TÉCNICO LISBOA

Measuring quantities with analogue-digital systems

Eduardo de Arbués Moreira Castro Skapinakis

Thesis to obtain the Master of Science Degree in

Mathematics and Applications

Supervisor: Prof. José Félix Costa

Examination Committee

Chairperson: Prof. Paulo Mateus
Supervisor: Prof. José Félix Costa
Member of the Committee: Prof. Isabel Oitavem Rocha

March 29, 2022

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I thank my advisor, José Félix Costa, for both the guidance and availability, during the elaboration of this thesis, and for introducing me to this subject with such care and patience.

I would also like to thank my family for their support and Luís and Beatriz for many helpful conversations.

Abstract

We consider a model of computation in which a physical device, capable of performing measurements, is coupled with a Turing machine, functioning as an oracle. This interaction, between the machine and the experiment, is mediated by a protocol, that specifies the experimental precision, and a schedule, that clocks the number of machine steps during a call to the physical oracle.

We start by studying the class of sets that can be decided using this hybrid model of computation, in polynomial time, when the schedule is an exponential function and considering three types of experimental precision: infinite, unbounded and finite.

We then introduce Hempel's theory of measurement, which captures the intuitive notion of a measurement procedure, and present a new axiomatization, which, including the concept of the duration of an experiment, recovers Hempel's theory as we allow time to approach infinity. We look at three forms of physical measurement and prove that, in each case, an experimental apparatus and a process can be devised to satisfy this axiomatization.

Finally, we study the physical parameters (regarded as real numbers) that can be measured with a given schedule – the measurable numbers. We consider the case where we allow the schedule to be an arbitrary (time constructible) function and the case where the complexity of the schedule is to be fixed a priori. In this last case, we characterize the real numbers that can be measured with two types of exponential schedules and with a primitive recursive schedule.

Keywords

Digital-analogue computation. Physical oracle. Fundamental measurement. Forms of physical measurement. Measurable numbers. Measurement complexity of a real number.

Resumo

Consideramos um modelo de computação no qual uma experiência física, que permite realizar medições, é acoplada a uma máquina de Turing, funcionando como um oráculo. Esta interacção, entre a máquina e a experiência, é mediada por um protocolo, que determina a precisão experimental, e um relógio, que cronometra as transições da máquina durante uma chamada ao oráculo físico.

Começamos por estudar a classe de conjuntos que podem ser decididos por este modelo híbrido, em tempo polinomial, quando o relógio é exponencial e considerando três tipos de precisão experimental: infinita, ilimitada e finita.

De seguida, introduzimos a teoria da medição de Hempel, que captura a noção intuitiva de um procedimento de medição, e apresentamos uma nova axiomatização que, incluindo na medição o conceito da duração de uma experiência, recupera a teoria do Hempel quando permitimos que o tempo se aproxima do infinito. Observamos três formas de medição física e provamos que, em qualquer um dos casos, é possível criar um aparato e um procedimento que satisfaçam esta axiomatização.

Finalmente, estudamos os parâmetros físicos (vistos como números reais) que podem ser medidos com um dado relógio – os números mensuráveis. Consideramos o caso em que o relógio pode ser uma função arbitrária (construtível no tempo) e o caso em que a complexidade do relógio é fixa à priori. Neste último caso, caracterizamos os números reais que podem ser medidos com dois tipos de relógios exponenciais e com um relógio primitivo recursivo.

Palavras Chave

Computação digital-analógica. Oráculo físico. Medição fundamental. Formas de medição física. Números mensuráveis. Complexidade de medição de um número real.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

A	Constant used in the lower bound for the physical duration of an experiment
a_k	The number of bits until the end of the block u_k, i.e., $a_k = u_1 + \cdots + u_k$
$\mathcal{A}_m(n, s)$	The largest possible difference between the acceptance probability of two m-ary trees with depth n, whose maximum distance is bounded above by s
BB	The Busy Beaver function
BBE	Broken Balance Experiment
C	Constant used in the upper bound for the physical duration of an experiment
$\mathcal{C}_3$	The set of Cantor numbers
CME	Collider Machine Experiment
$D(\sigma_1, \sigma_2)$	The maximum distance between two probabilistic query trees $T^{\sigma_1}_{m,n}$ and $T^{\sigma_2}_{m,n}$
ε	A fixed precision equal to 2^{-q}, for some positive integer q. It can also be used to represent the special word in $\Sigma^\star$ which contains no letters, i.e., the empty word
E	The set of edges in a query tree
e_i	The edge connecting a node to its ith child
$E_{m,n}$	The set of edges of $T_{m,n}$
$\mathcal{E}^n$	The nth layer of the Grzegorczyk hierarchy
$\{l_i\}_{i=1}^n$	The concatenation of the words l_1 to l_n.
$l_1 \cdot l_2$	The concatenation of the words l_1 and l_2. It might also be denoted by $l_1 l_2$
l_k	A left boundary number, i.e., a boundary number such that $l_k < y$
$L_{m,n}$	The set of inner nodes of $T_{m,n}$

$l\rfloor_n^m$	Digits n to m of l
$\mathcal{O}$	A class of physical objects, endowed with some attribute, such as mass or length
π	A path in the query tree
$\pi[i]$	The i-th edge belonging to the path π
PR	The class of primitive recursive functions
Prot_FP(z)	Communication protocol for fixed precision with query z
Prot_IP(z)	Communication protocol for infinite precision with query z
Prot_UP(z)	Communication protocol for unbounded precision with query z
$P(T_{m,n}^{\sigma})$	The acceptance probability of $T_{m,n}^{\sigma}$
q_l	State that the Turing machine is in, after the particle is detected in the left collecting box
q_r	State that the Turing machine is in, after the particle is detected in the right collecting box
q_t	State that the Turing machine is in, after a time out has occurred
r_k	A left boundary number, i.e., a boundary number such that $y < r_k$
$\rho(T_{m,n})$	The set of all assignments for the m-ary probabilistic query tree with height equal to n
Σ	An alphabet
$\Sigma^{\star}$	The set of all words over the alphabet Σ
SmSE	Smooth Scatter Experiment
SmSM	Smooth Scatter Machine
T	Time schedule
TM	Turing Machine
$T_{m,n}$	The m-ary probabilistic query tree with height equal to n
$T_{m,n}^{\sigma}$	A query tree $T_{m,n}$ with assignment σ
TSmSE	Two wedge Smooth Scatter Experiment
$t(z)$	Physical time of an experiment performed with a query z
u_i	The ith child of a node u of a query tree

u_k	The number of digits in the k-th group of the binary expansion of a number in $(0, 1)$, where $k \in \mathbb{N}$, $u_1 \geq 0$ and $u_i \geq 1$ for $i \geq 2$
V	The set of nodes in a query tree
VBE	Vanishing Balance Experiment
$V_{m,n}$	The set of nodes of $T_{m,n}$
y	Vertex position that lies in $(0, 1)$
z	Shooting position of the cannon of the SmSM. It represents both a binary word and the corresponding dyadic rational
$\|z\|$	Size of the word z
z_i	The ith bit of z
$z\downarrow_l$	The pruning or the padding of the word z, until it has l bits

Glossary

Acceptance probability	The sum of the probabilities of each accepting path in the probabilistic query tree.
Accepting path	A path in the probabilistic query tree that defines the accepted words.
Accepting state	A state of the Turing machine that defines the accepted words.
Advice function	A total function $f : \mathbb{N} \to \Sigma^*$.
Arity	Number of arguments taken by a function or operation.
BCT conjecture	For every “reasonable" physical theory, which supports measurement experiments, the time intrinsic to a measurement is at least exponential.
Boundary numbers	The real numbers in $(0, 1)$, l_k and r_k, such that $t(l_k) = t(r_k) = T(k)$ and $l_k < y < r_k$.
Busy Beaver	The total function $BB : \mathbb{N} \to \mathbb{N}$, defined by: $BB(0) = 0$ and $BB(n)$ is the maximum output for input 0 among all Turing machines with n states that halt on input 0.
Cantor numbers	Set of real numbers x of the form $x = \sum_{k=1}^{\infty} x_k 2^{-3k}$, for $x_k \in \{1, 2, 4\}$.
Characteristic function	Given a set $A \subseteq \Sigma$, the characteristic function of A is the one which, given an input n, returns 1, if $n \in A$, and 0, otherwise.
Chebyshev's inequality	If X is a random variable, with finite expected value μ and finite non-zero variance σ^2, then, for any real number t, $P(\|X - \mu\| \geq t\sigma) \leq 1/t^2$.
C^n	Given a function $f : D \subset \mathbb{R} \to \mathbb{R}$ and a number $n \in \mathbb{N}$, $f \in C^n$ if f and all of its derivatives $f^{(1)}, \ldots, f^{(n)}$ are continuous.
Comparative concept	Given two binary relation $\mathcal{L}$ and $\mathcal{E}$ over $\mathcal{O}$, we say that $\mathcal{L}$ and $\mathcal{E}$ are a comparative concept if $\mathcal{E}$ is an equivalence relation and $\mathcal{L}$ is transitive, $\mathcal{E}$-irreflexive, and $\mathcal{E}$-connected.
Computable	We say that a function f is computable if there exists a Turing machine that, on input x, outputs $f(x)$ on the output tape.

