $1 \leq i \leq b\lceil \log(n) \rceil + c$, with at least $i + d + a\lceil \log(n) \rceil$ bits of precision, as desired.

Therefore, to decide B in polynomial time, using the prefix advice $f \in \log^2$, we construct a Turing machine $\mathcal{M}'$ that simulates $\mathcal{M}$ on the input word but whenever $\mathcal{M}$ queries the oracle with z, $\mathcal{M}'$ compares the query z with the corresponding boundary numbers, i.e., with l_k and r_k, where $k = |z|$; checks the shooting cases; and computes the approximations to the probabilities with an error less than $2^{-d-a\lceil \log(n) \rceil}$. Then $\mathcal{M}'$ simulates a path in the probabilistic query tree, that represents the oracle consultation, by means of the computed probabilities, by tossing a coin $d + a\lceil \log(n) \rceil$ times. Note that this probabilistic tree has a depth of at most αn^a and the edge difference is less than $2^{-d-a\lceil \log(n) \rceil}$. After simulating this path, the machine $\mathcal{M}'$ proceeds as $\mathcal{M}$. Since the difference in the probability of acceptance is bounded by $1/2 - \gamma$, $\mathcal{M}'$ gives a wrong answer with probability less than $\gamma + 1/2 - \gamma = 1/2$.

Recalling that the *SmSM* $\mathcal{M}$ runs in polynomial time, that comparing query words with boundary numbers and computing probabilities can also be done in polynomial time, we conclude that B is decidable in polynomial time with advice $g(n) = f(\lceil \log(n) \rceil)$. □

4.10 Upper bound for the fixed precision

The error-prone *SmSM* with fixed precision ϵ has also probabilistic computation trees. Thus, once again, we need approximations to the boundary numbers (see Sections 4.6, 4.7 and 4.9) and to the probabilities in order to simulate the oracle whenever the *SmSM* calls the *SmSE* with the protocol *Prot_FP*.

Consider a *SmSM* with vertex at position y, fixed precision $\epsilon = 2^{-q}$, for some positive integer q, and physical time t. Suppose that the *SmSM* writes a query z with $|z| = k$ for $k \in \mathbb{N}$. Then consider the two boundary numbers l_k and r_k for the schedule $T(k)$.

Since $Prot\_FP(z)$ will choose uniformly some position $z' \in [z - \epsilon, z + \epsilon]$ from where to shoot, we have eight possible shooting cases represented also in Figure 13. As previously discussed in Section 4.9, we assume that we always have k large enough to have the shooting interval inside $]0, 1[$, excluding the cases 6 and 8. We also know that the interval $]l_k, r_k[$ shortens, so that we consider that the case 7 does not occur also.

For each one of the remaining cases, from 1 to 5, we will have the following probabilities:

1. $0 < z - \epsilon < l_k < r_k < z + \epsilon < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= \frac{l_k - z + \epsilon}{z + \epsilon - z + \epsilon} = \frac{1}{2} - \frac{z - l_k}{2\epsilon} \\ P(\text{“}q_t\text{”}) &= \frac{r_k - l_k}{z + \epsilon - z + \epsilon} = \frac{r_k - l_k}{2\epsilon} \\ P(\text{“}q_r\text{”}) &= \frac{z + \epsilon - r_k}{z + \epsilon - z + \epsilon} = \frac{1}{2} - \frac{r_k - z}{2\epsilon} \ ; \end{aligned}$$

2. $0 < z - \epsilon < l_k < z + \epsilon < r_k < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= \frac{l_k - z + \epsilon}{z + \epsilon - z + \epsilon} = \frac{1}{2} - \frac{z - l_k}{2\epsilon} \\ P(\text{“}q_t\text{”}) &= \frac{z + \epsilon - l_k}{z + \epsilon - z + \epsilon} = \frac{1}{2} - \frac{l_k - z}{2\epsilon} \\ P(\text{“}q_r\text{”}) &= 0 \ ; \end{aligned}$$