Computable number	A number y for which there exist a Turing machine capable of printing approximations of y given any precision.
Decidable	We say that a set A is decidable if there exists a Turing machine that accepts all the words in A and rejects all the words not in A.
Deterministic Turing machine	A Turing machine for which the transition function is deterministic, i.e., each rule consists in only one possible action.
DTIME	For $T : \mathbb{N} \to \mathbb{N}$ a total function, we define DTIME(T) as the class of sets that can be decided by a deterministic Turing machine, whose time is bounded by T.
Dyadic rational	A number of the form $n/2^k$ where n is an integer and k is a natural number. Every dyadic rational has a finite binary expansion.
$\mathcal{E}$-connected	Given two binary relation $\mathcal{L}$ and $\mathcal{E}$ over $\mathcal{O}$, we say that $\mathcal{L}$ is $\mathcal{E}$-connected if, for every $a, b \in \mathcal{O}$, if $a\mathcal{E}b$ does not hold, then $a\mathcal{L}b$ or $b\mathcal{L}a$ holds.
$\mathcal{E}$-irreflexive	Given two binary relation $\mathcal{L}$ and $\mathcal{E}$ over $\mathcal{O}$, we say that $\mathcal{L}$ is $\mathcal{E}$-irreflexive if, for every $a, b \in \mathcal{O}$, if $a\mathcal{E}b$ holds, then $a\mathcal{L}b$ does not hold.
Efron's dice	A set of four six-sided dice, $A_0, \ldots A_3$, such that, for each $i = 0, \ldots, 3$, die A_i is twice as likely to beat die $A_{i+1 \bmod 4}$.
Equivalence relation	A binary relation $\mathcal{E}$ over $\mathcal{O}$ is an equivalence relation if it is reflexive, symmetric and transitive. For arbitrary $a, b \in \mathcal{O}$, we use the notation $a\mathcal{E}b$ to mean that a is in relation with b.
Linear search algorithm	A root-finding method that repeatedly bisects an interval and then selects the sub-interval in which a root must lie for further processing.
m-ary query tree	A query tree where each internal node has exactly m children.
Mean value theorem	If a function f is continuous on the closed interval $[a, b]$ and differentiable on the open interval (a, b), then there exists at least one point $c \in (a, b)$ such that $f(b) - f(a) = (b - a)f'(c)$.
Measurable number	A number y for which there exist a Turing machine coupled with a physical oracle with attribute y, running a measurement algorithm, capable of printing approximations of y given any precision.
Negative binomial distribution	The distribution of the number of trials until we have k successes, where each trial has a probability of success p.

Non-deterministic Turing machine	A Turing machine for which the transition function is non-deterministic, i.e., each rule consists in more then one possible action.
Non-uniform complexity class	A class that abstracts the infinite set of families of finite machines, $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{C}_n$ decides a restriction of the problem for inputs of size n. Nonuniformity arises from the fact that $\mathcal{C}_n$ and $\mathcal{C}_m$ are in general unrelated for every n and m such that $n \neq m$.
$O(f)$	For a function f, we define $O(f)$ as the class of functions g, for which there are positive constants C and k, such that $g(n) \leq Cf(n)$, for all $n \geq k$.
One-sided type experiment	An experiment that approximates the unknown value just from one side, i.e., it approximates the unknown value, y, either with values from below or with values from above x, checking if $y < x$ or if $x < y$.
Oracle	A set used by the Turing machine, that can be consulted in one time step, querying about set membership of a word.
Oracle Turing machine	A Turing machine with an extra tape, the query tape, that allows it to communicate with an external device, the oracle.
Pairing function	Function that encodes two words in a single word over the same alphabet.
Physical oracle	A set of physical parameters that encodes information.
Physical time	The time needed to conclude the experiment.
Prefix advice function	An advice function f such that, for every n, $f(n)$ is a prefix of $f(n+1)$.
Probabilistic query tree	A query tree were each branch has a probability associated with it.
Probabilistic Turing machine	A Turing machine for which the transition function is probabilistic, i.e., each rule consists in one or more possible actions with a probability associated with them.
Probability assignment	For $n, m \in \mathbb{N}$ a probability assignment is a total function $\sigma : E_{m,n} \to [0, 1]$, such that the sum of the function σ for the m outcomes of each node is 1.
Query state	A special state, defined for oracle Turing machines, which the machines enters to perform an oracle consultation.
Query Tape	A special tape, defined for oracle Turing machines, where the machine can write a word to be sent to the oracle.
Rejecting state	A state of the Turing machine that defines the rejected words.

Running time	Working time of a Turing machine.
Taylor theorem	We consider Taylor's theorem with the mean-value forms of the remainder. If f is a function, n times differentiable near $x = a$, then $f(x) = \sum_{k=0}^{n-1} f^{(k)}(a)(x - a)^k/k! + h_n(x)$, where, for some ξ between a and x, $h_n(x) = f^{(n)}(\xi)(x - a)^n/n!$.
Time constructible	A function $f : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if there exists a deterministic Turing machine M and a number $p \in \mathbb{N}$ such that, for any input word of size $n \geq p$, M halts after exactly $f(n)$ transitions.
Time schedule	A time constructible function, in the size of the query, that establishes the time that the machine must wait for an answer from the physical oracle. We say "wait constructive $T(n)$ units of time" to mean "wait for as long as it takes for T to run on input n".
Turing machine	Abstract computing device, introduced by Alan Turing, that manipulates symbols on tapes according to a set of rules, the transition function.
Two-sided type experiment	An experiment that approximates the unknown value from two sides, i.e., it approximates the unknown value y with values from above or with values from below x, checking if $y < x$ and $x < y$.
Universal measuring procedure	An algorithm for a SmSM, such that, for every measurable number, y, there exists a time constructible schedule T, such that the machine with T measures y.
Vanishing type experiment	An experiment that approximates the unknown value y from the physical time taken by the experiment.
$\Omega(f)$	For a function f, we define $\Omega(f)$ as the class of functions g, for which there are positive constants C and k, such that $g(n) \geq Cf(n)$, for all $n \geq k$.

1

Introduction

1.1 Introduction

“The study of (...) computable (...) functions stands at the intersection of three fields: mathematics, theoretical computer science, and philosophy" (see [37]).

Computation is an interesting and vast field, with applications ranging from Philosophy, namely in the areas of proof theory and cognition (see [85] and [80]), to Mathematics and Theoretical Computer Science. This is not surprising, if one reflects on how profound the concept of an “algorithm" is: what exactly *is* an algorithm? How can we thoroughly distinguish an algorithmic procedure from a non-algorithmic one? And does non-algorithmic imply not implementable? These questions, which, in Mathematics, are addressed in the field of computability theory, regard the *concept* of an algorithm. On the other hand, the classification of algorithms according to their *inherent complexity*, which is studied in the branch of complexity theory, is a field which led to the creation of a vast “Zoo" of complexity classes (see [1]) and produced one of the Millennium Problems: the *P versus NP* problem, introduced by Cook in 1971 (see [30]).

Even though algorithms have been around for millennia[1], it wasn't until the 1920's that mathematicians started dealing with the question of whether or not certain problems *have an algorithmic solution*. The issue was raised in 1928, when Hilbert and Ackermann presented the mathematical community with the *Entscheidungsproblem*, which had appeared in lectures by Hilbert in the 1920's, and asked if there is an algorithm for deciding if a given sentence is logically valid or not (see [74] and [47]). In 1936 Church, Turing[2] and Post each published papers characterizing the class of computable functions, thus reducing a proof of unsolvability to one of a belonging to a class. These definitions were proven to be equivalent (see [81] and [76]), but it was Turing's solution that convinced the founders of the subject, Gödel, Church and Kleene, as the correct definition of computability.

In his paper, which he later corrected in [82], Turing gave the first machine-based model of what it means for a function to be computable: the Turing machine model. The paper also included a description of what we now call a universal Turing machine: one with the ability to simulate any other (see [31]). Now, even though no physical device can emulate a *true* universal Turing machine, namely since these are taken to have an infinite or unlimited memory tape, the existence of such a machine lead to the goal of building a device that could usefully *approximate universality*, a task which was first successfully achieved by Zuse in Berlin in 1941. In this thesis we will use the Turing machine as the standard model of computation.

The assertion that every *intuitively* computable function can be computed by a Turing machine is known as the Church-Turing thesis (or Church's thesis – see [74]), and the study of models of computation that violate this assertion has been termed, by Copeland and Proudfoot, in 1999, as "hypercomputation" (see [32]). Some examples of such models are Turing's O-machines, accelerated Turing machines and infinite time Turing machines.[3] Interestingly, some models of hypercomputation, in a more loose sense, can also be found in Literature. Take, for instance, Frank Herbert's science fiction saga, Dune, in which one of the character, Paul Atreides, the *Kwisatz Haderach*, is "an intellect whose capacities surpassed those of the religiously proscribed mechanical computers used by the ancients" (see [46]).

Some hypercomputation models draw their power from physical theories (see [24] and [78]). A famous example is the neural net model, which, in 1995, Siegelmann proved to compute more than a Turing machine (see [71]). More recently, Beggs and Tucker introduced an abstracted physical process, called the scatter experiment, capable of approximating any real number up to any precision, by shooting particles at a sharp wedge vertex (see [21]). Therefore, if we could couple the experiment with a Turing machine, we could use the position of the wedge as an *advice*, to help the machine with a given decision problem. To understand how this cooperation is achieved, we have to go back to the 1930's.

[1] Algorithmic procedures can be traced back as far as the Babylonians (see [39]) and the Egyptians (see [50]). The name "algorithm" comes from the name of the ninth century Persian mathematician Abu Ja'far Mohammed ibn Mûsâ al-Khowârizm (see [62])

[2] The Bank of England has issued a new £50 note featuring Alan Turing.

[3] It has been argued that accelerated Turing machines do not actually perform hypercomputation (see [70]). For more examples of models of hypercomputation, see, for example, [60] or [78].

In his thesis from 1939, Turing introduced the notion of an oracle machine (see [83]), in which a standard Turing machine is equipped with an oracle tape, that allows it to query an oracle during the computation. This notion was later developed by Post, who introduced what he termed the “Turing reducibility" (see [63], [64] and [75]). Then, using this idea, in [11] and [12] the authors combined a Turing machine with the scatter experiment from [21], by viewing it as an oracle that the machine would query by running the physical experiment. With this formalisation, they classified the computational power of this model using non-uniform complexity classes, which can be characterized by giving a Turing machine an *advice* function that depends only on the length of the input.

Since these systems combine a digital component (the Turing machine) and an analogue[4] component (the experiment), they are said to be analogue-digital, or hybrid.[5] Since, unlike the standard oracle consultation, which is done in a single transition, a consultation of the physical oracle requires the execution of a physical experiment, we have to be careful with how the data is communicated between the analogue and the digital part of the machine and how we should classify the performance of such models of computation (see [11]). Therefore, the coupling of the Turing machine with the physical oracle is mediated by “protocols", which define how the data is exchanged between them.

Studying a different experiment in [16], the collider experiment, the authors stated the BCT conjecture: for every “reasonable" physical theory, which supports measurement experiments, the time intrinsic to an experiment is at least exponential.[6] This implies a bound on the power of analogue-digital machines, clocked in polynomial time, which was expressed in [15] as an analogue-digital Church-Turing thesis: “No possible abstract analogue-digital device can have more computational capabilities in polynomial time than $\mathtt{BPP}//\log\star$". Note that this bound restricts the computational power of Siegelmann's neural net model, which could decide any set in $\mathtt{P}/\mathtt{poly}$.

After the initial scatter experiment, a new scatter machine was introduced in [19]: the SmSM, which coupled a Turing machine with the SmSE. The authors, referring to the original scatter machine, argued that “the sharp wedge is (...) unrealistic, and, therefore, a model that removes the discontinuity should be sought". The sharp wedge was then replaced by an arc of a C^n smooth curve, with nth derivative near the vertex. This change resulted in a hyperbolic time for an experiment to be completed. We will assume the BCT conjecture and consider the SmSM as the standard model of analogue-digital computation.

In this thesis we will present this model's computational power, when polynomial time restrictions are imposed, and see that they can be viewed as ideal technicians, controlling a measurement experiment. We will see how to make the leap from comparing object in a domain to being able to assign them a numerical value, using the axiomatization of measurement given by Carl Hempel in [44]. We will then present the work from [16], in which the time of an experimental call was introduced in the axiomatiza-

[4]The term analogue comes from the fact that, unlike a digital computer, which has a fixed internal structure, an analogue computer is programmed by changing its structure until it forms a model (an analogue) of a given problem (see [84]).

[5]Hybrid machines, such as the ADDAVERTER, were built around the sixties. For a more recent example see [57].

[6]Other interested reader may refer to [5], where different experiments are analysed and proven to satisfy this statement

tion and Hempel's notion was obtained as a limit concept. We will propose a different axiomatization of measurement, which arises from a particular way of performing comparisons, and prove that, in the limit, we can again recover Hempel's notion of a measurement procedure. As a consequence of introducing the notion of time in an experiment, we study the numbers whose binary expansion takes a given amount of time to be obtained. This concept will motivate the definition of a measurable number and the classification of a real number according to how long it takes to obtain its binary expansion.

Arguments have been raised against the possibility of actually achieving hypercomputation. For example, Davis remarks the need for non-computable properties, a priori, to be able to "program" a hyper-machine, and the fact that no finite amount of data is enough to distinguish a computable from a non-computable sequence (see [35] and [36]). Regarding hybrid models that surpass the Turing barrier, it is argued in [29] that the models of hypercomputation that draw on the notions of infinite or continuous "are not materially realisable and so cannot constitute new forms of effective calculability".

As for the SmSM performing hypercomputation, these models fall under the previously presented critiques, as we can only program a SmSM to decide a non-computable set when we encode a wedge vertex position into a (necessarily infinite) non-computable real number. However, we are not interested in performing hypercomputation; instead, we are only concerned with classifying these models' computational power and the results obtained by viewing them as measurement experiments.

So, to conclude this introduction, why should we be concerned with hypercomputation? On the side of realisability, even if no hyper-machine can ever be effectively developed, it motivates the study of different computing paradigms, which may come to avoid the "doomsday for computing" (see [78]). From the theoretical point of view, we refer to Davis again (see [36]), who, while presenting his critics to the field, calls it an "inviting subject for theoretical study" and remarks the work by Ord and Kieu in [61], who were brought together by their common interest in hypercomputation.

As Teuscher and Sipper wrote in [79]: "So, hype or computation? At this juncture, it seems the jury is still out – but the trial promises to be riveting".

1.2 Organization of the document

In chapter 2 we introduce the SmSM and its computational power, when a polynomial time constraint is imposed and the time schedule is taken to be an exponential function; chapter 3 regards the study of the measurement performed by the SmSM from the perspective of fundamental measurement; in chapter 4 we study the concept of a measurable number, classifying certain classes of real number according to their measurement complexity; finally, in chapter 5, we end with a summary of the content of this thesis and with a survey of topics for future research.

2

State of the art

2.1 Complexity classes

This section serves as a small introduction to complexity classes. The reader is directed to [6] or [7] for a more thorough exposition of the topic.

An alphabet is any non-empty finite set, which we will represent with the Greek letter Σ. From now on, Σ will denote the set $\{0, 1\}$. We say that the elements of an alphabet are letters, or symbols, and denote by $\Sigma^\star$ the set of all words over Σ, i.e., finite sequences of letters of Σ. The special word which contains no letters is denoted by ε. A set $A \subseteq \Sigma^\star$ is called a language and its characteristic function is the one which, given a word in A, returns 1, and given a word in $\Sigma^\star \setminus A$, returns 0. We are interested in defining algorithms that can implement the characteristic function of a given language.

We define a Turing machine as an abstract object containing: a finite set of tapes, for storing information, a transition function, which details how the machine should work, and a finite set of internal states. Two types of states which play an important role are the initial state, the state where the computation begins, and the final state, which can be accepting or rejecting and is used to distinguish the inputs that are to be accepted from those that are not. The transition function, which can be deterministic,

non-deterministic, or have its computations determined probabilistically, determines the type of Turing machine we are working with.

We say that a set is decidable if it can be decided by a Turing machine (the decision criteria depends on the type of machine we consider). An example of a non-decidable set is the halting set, which consists of codes of Turing machines that halt on input 0 (see, for example, [37]). We say that a real number y is computable if there is a Turing machine capable of approximating y to any given any precision. As an example of a non-computable value, consider the number whose nth binary place is 1 if and only if the nth program halts on input 0.

Consider a function $f : \mathbb{N} \to \mathbb{N}$. We say that f is computable if it can be computed by a Turing machine and *time constructible* if there is a Turing machine which, for a constant k, halts in exactly $f(n)$ steps, for every input of length $n \geq k$. We define $O(f)$ as the class of functions g, for which there are positive constants C and k, such that $g(n) \leq Cf(n)$, for all $n \geq k$. We define $\Omega(f)$ as the class of functions g, for which there are positive constants C and k, such that $g(n) \geq Cf(n)$, for all $n \geq k$.

An oracle Turing machine is one that is coupled with an an external device, called an oracle, which asserts the belonging of a word to a given set. These Turing machines have a new type of state, the query state, in which the machine writes a binary word in a special tape, the query tape, and sends it to the oracle. This process of querying and receiving an answer is assumed to be done in a single transition i.e., in a single step of the computation.

Complexity theory deals with the computational power of Turing machines when a constraint on resources, such as time or space, is imposed. Usually, we define these constraints with respect to the length of the machine's input. For a polynomial constraint, for example, we have the classes P and PSPACE, depending on whether the constraint is placed in the time or the space available. In the first case, we say that a set is in P if it can be decided by a Turing machine *clocked* in polynomial time.

2.1.1 Probabilistic computation

"Making people take an unconventional step requires compelling reasons, and indeed the study of randomized algorithms was motivated by a few compelling examples" (see [42]).

A randomized algorithm is one that has access to a random number generator, like the tossing of a fair coin, which it can use to decide the next step at several branches of its computations. The interest in studying such algorithms is that randomness can imply a tremendous decrease in the running time, although with a cost: the answer may have some probability of being incorrect (see [48] and [58]).

A famous example of an algorithm that relies on randomness is due to Rabin, who, in 1977, devised a probabilistic version of an algorithm for testing primality. Rabin's algorithm, instead of running in $O(\log^4(n))$ steps, ran in $O(\log^2(n))$ (see [56] and [65]).

We will introduce the concept of a probabilistic Turing machine and present four probabilistic com-

plexity classes. These were first defined by Gill, Adleman and Manders and always impose a polynomial time restriction on the Turing machines that define them (see [41] and [3]).

A probabilistic Turing machine M is a type of non-deterministic Turing machine where each nondeterministic step, called a *coin-flip step*, has two possible next moves. We will make the assumption that all the computations of a probabilistic Turing machine have the same depth. Let k be the number of coin-flip steps that occur on branch b, i.e., the depth of branch b. Then, the probability of b is defined as $P(b) = 2^{-k}$. The probability of a probabilistic Turing machine M accepting a word w is defined as the sum of the probability of each accepting branch of M. The probability of M rejecting w is defined as $P(M \text{ rejects } w) = 1 - P(M \text{ accepts } w)$.

A word w is considered accepted by a probabilistic machine M if and only if more than half the computations of M on w end in the accepting final state. Since we suppose that all computations have the same length, we can state the acceptance criteria as: "the ratio of accepting paths to the total number of computations is greater than $1/2$". The error probability of a probabilistic Turing machine is the probability of it wrongly accepting or rejecting a word.

We will now define four probabilistic complexity classes, according to the probabilistic Turing machines that can decide them. These Turing machines are of the "Monte Carlo" type, if they are allowed to make a mistake, and of the "Las Vegas" type, if the may terminate with the symbol "?", but are not allowed to make a mistake.[1] The first three classes are of the Monte Carlo type and the last of the Las Vegas type.

Definition 2.1. The class PP, for Polynomial Probabilistic time, is the class of languages accepted by polynomially clocked probabilistic Turing machines.

The problem with the class PP is that repeating a computation a polynomial number of times may not be enough to guarantee a given reliability. This happens because an algorithm may have an error probability which approaches $1/2$ as the size of the input increases, so we could potentially require an exponential number of trials to achieve a fixed error probability (see [48]).

We will thus consider classes for which the error probability, or, in the last case, the probability of having the result "don't know", is limited by a constant smaller than $1/2$.

Definition 2.2. The class RP, for Randomized Polynomial time (also denoted by R – see [7]), is the class of languages accepted by polynomially clocked probabilistic Turing machines, which have zero error probability, for inputs not in the language, and error probability bounded above by some positive constant $\varepsilon < 1/2$, for inputs in the language.

[1] For this type of algorithms, we add to the definition of a Turing machine the final state "don't know". The error probability of these machines is calculated considering that the result "don't know" is not an error. We may refer to these machines as 3-output probabilistic Turing machines.

RP-algorithms are known as yes-biased Monte Carlo algorithms, because a “yes" answer is always correct, but a “no” answer might be incorrect.

Definition 2.3. The class BPP, for bounded-error probabilistic polynomial time, is the class of languages recognized by polynomially clocked probabilistic Turing machines, whose error probability is bounded above by some positive constant $\varepsilon < 1/2$.