3. $0 < l_k < z - \epsilon < r_k < z + \epsilon < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= 0 \\ P(\text{“}q_t\text{”}) &= \frac{r_k - z + \epsilon}{z + \epsilon - z + \epsilon} = \frac{1}{2} - \frac{z - r_k}{2\epsilon} \\ P(\text{“}q_r\text{”}) &= \frac{z + \epsilon - r_k}{z + \epsilon - z + \epsilon} = \frac{1}{2} - \frac{r_k - z}{2\epsilon} \ ; \end{aligned}$$

4. $0 < z - \epsilon < z + \epsilon < l_k$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= 1 \\ P(\text{“}q_t\text{”}) &= 0 \\ P(\text{“}q_r\text{”}) &= 0 \ ; \end{aligned}$$

5. $r_k < z - \epsilon < z + \epsilon < 1$

$$\begin{aligned} P(\text{“}q_l\text{”}) &= 0 \\ P(\text{“}q_t\text{”}) &= 0 \\ P(\text{“}q_r\text{”}) &= 1 \ . \end{aligned}$$

Considering error propagation, we can conclude that if we know $q + d$ bits of l_k and r_k, then we can approximate these probabilities with an error at most 2^{-d}. (Note that, if we consider a real valued ϵ, we can also approximate the probabilities within the desired precision by providing the bits of ϵ too.)

We can now prove the upper bound for the fixed precision case.

Theorem 8. *If B is decidable by a smooth scatter machine with fixed precision $\epsilon = 2^{-q}$, for some positive integer q, and exponential protocol T, clocked in polynomial time, then $B \in BPP//\log^2\star$.*

Proof: Suppose that B is decidable by a *SmSM* $\mathcal{M}$, with fixed precision $\epsilon = 2^{-q}$, for some positive integer q, and exponential protocol T, in polynomial time $\mathcal{O}(n^a)$. Since $\mathcal{M}$ has an exponential time

schedule $T(k)$, we can conclude that the size of the oracle queries must be at most logarithmic in the size of the input, i.e., less then or equal to $b\lceil\log(n)\rceil + c$. Moreover, there exists a constant α such that the number of queries does not exceed αn^a and, consequently, the probabilistic query trees of the $SmSM$ will have a depth of at most αn^a.

Let γ be the error probability of machine $\mathcal{M}$ and $d \in \mathbb{N}$ such that $2^d > 2\alpha/(1/2-\gamma)$. The probabilities of each outcome of all oracle queries will be truncated up to the precision $2^{-d-a\lceil\log(n)\rceil}$, according with Proposition 11, in order to obtain error probability of acceptance less than $1/2-\gamma$:

$$\begin{aligned}\mathcal{A}(\alpha n^a, 2^{-d-a\lceil\log(n)\rceil}) &\leq 2 \times \alpha n^a \times 2^{-d-a\lceil\log(n)\rceil} \\ &= \frac{2\alpha n^a}{2^d \times 2^{a\lceil\log(n)\rceil}} \\ &= \frac{2\alpha}{2^d} < (1/2-\gamma)\ .\end{aligned}$$

Hence to approximate the probabilities of all queries with precision $2^{-d-a\lceil\log(n)\rceil}$ we have to know $q + d + a\lceil\log(n)\rceil$ bits of l_i and r_i, for $1 \leq i \leq b\lceil\log(n)\rceil + c$. Consider the prefix advice function f defined recursively as follows:

1. $f(0) = q\#l_1\downarrow_{q+d}\#r_1\downarrow_{q+d}\#l_2\downarrow_{q+d}\#r_2\downarrow_{q+d}\#\cdots\#l_c\downarrow_{q+d}\#r_c\downarrow_{q+d}$;
2. $f(x+1)$ is obtained by concatenating $f(x)$ with the bits $q+d+ax+1$ to $q+d+ax+a$ of l_i and r_i, for $1 \leq i \leq bx+c$; then by adding the first $q+d+ax+a$ bits of l_i and r_i for $bx+c+1 \leq i \leq bx+c+b$. All the blocks of bits are separated by #.