Definition 2.4. The class ZPP, for Zero-error Probabilistic Polynomial time, consists of the languages recognized by polynomially clocked 3-output probabilistic Turing machines, with zero error probability, such that the probability of the outcome being “don’t know" is bounded above by some positive constant $\varepsilon < 1/2$.

This means that a “yes" or “no" answer will always be correct, but, with a given probability, the machine may terminate in the state “don’t know".

Proposition 2.5. A set A is in BPP if and only if, for each polynomial p, a polynomial time probabilistic machine can be constructed, which accepts A with an error probability of at most $(1/2)^{p(|x|)}$.

A set A is in RP if and only if, for each polynomial p, a polynomial time probabilistic machine can be constructed which accepts A with an error probability of at most $(1/2)^{p(|x|)}$, on inputs in A, and zero error probability on inputs not in A.

A set A is in ZPP if and only if, for each polynomial p, a polynomial time probabilistic machine can be constructed, for which return “don’t know" with a probability of at most $(1/2)^{p(|x|)}$, on any input.

Proof. We will only sketch the proof, but the result can be found, for example, in [72] and [42]. The idea is that, given a probabilistic Turing machine M, we can make another machine M' that will simulate M a polynomial number of times. For the case of BPP, M' returns the most often outcome from M. For the case of RP, M' returns “yes", if there is a run from M that returned “yes", and “no", otherwise. Finally, for the case of ZPP, M' returns “don’t know", if every run on M returned “don’t know", and, otherwise, the “yes" or “no" answer that occurred (which would be unique by definition of ZPP). □

Therefore, we can, in polynomial time, reduce the error associated with a probabilistic Turing machine to a value so small, that it will be more likely for an undetected hardware failure to occur, than for the program failing due to its inherent error probability.

2.1.2 Non-uniform complexity

A non-uniform complexity class is a way of characterising families $\{C_n\}_{n\in\mathbb{N}}$, of finite machines, where each element C_n decides a restriction of some problem to inputs of size n. Non-uniformity arises because, for $n \neq m$, C_n may be unrelated to C_m, so there might be no computable way to call upon each C_n for every possible input size.

In 1980, Karp and Lipton introduced the notion of an *advice* function, which could be used to unify a non-uniform class under a single algorithm, by providing it with enough additional information (see [49]). The sequence would thus be divided into a uniform part, given by the program, and a non-uniform part, given by the advice function. We can see the size of the advice as a *measure* of the non-uniformity of a sequence. In this section we will introduce non-uniform complexity classes, by means of an advice function[2], which will be used when classifying the computational power of the SmSM. For a further reading on the topic see, for example, [9] and Chapter 5 of [7].

Consider the pairing function, $\langle .,. \rangle : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$, which duplicates every symbol in both words and assigns 01 to the separation symbol. An advice function is a total map $f : \mathbb{N} \to \Sigma^\star$ and a prefix advice function is an advice function with the extra condition that $f(n)$ is a prefix of $f(n+1)$.

We will consider the classes $\mathtt{poly}$ and $\mathtt{log}$, of functions whose size is bounded by some polynomial or logarithmic function, respectively.

Definition 2.6. Let $\mathcal{C}$ be a class of languages and $\mathcal{F}$ a class of integer-valued functions. The non-uniform class $\mathcal{C}/\mathcal{F}$ is defined as the sets B for which there is $f \in \mathcal{F}$ and $A \in \mathcal{C}$ such that $w \in B$ if and only if $\langle w, f(|w|) \rangle \in A$.

So if $\mathcal{C}$ is a complexity class, the sets $B \in \mathcal{C}/\mathcal{F}$ are those for which a function in $\mathcal{F}$ provides enough additional information to decide B within the bounds specified by $\mathcal{C}$. Note that we are not saying that the set B is in $\mathcal{C}$; instead, we are saying that the decision problem "is w in B", is equivalent to the decision problem "is $\langle w, f(|w|) \rangle$ in A", for some $A \in \mathcal{C}$.

Examples of such classes are $\mathtt{P/poly}$ and $\mathtt{P/log}$, of the sets that can be decided by a Turing machine clocked in polynomial time, with a polynomial or logarithmic advice function, respectively. Note that, in this case, allowing for the advice function to be polynomial yields a strict increase in computational power, i.e., $\mathtt{P/log} \subsetneq \mathtt{P/poly}$. If we consider $\mathtt{exp}$, the set of advice functions bounded in size by functions in the class $2^{O(n)}$, then $\mathtt{P/exp}$ contains all sets (see [86]).

Definition 2.7. Given a class $\mathcal{C}$ and a set of prefix advice functions $\mathcal{F}$, the class $\mathcal{C}/\mathcal{F}\star$ is defined as the class of sets B for which there is $A \in \mathcal{C}$ and $f \in \mathcal{F}$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^\star$, with $|w| \leq n$, $w \in B \;\Leftrightarrow\; \langle w, f(n) \rangle \in A$.

In some cases the advice and prefix advice classes coincide, but this is not always the case. For example, $\mathtt{P/poly} = \mathtt{P/poly}\star$, but $\mathtt{P/log}\star \subsetneq \mathtt{P/log}$ (see [87]).

For the classes of the form $\mathcal{C}/\mathcal{F}$, the advice function is fixed after choosing the Turing machine that decides a given set A. For the case of probabilistic computation, this means that we first choose a Turing machine, with an associated error ε, and then find the advice function. We will consider a less restrictive definition, where we fix the Turing machine *after* the suitable advice function has been chosen.

[2]The interested reader is directed to Section $1.2.4$ of [42], where both a characterization by means of advice functions and by means of Boolean circuits are given.

Definition 2.8. Given a class $\mathcal{C}$ and a set of prefix advice functions $\mathcal{F}$, $\mathcal{C}//\mathcal{F}$ is the class of sets B, for which, given a prefix advice function $f \in \mathcal{F}$, there is $A \in \mathcal{C}$ such that, for every word w, with $|w| \leq n$, we have that $w \in B \Leftrightarrow \langle w, f(n)\rangle \in A$.

An example is the class $\mathtt{BPP}//\log\star$, of sets B for which, given a prefix advice function $f \in \log$, there is a probabilistic Turing machine M and a constant $\gamma < 1/2$ such that, for every word w with $|w| \leq n$, the probability of M rejecting $\langle w, f(|n|)\rangle$, with $w \in B$, or accepting $\langle w, f(|n|)\rangle$, with $w \notin B$, is, at most, γ.

While it holds that $\mathcal{C}/\mathcal{F} \subseteq \mathcal{C}//\mathcal{F}$, for any complexity class $\mathcal{C}$ and set of advice functions $\mathcal{F}$, it is unknown whether or not, for the case of probabilistic classes[3], the other inclusion must hold. It holds, for example, that $\mathtt{BPP}//\mathtt{poly} = \mathtt{BPP}/\mathtt{poly}$, but it is still an open problem to know if $\mathtt{BPP}//\log\star$ is contained in $\mathtt{BPP}/\log\star$ (see [11]).

In this work we are interested in the non-uniform complexity classes $\mathtt{P}/\log\star$ and $\mathtt{BPP}//\log\star$, which, since $\mathtt{P} \subseteq \mathtt{BPP}$, satisfy the inclusion $\mathtt{P}/\log\star \subseteq \mathtt{BPP}//\log\star$.[4] For both of these classes, we have that the length of an advice function $f(n)$ is given by $a\lceil\log(n)\rceil + b$, for some a and b that depend on f (see [8]).[5]

Finally, it is important to note that these non-uniform complexity classes contain non-decidable sets. For example, $\mathtt{P}/\mathtt{poly}$ contains the sparse halting set and $\mathtt{P}/\log$ contains the halting set defined as $\{0^{2^n} : n$ codes for a TM that halts on input $0\}$. However, as shown in [7], there are sets in $\mathtt{EXPSPACE}$, and therefore decidable, that are not in $\mathtt{P}/\mathtt{poly}$. It is also interesting to note that, since we can just simulate a computable advice within the algorithm, a computable advice can only result in a computable behaviour, with a possible speed up.

2.2 The Smooth scatter machine model

We are considering a physical oracle that enables a Turing machine to perform a measurement. Of the three types of measurement presented in [20], one-sided, two-sided and vanishing, we will only study the two-sided case. One sided and vanishing type measurements have been studied in [13] and [14], respectively, and will be presented in Section 3.4.

In this section we will describe how the SmSM works and, for each communication protocol between the digital and analogue component, present its computational power, when a polynomial time restriction is imposed and the schedule is taken to be an exponential time constructible function. These computational results, for two-sided type experiments, appeared for the first time in [5] and [13].

[3] If $\mathcal{C}$ is not a probabilistic class, it doesn't matter if we choose the advice function before or after the Turing machine has been fixed.

[4] It is unknown whether or not the reciprocal inclusion holds (see, for example, [6]).

[5] In reality, the authors consider the size of a logarithmic advice to be just $c\log(n)$, for some constant c. However, since adding a constant does not change the logarithmic size of an advice, our form is equivalent to theirs.

2.2.1 Description and time of the experiment

A SmSM combines a digital computation, performed by a Turing machine, and an analogue computation, performed by the SmSE. The SmSE is setup with a curve that is n times differentiable at its wedge, which is placed at $y \in (0, 1)$, such that the curve is symmetrical with respect to the wedge position. The SmSE is governed by a fragment of Newtonian mechanics, consisting of the following laws and assumptions:

- Particles obey Newton's laws of motion in the two dimensional plane;
- collisions between barriers and particles are perfectly elastic, i.e., kinetic energy is preserved;
- the barriers are rigid and do not deform upon impact;
- the cannon projects a particle with a given velocity and direction;
- the detectors are capable of telling if a particle has crossed them.

In order to invoke the SmSE, the Turing machine writes a word z in the query tape and enters the shooting state: the cannon is aimed at z and an experiment is run. The particle thrown with velocity v will be reflected off the wedge and the experiment will inform the machine if the particle reached a collecting box, and, if so, in which collecting box it was detected.

The SmSM is also setup with a protocol, which determines how accurate the positioning of the cannon is, and a schedule $T : \mathbb{N} \to \mathbb{N}$, whose time to compute an input k determines how long the machine waits for an experiment to return an answer. The physical duration of an experiment, when performed with a query z, is denoted by $t(z)$.