From advice $f(x)$ a Turing machine has access to the value q (and then compute ϵ), and to the approximations of the boundary numbers l_i and r_i, for $1 \leq i \leq bx+c$ by doing the following: the machine reads $2(bx+c)$ nonsequential blocks of size a from $f(x)$, updating at the same time the approximations of l_i and r_i, for $1 \leq i \leq bx+c$, and $2b$ nonsequential blocks of size $q+d+ax+a$, to get approximations of l_i and r_i, for $bx+c+1 \leq i \leq bx+c+b$. Thus, given advice $f(x)$, a Turing machine can approximate l_i and r_i with $q+d+ax$ bits of precision, for $1 \leq i \leq bx+c$.

Analyzing the advice function we can conclude that:

$$|f(x)| \leq 2 \times (q+d+ax) \times (bx+c) + \sum_{i=0}^{x} 2(bx+c) = \mathcal{O}(x^2)\ .$$

Since we only consider x at most logarithmic in the input size, n, we have that $|f(\lceil\log(n)\rceil)| = \mathcal{O}(\log^2(n))$. Thus, the advice function $g(n) = f(\lceil\log(n)\rceil)$ provide approximations of l_i and r_i, for $1 \leq i \leq b\lceil\log(n)\rceil + c$, with at least $q + d + a\lceil\log(n)\rceil$ bits of precision, as desired.

Therefore, to decide B in polynomial time on help by a prefix advice $f \in \log^2$, we construct a Turing machine $\mathcal{M}'$ that simulates $\mathcal{M}$ on the input word but whenever $\mathcal{M}$ queries the oracle with z, $\mathcal{M}'$ compares the query with the corresponding boundary numbers, i.e., with $l_{|z|}$ and $r_{|z|}$; checks the shooting cases; and computes the approximations to the probabilities with an error less than $2^{-d-a\lceil\log(n)\rceil}$. Then $\mathcal{M}'$ simulates a path in the probabilistic query tree, that represents the oracle consultation, by means of the computed probabilities, by tossing a coin $d + a\lceil\log(n)\rceil$ times. Note that this probabilistic query tree has a depth of at most αn^a and that the edge difference is less than $2^{-d-a\lceil\log(n)\rceil}$. After simulating this path the machine $\mathcal{M}'$ proceeds the computation as $\mathcal{M}$.

Turing machine $\mathcal{M}'$ gives a wrong answer with probability less than $\gamma + 1/2 - \gamma = 1/2$. Recalling that the $SmSM$ $\mathcal{M}$ runs in polynomial time, that comparing query words with boundary numbers and computing probabilities can also be done in polynomial time, we conclude that B is decidable in polynomial time with advice $g(n) = f(\lceil\log(n)\rceil)$. □

5 Upper bound with explicit time technique

As we have seen in the previous sections, the use of boundary numbers raises the question of whether we really can achieve the upper bound of $BPP//\log^2\star$. We can equally ask, under what circumstances is the upper bound actually $BPP//\log\star$? Here we consider a special case where the upper bound does reduce to $BPP//\log\star$.

Using the physical time explicitly means that, given an exact expression for the experimental time for a $SmSM$, we take good use of it to compute the boundary numbers. We assume that we have an exact expression for the experimental time $t(z) = f(z-y)$ for cannon position z As usual, the vertex position y is unknown, but we have the explicit form of the function f. Of course, this approach can cost a lot of computational resources since the the function f may be computationally difficult to compute.

In order to understand better the explicit time idea note that, in the unbounded precision case, we use the advice function to encode approximations of the boundary numbers allowing to simulate the oracle answers and to compute approximations to the probabilities. The simplest formula for an explicit time consistent with our assumptions in (1) would be, for some constant $C > 0$,

$$t(z) = \frac{C}{|y-z|} \ . \tag{3}$$

To make use of this formula, we need C to be computable, but also we need bounds on how quickly we can compute approximations to C. To further simplify matters, we shall assume that $C = 1$.