The schedule is intended to represent a physical clock, which counts the duration of an experiment. We thus make the additional assumption that the unit of physical time we consider, denoted by Δt, relates to the physical amount of time of a machine's step, denote by ΔT, according to an expression of the form $\Delta t = h(n)\Delta T$, where n is the nth step of the machine. We will only consider the case where h is constantly equal to 1, meaning that a machine's step represents a unit of physical time (e.g. 1μs).

Even though a schedule can be any time constructible function, we will be interested in machines that operate with an exponential schedule.[6] Unless explicitly stated otherwise, the letter y will represent a wedge vertex (and the real number corresponding to its position) and the letter T a schedule. The upper collecting box will be called *right collecting box* and the lower one will be called *left collecting box*. A schematic drawing of the SmSE is represented in figure 2.1.

The word placed in the query tape is either 1 or a binary word beginning with 0. We will use z_i to denote the ith bit of z and the letter z to denote both the word $z_1, \ldots, z_n \in \{1\} \cup \{0s : s \in \{0, 1\}^\star\}$ and the corresponding dyadic rational $z_1 + 2^{-1}z_2 + \cdots + 2^{-n+1}z_n$. We write $|z|$ to denote n, i.e., the size of $z_1, \ldots, z_n$, and say that the SmSM is aiming at z. If we know n binary digits of z, we say that we know z to a *precision* of 2^{-n} or to an *accuracy* of 2^n. Note that, by padding each query z with a finite number of

[6] The fact that these functions are time constructible will be discussed in Section 4.2.

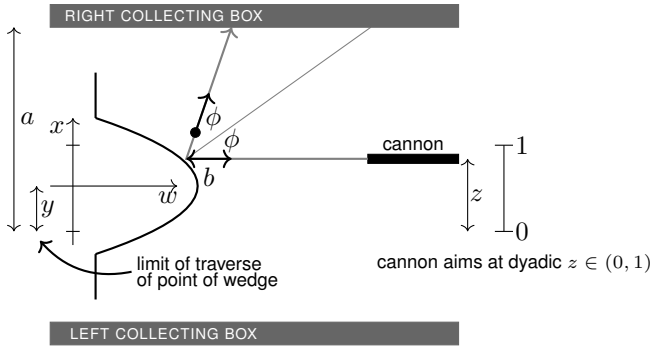

Figure 2.1: Schematic representation of the SME.

0's, we can assume that every query has the same size.

Proposition 2.9. Suppose the shape of the edge of the wedge is given by a function $w(x)$, which is n times continuously differentiable near $x = 0$, and that the first non-zero derivative of w, at $x = 0$, is its nth derivative. Then, for a position z such that $|y - z|$ is sufficiently small, the SmSE takes an hyperbolic time to return an answer, i.e., for some $A, C > 0$, we have:

$$\frac{A}{|y-z|^{n-1}} \leq t(z) \leq \frac{C}{|y-z|^{n-1}} \tag{2.1}$$

Proof. Using the (x, w) coordinate system, with the w axis drawn horizontally and the x axis drawn vertically, the shape of the wedge can be given by a function $w(x)$, which takes a single maximum value at $x = 0$, corresponding to the position of the wedge vertex. We denote by y the distance between the vertex and the zero line of the cannon. Let ϕ be the angle of incidence of a particle to the normal of the curve at the point of impact, which has x coordinate $x = z - y$. If we suppose that $x > 0$ (the other case is analogous), then $\tan\phi = -w'(x)$ and the incident particle travels at an angle 2ϕ to the horizontal. We then have:

$$\tan(2\phi) = \frac{\sin(2\phi)}{\cos(2\phi)} = \frac{2\sin(\phi)\cos(\phi)}{\cos^2(\phi) - \sin^2(\phi)} = \frac{2\frac{\sin(\phi)}{\cos(\phi)}}{1 - \frac{\sin^2(\phi)}{\cos^2(\phi)}} = \frac{2\tan(\phi)}{1 - \tan^2(\phi)} = -\frac{2w'(x)}{1 - w'^2(x)}$$

Let a be the distance between the zero line of the cannon and the right collecting box, and let b be the horizontal distance between the collision point with the curve and the collision point with the right collecting box. We have that:

$$\tan(2\phi) = \frac{a-z}{b} \Rightarrow b = \frac{a-z}{\tan(2\phi)} = -(a-z)\frac{1 - w'^2(x)}{2w'(x)}$$

Denoting by c the distance between the cannon and the flat part of the wedge, we know that the distance

traveled by the particle is given by:

$$d(x) = c - w(x) + \sqrt{(a-z)^2 + b^2} = c - w(x) + (a-z)\sqrt{1 + \left(\frac{1 - w'^2(x)}{2w'(x)}\right)^2}$$

Since we consider that all the shots are made with the same speed v and $t(z) = dist(z-y)/v$, we have:

$$t(z) = \frac{1}{v} \times \left(c - w(z-y) + (a-z)\sqrt{1 + \left(\frac{1 - w'^2(z-y)}{2w'(z-y)}\right)^2}\right)$$

For the upper bound case, consider that $c - w(z-y) \leq c$, $a - z \leq a$ and $1 - w'^2(z-y) \leq 1$. Therefore, we conclude that:

$$t(z) \leq \frac{1}{v} \times \left(c + a\sqrt{1 + \left(\frac{1}{2w'(z-y)}\right)^2}\right) \leq \frac{1}{v} \times \left(c + a + \frac{a}{2w'(z-y)}\right) \tag{2.2}$$

Now suppose that the function $w(x)$ is $n \geq 2$ times continuously differentiable near $x = 0$. Then, as $w(x)$ has a maximum at $x = 0$ we have that $w'(0) = 0$. Now, let n be the order of the first non-zero derivative of w at $x = 0$. Then using Taylor's theorem with remainder, for the function w', we have that, for some ξ between 0 and x,

$$w'(x) = \frac{w^{(n)}(\xi)x^{n-1}}{(n-1)!}$$

whence, for some neighbourhood of $x = 0$, we have constants $D, E > 0$ such that $D|x|^{n-1} \leq w'(x) \leq E|x|^{n-1}$. Finally, from equations 2.2, and considering that w' has a bounded value in the interval $(0, 1)$, there is $C > 0$ such that $t(z) \leq C/|y-z|^{n-1}$ (analogously, we can find a lower bound for $t(z)$). Therefore, there are constants $A, C > 0$ such that, for $|y - z|$ close enough to 0,

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

□

Note that the result would still be valid if the curve were to be $m > n$ times differentiable, or even infinitely differentiable, near $x = 0$. We only require that the first non-zero derivative is the nth derivative.

Remark 2.10. The previous proposition establishes an asymptotic lower and upper bound for the time of an experiment, which depends on the first non-zero derivative of w. However, as has been remarked in [5], we can assume, without loss of generality, that $n = 2$, as the computational power of this model is

the same for values of $n > 2$. We will thus assume that, for some constants $A, C > 0$,

$$\frac{A}{|y-z|} \leq t(z) \leq \frac{C}{|y-z|} \tag{2.3}$$

Moreover, since we are only interested in the asymptotic behaviour of an algorithm, we will use these bounds as if they were valid for every experiment, as only a finite number of first firings will violate this assertion.

From now on, the letters A and C will always represent these constants.

We are interested in studying the class of sets that can be decided by the SmSM, when clocked in polynomial time. We thus have the following definitions.

Definition 2.11. Let $B \subseteq \Sigma^\star$ and M be an error-free SmSM, clocked by a polynomial $p(n)$. We say that M decides B if, for every $w \in \Sigma^\star$, M accepts w in, at most, $p(|w|)$ steps, if $w \in B$, and M rejects w in, at most, $p(|w|)$ steps, if $w \notin B$.

Definition 2.12. Let $B \subseteq \Sigma^\star$ and M be an error-prone SmSM, with unbounded (fixed) precision, clocked by a polynomial $p(n)$. We say that M decides B if there is a constant $\gamma < 1/2$ such that, for every $w \in \Sigma^\star$, M satisfies the following condition, with an error probability of γ: if $w \in B$, then M accepts w in $p(|w|)$ steps; if $w \notin B$, then M rejects w in $p(|w|)$ steps.

2.2.2 Communication with the SmSE and measurement algorithms

We consider that our experiment can set the cannon's position in three different ways, depending on the protocol that rules the data exchange. For each protocol, we will have to consider a different algorithm for measuring the position of the wedge vertex. For the error-prone protocols, we will assume that the position of cannon z' is chosen uniformly. The reader may refer to [10], where it is proven that this assumption can be generalized without a change in the computational power of these models. The three protocols are represented in Figures 2.2, 2.3 and 2.4. A SmSM may be classified as being *error-free*, *error-prone* with *unbounded precision*, or *error-prone* with *fixed precision*, depending on the protocol with which it is defined.

PROTOCOL PROT_IP(z)

Receive as input a dyadic rationals z of size n;
Run an experiment with the cannon set to z and wait constructive $T(n)$ units of time;
If the particle reaches the left collecting box, return q_l;
If the particle reaches the right collecting box, return q_r;
If the particle doesn't reach one of the collecting boxes in $T(n)$ units of time, return q_t.

Figure 2.2: Protocol for the infinite precision case.

PROTOCOL PROT_UP(z)

Receive as input a dyadic rationals z of size n;
Run an experiment with the cannon set to $z' \in [z - 2^{-n}, z + 2^{-n}]$ and wait constructive $T(n)$ units of time;
If the particle reaches the left collecting box, return q_l;
If the particle reaches the right collecting box, return q_r;
If the particle doesn't reach one of the collecting boxes in $T(n)$ units of time, return q_t.

Figure 2.3: Protocol for the unbounded precision case.

PROTOCOL PROT_FP(z)

Receive as input a dyadic rationals z of size n;
Run an experiment with the cannon set to $z' \in [z - \varepsilon, z + \varepsilon]$ and wait constructive $T(n)$ units of time;
If the particle reaches the left collecting box, return q_l;
If the particle reaches the right collecting box, return q_r;
If the particle doesn't reach one of the collecting boxes in $T(n)$ units of time, return q_t.

Figure 2.4: Protocol for the fixed precision case.

From now on, we will denote by $z\lfloor_l$ the first l digits of z, if z has at least l digits, or z padded with k zeros such that $|z0^k| = l$, otherwise. We will now present, for each protocol, an algorithm that can be used to measure the position of the wedge vertex of the SmSM.