Suppose that the $SmSM$ writes a query z with $|z| = k$ for $k \in \mathbb{N}$. Then consider the two boundary numbers l_k and r_k for the schedule $T(k)$. If we consider the explicit time, as the boundary numbers satisfy the property $t(l_k) = t(r_k) = T(k)$, we have

$$l_k = y - \frac{1}{T(k)} \quad , \quad r_k = y + \frac{1}{T(k)}. \tag{4}$$

Thus, given approximations to y, we can obtain approximations of l_k and r_k. We consider that the schedule T is internal to the Turing machine, i.e. the Turing machine is capable of computing it up to any precision. Thus, by looking at the expressions and considering the error propagation rules, we conclude that if we have $d+1$ bits of precision of y and $A = 1/T(k)$, we can compute the boundary numbers with an error less than 2^{-d}.

Theorem 9. *If B is decidable by a smooth scatter machine with unbounded precision and exponential protocol T, clocked in polynomial time, then, considering explicit time in the form (3) with $C = 1$, $B \in BPP//\log\star$.*

Proof: We resume to the proof of Theorem 7, providing now the advice function that solves the problem in $BPP//\log\star$. All variables and constants are as in the proof of Theorem 7.

Consider a $SmSM$ $\mathcal{M}$ running in polynomial time and with schedule $T(k)$, exponential in k. By Theorem 7, we know that B can be decided by a probabilistic Turing machine, $\mathcal{M}'$ in polynomial time with access to an advice function f of size $\mathcal{O}(\log^2(n))$, which contains the first $d + c + 2\beta x$ bits of l_i and r_i for $1 \leq i \leq \beta x + c$.

We consider another function g, defined recursively as follows: $g(0)$ is the concatenation of the first $d+c$ bits of y; $g(x+1)$ is the concatenation of $g(x)$ with the bits $d+c+2\beta x+1$ to $d+c+2\beta x+2\beta$ of y. We can use $g(x)$ to get the first $d + c + 2\beta x$ bits of y, therefore, by the previous reasoning,

we can use the approximations of y in order to compute the approximation of all l_k and r_k, for $1 \leq k \leq bx+c$, with $d+c+2\beta x$ bits of precision as the Turing machine can compute in polynomial time A with $d+c+2\beta x$ bits of precision.

Analyzing $g(x)$ we conclude that $|g(x)| = (d+c+2\beta x)+(x+1) = \mathcal{O}(x)$. As in our case as x will be at most logarithmic in the size of the input we conclude that $|g(\lceil \log(n) \rceil)| = \mathcal{O}(\lceil \log(n) \rceil)$.

Thus, we define a probabilistic Turing machine $\mathcal{M}''$ that on input w simulates $\mathcal{M}'$ on w but instead of using the advice $f(|w|)$, uses the advice $g(|w|)$. Since we can recover the information of f from g in polynomial time and $\mathcal{M}'$ runs in polynomial time too, we conclude that our Turing machine runs in polynomial time and decides B. □

Using Theorems 3 and 9 we can trivially state the following corollary.

Corollary 2. *Considering explicit time in the form (3) with $C = 1$, B is decidable by a smooth scatter machine with unbounded precision and exponential protocol T, clocked in polynomial time, if and only if $B \in BPP//\log\star$.*

Theorem 10. *If B is decidable by a smooth scatter machine with fixed precision $\epsilon = 2^{-q}$, for some positive integer q, and exponential protocol T, clocked in polynomial time, then, considering explicit time in the form (3) with $C = 1$, $B \in BPP//\log\star$.*

Proof: We resume to the proof of Theorem 8, providing now the advice function that solves the problem in $BPP//\log\star$. All variables and constants are as in the proof of Theorem 8.

Consider a $SmSM$ $\mathcal{M}$ running in polynomial time, with fixed precision $\epsilon = 2^{-q}$, for some positive integer q, and a schedule $T(k)$, exponential in k. By Theorem 8, we know that B can be decided by a probabilistic Turing machine $\mathcal{M}'$ in polynomial time with access to an advice function f of size $\mathcal{O}(\log^2(n))$, which contains the first $q+d+ax$ bits of l_i and r_i for $1 \leq i \leq bx+c$ and the the value q.