2.2.2.A Infinite and unbounded precision

For the infinite precision, and for unbounded precision as well, the only change being the call for the protocol, we have the measurement Algorithm 2.1. In the case of the unbounded precision, we set the error of the cannon to be 2^{-l}.

Algorithm 2.1: Measurement algorithm for infinite and unbounded precision.

Data: Positive integer l, representing the desired accuracy;
$z_0 = 0$; $z_1 = 1$; $z = 0$;
while $z_1 - z_0 > 2^{-l}$ **do**
  $z = (z_0 + z_1)/2$;
  $s =$ Prot_IP($z\lfloor_l$) (resp. Prot_UP($z\lfloor_l$));
  if $s == q_r$ **then**
    $z_1 = z$;
  if $s == q_l$ **then**
    $z_0 = z$;
  else
    $z_0 = z$; $z_1 = z$;
return z;

Proposition 2.13. Consider a SmSM, with wedge vertex position y and time schedule T, and let s be the result of Prot_IP($z\rfloor_l$). Then,

- If $s = q_l$ (resp. q_r), then $z\rfloor_l$ is smaller (resp. greater) than y;
- if $s = q_t$, then $|y - z\rfloor_l| < C/T(l)$.

Proof. The first statement is obvious. For the case where a time out occurred, we have:

$$\frac{C}{|y - z\rfloor_l|} \geq t(z\rfloor_l) > T(l) \;\Leftrightarrow |y - z\rfloor_l| < C/T(l)$$

□

Proposition 2.14. Consider a SmSM with wedge vertex position y and time schedule T, and let s be the result of Prot_UP($z\rfloor_l$). Then:

- If $s = q_l$, then $z\rfloor_l < y + 2^{-l}$;
- if $s = q_r$, then $z\rfloor_l > y - 2^{-l}$;
- if $s = q_t$, then $|y - z\rfloor_l| < C/T(l) + 2^{-l}$.

Proof. Let z' be the (uniformly) chosen position by Prot_UP($z\rfloor_l$). Then, $z' \in [z\rfloor_l - 2^{-l}, z\rfloor_l + 2^l]$. If $s = q_l$, then $z\rfloor_l - 2^l \leq z' < y$, whence $z\rfloor_l < y + 2^{-l}$; if $s = q_r$, then $z\rfloor_l + 2^l > z' > y$, whence $z\rfloor_l > y - 2^l$; if a time out occurs, then $C/|y - z'| \geq t(z') > T(l)$, whence $|y - z'| < C/T(l)$ and $|y - z\rfloor_l| < C/T(l) + 2^{-l}$. □

Proposition 2.15. Consider a SmSM, with unbounded or infinite precision, with vertex position at y and time schedule $T(l) = D2^l$, for $D \geq C$, running Algorithms 2.1 with input l. We then have:

- The time complexity of the measurement algorithm is $O(lT(l))$;
- with infinite precision, the output of the algorithm is a dyadic rational z, such that $|y - z| < 2^{-l}$;
- with unbounded precision, the output of the algorithm is a dyadic rational z, such that $|y - z| < 2^{-l+1}$

Proof. The first statement comes from the fact that the call to the protocol is repeated l times, each taking $O(T(l))$ steps. The other statements are an immediate consequence of the halting condition of Algorithms 2.1. In the infinite precision case, the execution of the algorithm halts when $t(z) > T(l)$, which happens when $|y - z| < C/T(l) \leq D/T(l) = 2^{-l}$. In the worst case of the unbounded precision, the execution of the algorithm halts when $|y - z| < C/T(l) + 2^{-l} \leq 2^{-l} + 2^{-l} = 2^{-l+1}$. □

2.2.2.B Fixed precision

For the fixed precision case we have Algorithm 2.2. The idea is to make enough firings to get a result with a small margin of error. The reason for the different increments of c will become clear in the proof of Proposition 2.16.

Algorithm 2.2: Measurement algorithm for fixed precision, with an error smaller than 2^{-h}.

Data: Positive integer l, representing the desired accuracy;
$x = 0$; $i = 0$; $\xi = 2^{2l+h}$;
while $i < \xi$ **do**
  $s =$ Prot_FP$(0.1\downarrow_l)$;
  if $s == q_l$ **then**
    $c = c + 2$;
  if $s == q_t$ **then**
    $c = c + 1$;
  i++;
return $c/(2\xi)$;

The following statement appears for the first time in [11], for the case of the scatter experiment, and in [5] and [13], for the case of the SmSM and the BBE[7], respectively. We present here a more detailed proof of the result.

Proposition 2.16. Consider an error-prone SmSM, with fixed precision $\varepsilon = 2^{-q}$, wedge vertex position at $y = 1/2 - \varepsilon + 2s\varepsilon$, for $s \in (0, 1)$ and time schedule T, running Algorithms 2.1 with input l. Then, for an error probability of $\delta = 2^{-h}$, we have:

- The time complexity of the measurement algorithm is $O(2^{2l+h}T(l))$;
- the output of the algorithm is a dyadic rational m, such that $|s - m| < 2^{-l}$, with error probability less than 2^{-h}.

Proof. The first statement derives from the fact that the Algorithm 2.2 perform the experiment 2^{2l+h} times, each taking a time bounded by the schedule.

For the second part, our method for guessing digits of s begins by commanding the cannon to shoot ζ times at $z = 1/2(\pm\varepsilon)$. Note that $y \in [1/2 - \varepsilon, 1/2 + \varepsilon]$, which is the interval with the possible position of the particle, when we aim at $z = 1/2$. Denote by $\eta > 0$ the value such that the experiment runs out of time in the interval $(y - \eta, y + \eta)$.

- The event $z < y$ happens if z is in the interval $[1/2 - \varepsilon, y - \eta]$, which happens with a probability $p = (y - \eta + \varepsilon - 1/2)/(2\varepsilon)$;
- the event $y < z$ happens with probability $q = (1/2 + \varepsilon - y - \eta)/(2\varepsilon)$;
- the event time out happens with probability $r = 2\eta/(2\varepsilon)$.

So, our experiment can be modeled as a multinomial distribution with three categories of success, each one with the stated probabilities. Denote by α, β and γ the random variables that count how many times the events $z < y$, $y < z$ and time out happen, respectively and consider the random variable,

[7] This experiment will be presented in Section 3.4.2

$X = (2\alpha+\gamma)/(2\zeta)$. This way, combining the event $z < y$ and "half" of the event time out into a single one, X represent the probability that z is smaller than y, including the case where we don't get a response due to a time out (recall the increment of c in Algorithm 2.2). Let $Y = 2\zeta X$. Then,

$$\overline{Y} = 2\overline{\alpha} + \overline{\gamma} = 2p\zeta + r\zeta = \zeta\left(2\frac{y-\eta+\varepsilon-1/2}{2\varepsilon} + \frac{2\eta}{2\varepsilon}\right) = \zeta\frac{2y+2\varepsilon-1}{2\varepsilon} = \zeta\frac{4s\varepsilon}{2\varepsilon} = 2\zeta s$$
$$Var(Y) = Var(2\alpha) + Var(\gamma) + 2Covar(2\alpha,\gamma) = 4Var(\alpha) + Var(\gamma) - 4(\overline{\alpha\gamma} - \overline{\alpha}\,\overline{\gamma})$$

We have the values (see, for example, pag. 639 of [51]): $\overline{\alpha} = \zeta p$, $\overline{\gamma} = \zeta r$, $Var(\alpha) = \zeta p(1-p)$ and $Var(\gamma) = \zeta r(1-r)$. The product expectation is given by

$$\begin{aligned}\overline{\alpha\gamma} &= \sum_{x,y} xyP(\alpha = x, \gamma = y) = \sum_{x,y} xyP(\alpha = x|\gamma = y)P(\gamma = y)\\ &= \sum_y y\sum_x xP(\alpha = x|\gamma = y)P(\gamma = y) = \sum_y yE(\alpha|\gamma = y)P(\gamma = y)\\ &= E(\gamma\cdot E(\alpha|\gamma)) = E(\gamma)\cdot E(\alpha|\gamma) = \zeta(\zeta-1)rp\end{aligned}$$

whence the variation is:

$$\begin{aligned}Var(Y) &= 4\zeta p(1-p) + \zeta r(1-r) - 4(\zeta(\zeta-1)pr - \zeta^2 pr) = \zeta(4pq + r(1-r))\\ &= \zeta\left(4\frac{(y-\eta+\varepsilon-1/2)(1/2+\varepsilon-y-\eta)}{4\varepsilon^2} + \frac{\eta\varepsilon-\eta^2}{\varepsilon^2}\right) = \zeta\left(\frac{-y^2-\eta\varepsilon+\varepsilon^2+y-1/4}{\varepsilon^2}\right)\\ &= \zeta\left(\frac{4s\varepsilon-4s^2\varepsilon-\eta}{\varepsilon}\right) = \zeta\left(\frac{4\varepsilon(1-s)s-\eta}{\varepsilon}\right) \le 4\zeta\left(\frac{\varepsilon(1-s)s}{\varepsilon}\right) \le {}^{8}4\zeta\end{aligned}$$

Therefore,

$$\overline{X} = \frac{1}{2\zeta}\overline{Y} = s; \quad Var(X) = \frac{Var(Y)}{4\zeta^2} \le \frac{4\zeta}{4\zeta^2} = \frac{1}{\zeta}$$

Let the number of firings be $\zeta = 2^{2l+h}$. Using Chebyshev's inequality, with $t = 2^{-l}/Var(X)$, we have:

$$P(|X-s| > 1/2^l) \le 2^{2l}Var(X) \le \frac{2^{2l}}{\zeta} = 2^{-h} = \delta$$

This inequality gives an upper bound on the probability of reading s to an accuracy of 2^ℓ *incorrectly*. Then, with $\zeta = 2^{2l+h}$ experiments, X will approximate s to an accuracy of 2^l with an error smaller than $\delta = 2^{-h}$. The proof is completed by remarking that Algorithm 2.2 calculates the value of X. □

[8] Recall that $s \in (0,1)$, whence $s(1-s) \le 1$.

2.2.3 Computational power of the SmSM

In this section we will characterize the computational power of the (error-free and error-prone) SmSM, when clocked in polynomial time and with an exponential schedule. We will show how to use the wedge vertex position as an advice function, introduce the concept of a boundary number and present a method to use the uncertainty from the error-prone protocols to simulate the tossing of a fair coin. Finally, we will give a characterization of the class of sets that can be decided using this models of computation.

2.2.3.A Encoding a function into the wedge vertex position

In this section we will show how to encode a prefix advice function f into the position of the wedge vertex of the SmSM. Hence, the problem of using the advice given by f, in the sense of Section 2.1.2, will be reduced to that of reading digits of the position of the wedge vertex.

Definition 2.17. The set of Cantor numbers, denoted by $\mathcal{C}_3$, is the set of real numbers x such that $x = \sum_{k=1}^{\infty} x_k 2^{-3k}$, where $x_k \in \{1, 2, 4\}$, i.e. the numbers composed of the triples 001, 010, or 100.

The advantage of using this type of numbers is their relation with the set of dyadic rationals, which we make evident in the following proposition. Its statement appears for the first time in [13]. We present a shorter proof of the statement, by considering only the worst case for the expansion of y.

Proposition 2.18. For every $y \in \mathcal{C}_3$ and for every dyadic rational $z \in [0, 1]$, with size n, if $|y - z| \leq 2^{-l-5}$, for $l \leq n$, the binary expansion of y and z coincide in the first l bits.

Proof. Suppose that y and z differ in the lth bit and that that $z < y$, meaning that $y_l = 1$ and $z_l = 0$. Then $|y - z|$ is the smallest if y has the biggest possible amount of zeros after l and the digits of z continue until n with 1's. Since y is composed of the blocks 100, 010 and 001, and $y_l = 1$, the smallest difference is attained when $l \equiv_3 1$ and the digits after y_l are $00001\ldots$. Thus, we have two possibilities, depending on whether the carry in the subtraction up to $l + 5$ is 1 or 0. If the carry is 1, we have:

$$|y - z| = 2^{-l}|1.00001\cdots - 0.11111\cdots| = 2^{-l}|0.00001\cdots| > 2^{-l}|0.00001| = 2^{-l-5}$$

If the carry is 0, we have:

$$|y - z| = 2^{-l}|1.00001\cdots - 0.11111\cdots| = 2^{-l}|0.00010\cdots| > 2^{-l-4} > 2^{-l-5}$$

Analogously, if $y < z$ the smallest distance is attained when z continues until $n = |z|$ with only 0's and y is as big as possible. This happens when $l \equiv_3 2$, with y_{l+1} at the end of a 001 block, and the next digits

after y_{l+1} are 100. In any case, we will have:

$$|y-z| = |z-y| = 2^{-l}|1.000\cdots - 0.1100\cdots| = 2^{-l}|0.01\cdots| > 2^{-l}|0.01| = 2^{-l-2} > 2^{-l-5}$$

So, in any case, if y and z aren't equal in the first l digits, we get that $|y-z| > 2^{-l-5}$. □

Definition 2.19. Given a prefix advice function $f : \mathbb{N} \to \{0,1\}^\star$, consider a sequence s_n that satisfies $f(n+1) = f(n)s_{n+1}$, for each $n \in \mathbb{N}$. We then set the position of the wedge vertex $y(f)$ to the limit of the sequence $y_f(n)$, recursively defined as follows:

$$y_f(0) = 0 \cdot c(f(0))$$
$$y_f(n+1) = \begin{cases} y_f(n)c(s_{n+1}) & n+1 \text{ is not a power of 2} \\ y_f(n)c(s_{n+1})001 & n+1 \text{ is a power of 2} \end{cases}$$

where, for $s \in \Sigma^\star$, $c(s)$ replaces each 0 and 1 in s with 100 and 010, respectively. The terms 001 work as separation markers, which we only place after a power of 2.

For example, $y_f(5) = 0 \cdot c(f(0))c(s_1)001c(s_2)001c(s_3)c(s_4)001c(s_5)$. The reason for only marking the powers of 2 is made evident in the following proposition.

Proposition 2.20. Consider a prefix advice function $f \in \texttt{log}^\star$. Then, to use the advice function on a word w, we only need to read a number of digits from $y(f)$ which is logarithmic in the size of w.

Proof. Consider $m = \lceil \log |w| \rceil$ and read the binary expansion of $y(f)$ until $m+1$ 001's have been read. Then, removing the extra 001, we can reconstruct $f(2^m)$ which, by Definition 2.8, can be used as an advice for w.

Now, since $f \in \texttt{log}^\star$, there are constants L, K such that $|c(f(2^m))| \leq L\log(2^m) + K = Lm + K$. Therefore, we have to read at most $Lm + K + 3m = (L+3)m + K$ digits of $y(f)$, when we add in the separators. Then, since m is logarithmic in the size of w, so is the number of digits we have to read. □

Thus, given a word w of size n, we will read the digits of $y(f)$ to obtain $f(2^{\lceil \log n \rceil})$, which we will use as advice for the computations on w.

2.2.3.B Boundary numbers

To find the upper bounds of the computational power of the SmSM, we will create a prefix advice function which will enable a regular Turing machine to simulate the answer given by the SmSM. This advice function will be built using boundary numbers, which, for each query size, give enough information to conclude the result of an experimental call, without having to run the actual experiment.

Definition 2.21. Consider a SmSM with vertex position $y \in (0,1)$ and an exponential time schedule T. For every $z \in \{0,1\}^\star$ of size k, we define the left and right boundary numbers, l_k and r_k, as the two real numbers in $(0,1)$ that satisfy the equation $t(l_k) = t(r_k) = T(k)$, with $l_k < y < r_k$.

Since the curve of the SmSM is symmetric, the boundary numbers are equidistant from the wedge vertex, i.e., $|y - l_k| = |y - r_k|$. Note that we are not defining the boundary numbers as being possible queries for the SmSM; in fact, given that they can take any real value in $(0,1)$, this might even be impossible. Instead, we are saying that, if a particle were to be shot at a boundary number, l_k or r_k, the experiment would take time $T(k)$ to return an answer. Also, note that we can only guarantee the existence of l_k and r_k if $T(k)$ is greater than the time it takes for an experiment to be run with the cannon aiming at 0 and 1. If we take an increasing schedule, this will happen eventually, so we will assume that it is always the case, as only a finite number of values violates the assumption.

Now suppose that we want to run an experiment with a query z. We then 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 at Algorithm 2.3, where we replace oracle consultations by a comparison of the query word with both $l_{|z|}$ and $r_{|z|}$. In reality, we will not use the boundary numbers themselves, but dyadic approximations.

Algorithm 2.3: 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;
if $z \geq r_{|z|}$ **then**
  A transition to the state q_r occurs;
if $l_{|z|} < z < r_{|z|}$ **then**
  A transition to the state q_t occurs;

Recall that, when the cannon is aimed at $z \in (0,1)$, the error-prone protocols choose its actual position, uniformly, in the interval $[z - \varepsilon, z + \varepsilon]$, where, for either protocol, ε is the error associated with the firing. We then have the six cases depicted in Figure 2.5.

We are assuming that the cannon has a restriction, so that it only fires in the interval $(0,1)$. This is because when a dyadic rational $z \in (0,1)$ is chosen, close enough to either 0 or 1, the interval $[z-\varepsilon, z+\varepsilon]$ might no longer be contained in $(0,1)$. We will thus use the convention that, if the position z', chosen by an error-prone protocol, is smaller (resp. greater) than 0 (resp. 1), the experiment immediately returns q_l (resp. q_r). For presentation purposes, the shooting cases from Figure 2.5 do not contemplate this option, of the shooting interval exceeding $(0,1)$.

Proposition 2.22. Suppose that a SmSM fires a particle query of size k under protocol Prot_UP(z). Then, knowing $k + d$ bits of l_k and r_k is enough to approximate the probabilities of having q_l, q_r or q_t, within an error less than 2^{-d}.

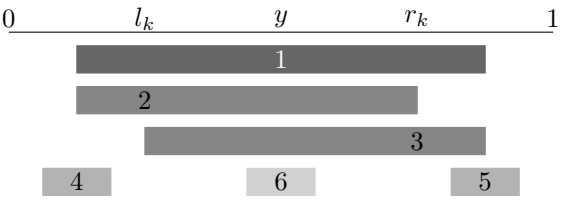

Figure 2.5: Shooting cases.

Proof. Protocol Prot_UP(z) chooses uniformly some $z' \in [z - 2^{-k}, z + 2^{-k}]$, which originates the six shooting cases from Figure 2.5. In each case, we can calculate the probability of each possible outcomes in terms of boundary numbers. This representation determines how many digits of the boundary number we must know, in order to predict the outcome of an experimental call with a given precision.

1. $z - 2^{-k} \leq l_k < y < r_k \leq z + 2^{-k}$:

$$P(q_l) = P(z' \in [z - 2^{-k}, l_k]) = \frac{l_k - z + 2^{-k}}{z + 2^{-k} - (z - 2^{-k})} = \frac{l_k - z}{2 \times 2^{-k}} + \frac{1}{2} = 2^{k-1}(l_k - z) + \frac{1}{2}$$
$$P(q_t) = P(z' \in (l_k, r_k)) = \frac{r_k - l_k}{z + 2^{-k} - (z - 2^{-k})} = \frac{r_k - l_k}{2 \times 2^{-k}} = 2^{k-1}(r_k - l_k)$$
$$P(q_r) = P(z' \in [r_k, z + 2^{-k}]) = \frac{z + 2^{-k} - r_k}{z + 2^{-k} - (z - 2^{-k})} = \frac{z - r_k}{2 \times 2^{-k}} + \frac{1}{2} = 2^{k-1}(z - r_k) + \frac{1}{2}$$

2. $z - 2^{-k} \leq l_k \leq z + 2^{-k} \leq r_k$:

$$P(q_l) = 2^{k-1}(l_k - z) + 1/2; \quad P(q_t) = 2^{k-1}(z - l_k) + 1/2; \quad P(q_r) = 0$$

3. $l_k \leq z - 2^{-k} \leq r_k \leq z + 2^{-k}$:

$$P(q_l) = 0; \quad P(q_t) = 2^{k-1}(r_k - z) + 1/2; \quad P(q_r) = 2^{k-1}(z - r_k) + 1/2$$

4. $z - 2^{-k} < z + 2^{-k} \leq l_k$:

$$P(q_l) = 1; \quad P(q_t) = 0; \quad P(q_r) = 0$$

5. $r_k \leq z - 2^{-k} < z + 2^{-k}$:

$$P(q_l) = 0; \quad P(q_t) = 0; \quad P(q_r) = 1$$

6. $l_k \leq z - 2^{-k} < z + 2^{-k} \leq r_k$:

$$P(q_l) = 0; \quad P(q_t) = 1; \quad P(q_r) = 0$$

Take, for example, the expression $P(q_l) = 2^{k-1}(l_k - z) + 1/2$ and let $a = l_k - z$ and $b = 2^{k-1}$. Note that, for a precision of 2^{-k-d}, there is no error associated with the approximations of z, 2^{k-1}, or $1/2$. Then, by Proposition A.7 from the Appendix, we have:

$$|\Delta P(q_l)| \leq a|\Delta b| + b|\Delta a| + |\Delta a||\Delta b| = b|\Delta a| \leq 2^{k-1}(|\Delta l_k| + |\Delta z|) = 2^{k-1}|\Delta l_k|$$

Thus, if we know $k + d$ bits of l_k and r_k, we can approximate the $P(q_l)$ with a precision of 2^{-d}.[9] The other cases are analogous.