We consider another function g, defined recursively as follows: $g(0)$ is the concatenation of q with the first $q+d$ bits of y; $g(x+1)$ is the concatenation of $g(x)$ with the bits $q+d+1+ax$ to $q+d+1+ax+a$ of y. We can use $g(x)$ to get ϵ and to get the first $q+d+ax$ bits of y. Therefore, by the previous reasoning, we can use the approximations of y to compute the approximation of all l_k and r_k for $1 \leq k \leq bx+c$ with $q+d+ax$ bits of precision, as the Turing machine can compute in polynomial time A with $q+d+ax$ bits of precision.

Analyzing $g(x)$ we conclude that $|g(x)| = (q+d+ax)+(x+2) = \mathcal{O}(x)$. As in our case as x will be at most logarithmic in the size of the input we conclude that $|g(\lceil \log(n) \rceil)| = \mathcal{O}(\log(n))$.

Thus, we define a probabilistic Turing machine $\mathcal{M}''$ that on input w simulates $\mathcal{M}'$ on w but instead of using the advice $f(|w|)$, uses the advice $g(|w|)$. Since we can recover the information of f from g in polynomial time and $\mathcal{M}'$ runs in polynomial time too we conclude that our Turing machine runs in polynomial time and decides B. □

Using Theorems 4 and 10 we can trivially state the following corollary:

Corollary 3. *Considering explicit time in the form (3) with $C = 1$, B is decidable by a smooth scatter machine with fixed precision $\epsilon = 2^{-q}$, for some positive integer q, and exponential protocol T, clocked in polynomial time, if and only if $B \in BPP//\log\star$.*

6 Conclusion

In the past two decades there was a growing of interest in non-conventional models of computation inspired by the natural processes in biology, physics and chemistry. Some of these models explore parallel processing, some others see advantage in analogue components translated into real numbers, appearing as parameters in the systems. In this paper we explored an abstraction of the last category of models, studying an analogue-digital (hybrid) model of computation where the Turing machine is coupled with a physical oracle.

Abstracting from other models, the model we propose is introduced as a laboratory, where the computational power can be studied depending on the protocol between the Turing machine and the analogue component — being it infinite precision, unbounded precision and fixed precision, and still open to other forms of communication. The physical oracle itself is a measurement that the Turing machine performs in the physical world as an abstract scientist. The communication between the Turing machine and the physical oracle is made through a query tape where the parameter needed to initialize the experiment is written. The consultation of the analogue oracle has a cost that is not just one time step of computation as a consequence of the unavoidable time costs inherent to the physical process.

We considered a particular physical oracle, the smooth scatter experiment or $SmSE$. This experiment is a symmetric two-sided measurement of distance and it is governed by elementary Newtonian mechanics. The $SmSE$ belongs to the class of physical oracles with exponential physical time t (the intrinsic time of the experiment) characterized by the following axioms:

1. Real values — The experiment is designed to find a physical unknown parameter $y \in]0, 1[$;
2. Queries — Each query is a binary string $z_1 z_2 \cdots z_k$ denoting a dyadic rational $z = 0.z_1 z_2 \cdots z_k$;
3. Finite output — The outcome is either $y < z$, $y > z$, with possible mistakes, or timeout;
4. Protocol timer — There is a time schedule $T : \mathbb{N} \to \mathbb{N}$, so that the time given to any query of length k is bounded by $T(k)$;
5. Sufficiency of the protocol — If $|y - z| > 2^{-|z|}$, then the result is not timeout;
6. Repeatability — Identical queries will result in identical results including identical timeouts.

This particular set of oracles led to a conjecture, already discussed in [20] and [5], stating that for all "reasonable" physical theories and for all measurements based on them, the physical time of the experiment is at least exponential, i.e., the time needed to access the n-th bit of the parameter being measured is at least exponential in n.

6.1 The computational power of the analogue-digital machine

Using different protocols we get different ways of communication between the Turing machine and the analogue device — the $SmSE$ in the present case (see Section 3). Different protocols relate with different measurement algorithms (see Section 3.2), defining three different types of smooth scatter machine or $SmSM$.

Codifying in the vertex of a $SmSE$ enough information, we were able to use the oracle to both generate sequences of non-biased coin tosses and, performing a measurement, solving decision problems of a suitable nonuniform complexity class. To measure the position of the vertex, we considered a bound for the consultation time — a time schedule —, exponential in the precision (number of bits) of the query. Note that, although the lower bounds were proved for a specific analogue-digital machine — the $SmSM$ —, they are the same for every oracle we studied (see [7]).

	Infinite	**Unbounded**	**Fixed**
Lower Bound	$P/\log\star$	$BPP//\log\star$	$BPP//\log\star$
Upper Bound	$P/\log\star$	$BPP//\log^2\star$ Exponential schedule	$BPP//\log^2\star$ Exponential schedule
Upper Bound Explicit Time	—	$BPP//\log\star$ Exponential schedule	$BPP//\log\star$ Exponential schedule

Table 1. Main non-uniform complexity classes relative to the different protocols of the analogue-digital machine clocked in polynomial time.

Then we constructed advice functions encoding enough information to simulate $SmSE$ queries and established upper bounds, namely we proved that logarithmic squared size advice suffice to encode approximations to the so-called boundary numbers (see Section 4.6) as in [10]. Afterwards, again as in [10], we used an explicit time technique to reduce logarithmic squared advice just to logarithmic advice. Since boundary numbers exist, at least for the two-sided oracles with exponential physical time, we also conclude that the upper bounds are common to all two-sided physical oracles with such physical times.

Our statements on the computational power of the analogue-digital machine clocked in polynomial time are summarized in the table of Figure 1.

6.2 Open problems

We have two non-trivial open problems related to oracles with exponential consultation time: (a) in the infinite precision case, to know if the lower and the upper bounds can be made to coincide without assumptions on the time schedule and (b) in the error-prone cases, to know if the lower and the upper bounds can be made to coincide without using the explicit time technique, namely, it is not known if there exists a set not belonging $BPP//\log\star$, decidable by a smooth scatter machine (or any other equivalent two-sided machine) in polynomial time.

Acknowledgements. To Bill Tantau for the use of pgf/TikZ applications. We thank Andrew Adamatzky for inviting this paper to the commemorative volume of the 10th Anniversary of the International Journal of Unconventional Computing.

References

1. Hava T. Siegelmann and Eduardo D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331–360, 1994.
2. Damien Woods and Thomas J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1-3):227–258, 2005.
3. Olivier Bournez and Michel Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2):417–459, 1996.

4. Rudolf Carnap. *Philosophical Foundations of Physics*. Basic Books, 1966.
5. Edwin Beggs, José Félix Costa, and John V. Tucker. Computational Models of Measurement and Hempel's Axiomatization. In Arturo Carsetti, editor, *Causality, Meaningful Complexity and Knowledge Construction*, volume 46 of *Theory and Decision Library A*, pages 155–184. Springer, 2010.
6. Robert Geroch and James B. Hartle. Computability and physical theories. *Foundations of Physics*, 16(6):533–550, 1986.
7. Edwin Beggs, José Félix Costa, and John V. Tucker. Three forms of physical measurement and their computability. *The Review of Symbolic Logic*, 7(4):618–646, 2014.
8. Carl G. Hempel. *Fundamentals of Concept Formation in Empirical Science*. International Encyclopedia of Unified Science II, 7, 1952.
9. David H. Krantz, Patrick Suppes, R. Duncan Luce, and Amos Tversky. *Foundations of Measurement*. Dover, 2009.
10. 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, September 2013.
11. Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. On the power of vanishing value measurements. *Submitted*, 2015.
12. Edwin J. Beggs, José Félix Costa, and J. V. Tucker. Limits to measurement in experiments governed by algorithms. *Mathematical Structures in Computer Science*, 20(06):1019–1050, December 2010.
13. 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.
14. 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, October 2008.
15. 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, May 2009.
16. Edwin J. Beggs, José Félix Costa, and John V. Tucker. Axiomatizing physical experiments as oracles to algorithms. *Philosophical Transactions of the Royal Society, Series A(Mathematical, Physical and Engineering Sciences)*, 370(12):3359–3384, June 2012.
17. José L. Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*, volume 11 of *Theoretical Computer Science*. Springer, 1990.
18. José L. Balcázar and Montserrat Hermo. The structure of logarithmic advice complexity classes. *Theoretical Computer Science*, 207(1):217–244, October 1998.
19. Hava T. Siegelmann. *Neural networks and analog computation : beyond the Turing limit*. Birkhäuser, 1999.
20. Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. An Analogue-Digital Church-Turing Thesis. *International Journal of Foundations of Computer Science*, 25(4):373-389, June 2014.

Appendix A: Nonuniform complexity classes

A nonuniform complexity class is a way of characterising families $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ of finite machines, such like logic circuits, where the element $\mathcal{C}_n$ decides a restriction of some problem to inputs of size n. Nonuniformity arises because for $n \neq m$, $\mathcal{C}_n$ may be unrelated to $\mathcal{C}_m$; eventually, there is a distinct algorithm for each input size (see [17]). The elements of a nonuniform class can be unified by means of a (possibly noncomputable) advice function, as introduced in Section 4, making up just one algorithm for all input sizes. The values of such a function provided with the inputs add the extra information needed to perform the computations (see [18]).

The nonuniform complexity classes have an important role in the *Complexity Theory*. The class P/poly contains the undecidable halting set $\{0^n\colon n$ is the encoding of a Turing machine that halts on input $0\}$, and it corresponds to the set of families decidable by a polynomial size circuit. The class $\mathrm{P}/\log$ also contains the halting set defined as $\{0^{2^n}\colon n$ is the encoding of a Turing machine that halts on input $0\}$.

The definition of nonuniform complexity classes was given in Section 4. Generally we consider four cases: $\mathbb{C}/\mathbb{F}$, $\mathbb{C}/\mathbb{F}\star$, $\mathbb{C}//\mathbb{F}$, and $\mathbb{C}//\mathbb{F}\star$. The second and the fourth cases are small variations of the first and third, respectively. To understand the difference, note that $\mathbb{C}/\mathbb{F}$ is the class of sets B for which there exists a set $A \in \mathbb{C}$ and an advice function $f \in \mathbb{F}$ such that, for every $w \in \{0,1\}^\star$, $w \in B$ iff $\langle w, f(|w|)\rangle \in A$. In this case, the advice function is fixed after choosing the Turing machine that decides the set A. As it is more intuitive to fix the Turing machine after choosing the suitable advice function, we considered a less restrictive definition, the type $\mathbb{C}//\mathbb{F}$: the class of sets B for which, given an advice function $f \in \mathbb{F}$, there exists a set $A \in \mathbb{C}$ such that, for every $w \in \{0,1\}^\star$, $w \in B$ iff $\langle w, f(|w|)\rangle \in A$.

The following structural relations hold between the nonuniform classes used throughout this paper:

$$P/\log\star \subseteq BPP/\log\star \subseteq BPP//\log\star\ .$$

This result is trivial since we can just use the same Turing machine and the same advice function.

Appendix B: The Cantor set

We prove Proposition 6 (see [10] and [12] for further details). This proposition allows us to frame the distance between a dyadic rational and a real number. Recall that a dyadic rational is a number of the form $n/2^k$, where n is an integer and k is a positive integer. If such a number belongs to $\mathcal{C}_3$ then it is composed by triplets of the form 001, 010 or 100.

Proposition 12. *For every $x \in \mathcal{C}_3$ and for every dyadic rational $z \in]0,1[$ with size $|z|= m$, if $|x - z| \leq 1/2^{i+5}$, then the binary expansion of x and z coincide in the first i bits and $|x - z| > 1/2^{-(m+10)}$.*

Proof: Suppose that x and z coincide in the first $i-1$ bits and differ in the i-th bit. We have two possible cases:

$z < x$: In this case $z_i = 0$ and $x_i = 1$ and the worst case for the difference occur when the binary expansion for z after i-th position begins with a sequence of 1s and the binary expansion for x after i-th position begins with a sequence of 0s.

	i	lower bound of $\lvert x - z\rvert$
z	$\cdots 011111 \cdots$	
$x(\text{case } i \equiv_3 0)$	$\cdots 100100 \cdots$	$> 2^{-(i+3)}$
$x(\text{case } i \equiv_3 1)$	$\cdots 100001 \cdots$	$> 2^{-(i+5)}$
$x(\text{case } i \equiv_3 2)$	$\cdots 100010 \cdots$	$> 2^{-(i+4)}$

$z > x$: In this case $z_i = 1$ and $x_i = 0$ and the worst case for the difference occur when the binary expansion for z after i-th position begins with a sequence of 0s and the binary expansion for x after i-th position begins with a sequence of 1s:

	i	lower bound of $\lvert x - z\rvert$
z	$\cdots 1000 \cdots$	
$x(\text{case } i \equiv_3 0)$	$\cdots 0100 \cdots$	$> 2^{-(i+2)}$
$x(\text{case } i \equiv_3 1)$	$\cdots 0101 \cdots$	$> 2^{-(i+2)}$
$x(\text{case } i \equiv_3 2)$	$\cdots 0110 \cdots$	$> 2^{-(i+3)}$

We can conclude that in any case $|x - z| > 2^{-(i+5)}$. Thus, if $|x - z| \leq 2^{-(i+5)}$, then x an z coincide in the first i bits.

The binary expansion of z after some position m is exclusively composed by $0s$ and since $x \in \mathcal{C}_3$, it has at most 4 consecutive 0s after the m-th bit. Thus, supposing that x and z coincide up to m-th position, after this position they can coincide at most in the next 4 positions so they cannot coincide in $m + 5$ bits. Therefore, by the first part of the statement, $|x - z| > 2^{-(m+10)}$. □

Appendix C: Random sequences

Propositions 7 and 8 show how the $SmSE$ with unbounded or fixed precision could be seen as a biased coin. Given a biased coin, as stated by Proposition 9, we can simulate a fair sequence of coin tosses. Herein, we present the proof of such a statement.

Proposition 13. *Given a biased coin with probability of heads $\delta \in]0, 1[$ and a constant $\gamma \in]0, 1[$, we can simulate, up to probability $\geq \gamma$, a sequence of independent fair coin tosses of length n by performing a linear number of biased coin tosses.*

Proof: Consider that we have a biased coin with probability of heads $\delta \in]0, 1[$. To simulate a fair coin toss we perform the following algorithm: Toss the biased coin twice and,

1. If the output is HT then output H;
2. If the output is TH then output T;
3. If the output is HH or TT then repeat algorithm.

As the probability of HT is equal to TH, we have the same probability of getting a H and a T and thus we simulate a fair coin. The probability that the algorithm halts in one run is $r = 2q(1-q)$ and the probability of run it again is $s = 1-\delta$. We want to run the algorithm until we get a sequence of fair coin tosses with size n. To get this sequence we may need to run the algorithm more than n times and thus we will study the total number of coin tosses required by considering the variable T_n denoting the number of runs until we get n fair coin tosses. The value T_n is a random variable that is given by the negative binomial distribution

$$T_n \stackrel{d}{=} NB(n, s) \ .$$

In this case we have the following mean and variance:

$$\mu = \frac{ns}{r} + n = \frac{n}{r}, \qquad \upsilon = \frac{ns}{r^2} \ .$$

Now, using the Chebyshev's inequality, we get

$$P(\mid T_n - \mu \mid \geq t) \leq \frac{\upsilon}{t^2} \ .$$

And thus, by considering $t = \alpha n$, for some α, we get

$$P(T_n \geq \mu + \alpha n) \leq \frac{ns}{r^2(\alpha n)^2} < \frac{1}{r^2\alpha^2 n} \ .$$

Since the worst case is for $n = 1$, in order to get the probability of failure less than $1 - \gamma$ we need

$$\alpha \geq \frac{1}{r\sqrt{(1-\gamma)}} \ .$$

Noticing that $T_n \geq \mu + \alpha n$, we find that the total number of runs is

$$\frac{n}{r} + \frac{1}{r\sqrt{(1-\gamma)}} \times n = \frac{n}{r}\left(1 + \frac{1}{\sqrt{1-\gamma}}\right) \ .$$

Since we toss a coin two times in each run, we get that the total number of coin tosses is linear in n

$$\frac{n}{\delta(1-\delta)}\left(1 + \frac{1}{\sqrt{1-\gamma}}\right) \ .$$

□