Note that, even though a comparison between $z \pm 2^{-k}$ and a finite number of bits of l_k and r_k is not enough to uniquely determine which case is valid, this does not affect the validity of the predictions. Suppose, for example, that $z - 2^{-k} < l_k\downarrow_{k+d}$ and that $z + 2^{-k} = r_k\downarrow_{k+d}$, where we assume that each dyadic rational has been padded to have a precision of $k + d$. Then, we can either have $z + 2^{-k} < r_k$ or $z + 2^{-k} = r_k$, depending on whether or not the rest of the expansion of r_k has only zeros, which means that both cases 1 and 2 are possible. We see that, either way, we have the following approximations:

$$P(q_l)_1 = 2^{k-1}\left(l_k\downarrow_{k+d} - z\right) + 1/2 = P(q_l)_2$$
$$P(q_t)_1 = 2^{k-1}\left(r_k\downarrow_{k+d} - l_k\downarrow_{k+d}\right) = 2^{k-1}\left(z + 2^{-k} - l_k\downarrow_{k+d}\right) = 2^{k-1}\left(z - l_k\downarrow_{k+d}\right) + 1/2 = P(q_t)_2$$
$$P(q_r)_1 = 2^{k-1}\left(z - r_k\downarrow_{k+d}\right) + 1/2 = 2^{k-1}\left(z + 2^{-k} - r_k\downarrow_{k+d}\right) = 0 = P(q_r)_2$$

Where, for $s = q_l, q_t, q_r$, $P(s)_i$ is the approximation of $P(s)$ using the expression from case i. □

Proposition 2.23. Suppose that a SmSM fires a particle query of size k under protocol Prot_FP(z), with precision $\varepsilon = 2^{-q}$. Then, knowing $q + d$ bits of l_k and r_k is enough to approximate the probabilities of having q_l, q_r or q_t, within an error less than 2^{-d}.

Proof. The proof is analogous to that of Proposition 2.22. Protocol Prot_FP(z) chooses uniformly some position $z' \in [z - 2^{-q}, z + 2^{-q}]$, which originates the six possible shooting cases represented in figure 2.5. Then, we can determine the probability of having q_l, q_r, or q_t in terms of the boundary numbers. This allows us to determine how many digits of the boundary numbers we must know, in order to predict the outcome with a given precision. Therefore, knowing $q + d$ bits of l_k and r_k allows us to approximate the probabilities within an error less than 2^{-d}. □

We will now prove that, if an assumption is made on the physical duration of an experimental call, we can calculate approximations of the boundary numbers using the position of the wedge vertex.

[9] In reality, we only need to know $k - 1 + d$ bits of the boundary number to approximate the probabilities to a precision of 2^{-d}. We consider $k + d$ to simplify the calculations.

Proposition 2.24. If we assume that the time of the experiment is of the form[10] $t(z) = 1/|y - z|$, then, if we have $d + 1$ bits of y, we can compute the boundary numbers with an error less than 2^{-d}.

Proof. Suppose that the SmSM writes a query z of size k. Then, considering the explicit time, as the boundary numbers satisfy the properties $l_k < y < r_k$ and $t(l_k) = t(r_k) = T(k)$, we have:

$$\frac{1}{|y - l_k|} = T(k) \;\Leftrightarrow\; y - l_k = \frac{1}{T(k)} \;\Leftrightarrow\; l_k = y - \frac{1}{T(k)}$$

Analogously, $r_k = y + 1/T(k)$.

Thus, given approximations to y, we can obtain approximations of l_k and r_k. Since the schedule T is internal to the Turing machine, the only error associated with $1/T(k)$ comes from the truncation we make to match the precision of y, a truncation in which we might discard non-zero digits.

Suppose we have $d + 1$ bits of y. Then, by computing $d + 1$ bits of $1/T(k)$, the error associated with l_k or r_k is bounded by $2^{-d-1} + 2^{-d-1} = 2^{-d}$ (see Section A.3). Therefore, knowing $d + 1$ bits of y allows us to compute the boundary numbers with an error less than 2^{-d}.[11] □

2.2.3.C The error-prone SmSM as a biased coin

To calculate the lower bounds for the computational power of the error-prone SmSM, we will use the fact that the randomness in the firing of the cannon, given by the error associated with the error-prone protocols, can be used to simulate the tossing of a fair coin, which the SmSM will be able to use to simulate random algorithms. In this section we prove that, with the bounded and unbounded precision protocols, the SmSM can be used to simulate the tossing of a biased coin, which, in turn, can be used to simulate the tossing of a fair one, up to a given probability of error (see Section A.2).

Since these results will regard the lower bounds for the computational power of the SmSM, we will make the assumption that the machine has been setup a non-dyadic y and has a schedule of the form $T(k) = D2^k$, for $D > C$. The first restriction is to ensure that we will not fire at y; the reason for the second will be made evident in the following proposition.

Proposition 2.25. Consider an error-prone SmSM with *unbounded* precision and with an exponential schedule $T(k) = D2^k$, for $D > C$. Then, the size of the interval $[l_k, r_k]$ is smaller than the size of the shooting interval $[z - 2^{-k}, z + 2^{-k}]$, for any dyadic rational z of size k.

Proof. For any dyadic rational z of size k, the size of the shooting interval is $z + 2^{-k} - (z - 2^{-k}) = 2^{-k+1}$.

[10] We can see that the result is still valid if the experimental time is of the form $t(z) = D/|y - z|^n$, for a constant $D > 0$ and a degree $n > 0$. We would just have to calculate $d + 1$ digits of y and $\sqrt[n]{D/T(k)}$.

[11] Note that, since $T(k)$ is polynomial in the size of the input, so is the time taken to perform this calculation.

Now, consider the boundary number l_k. From inequality 2.3, we have:

$$t(l_k) \leq \frac{C}{|y - l_k|} \Leftrightarrow |y - l_k| \leq \frac{C}{D} 2^{-k}$$

Then, since the left and right boundary numbers are equidistant, we have that $r_k - l_k \leq C/D2^{-k+1}$, whence the interval $[l_k, r_k]$ shrinks faster than the shooting interval. □

The following statement appears for the first time in [11]. We will prove it by making use of the probabilities calculated in Proposition 2.22.

Proposition 2.26. Given an error-prone SmSM with *unbounded* precision and an exponential schedule $T(k) = D2^k$, for $D > C$, there is a dyadic rational z, such that the outcome of Prot_UP on z is a random variable that produces q_l with probability $\delta = 2^{k-1}(l_k - z) + 1/2$.

Proof. Consider an error-prone SmSM in the statement's conditions and let z be a query of size k such that $z - 2^{-k} > 0$ and $y \in (z - 2^{-k}, z + 2^{-k})$. Then, since the interval $[l_k, r_k]$ shrinks faster than $[z - 2^{-k}, z + 2^{-k}]$, it must be contained in it, whence we will have the first shooting case from Figure 2.5. Then, by Proposition 2.22, the probability of Prot_UP(z) returning q_l is δ. □

Proposition 2.27. Given an error-prone SmSM with *fixed* precision $\varepsilon = 2^{-q}$ and an exponential schedule $T(k) = D2^k$, for $D > C$, there is a dyadic rational z, such that the outcome of Prot_FP on z is a random variable that produces q_l with probability $\delta = 2^{q-1}(l_k - z) + 1/2$.

Proof. The proof is analogous to that of Proposition 2.26. Consider a dyadic query z of size k, such that $2^{-k} < \varepsilon$, $y \in (z - 2^{-k}, z + 2^{-k})$ and $z - 2^{-k} > 0$. Then, the probability of Prot_FP(z) returning q_l is δ. □

Therefore, when working with the error-prone protocols, we can use the SmSM as the tossing of a biased coin. Given Proposition A.6 from the Appendix, we know that this is enough to simulate the tossing of a fair one, up to a given probability of failure.

2.2.3.D Computational power of the error-free SmSM

In this section we will characterize the class of sets that can be decided by an error-free SmSM, clocked in polynomial time and with an exponential schedule.

Proposition 2.28. If $B \in \mathtt{P/log}\star$, then there is an error-free SmSM, clocked in polynomial time and with an exponential schedule, that can decide B.

Proof. Let $B \in \mathtt{P/log}\star$. Then, there is a prefix advice function $f \in \log$ and a Turing machine M, clocked in polynomial time, that can decide B using the advice f. Recall that, since f is logarithmic, there are constants a and b such that, for each $n \in \mathbb{N}$, $f(n) = a\lceil \log(n) \rceil + b$. Therefore, given the encoding

described in Definition 2.19 and the process described in Proposition 2.20, we have to calculate $l = 3(a\lceil\log(n)\rceil + b) + 3(\lceil\log(n)\rceil + 1)$ digits of $y(f)$ to read $f(n)$, where n is a power of 2.

Consider an error-free SmSM M with the wedge vertex at $y(f)$ and schedule $T(l) = D2^l$, for $D > C$. We can then use Algorithm 2.1, with input $l + 5$, whose outcome, by Proposition 2.15, will be a dyadic rational m with $|y(f) - m| < 2^{-l-5}$. Then, given Proposition 2.18, $y(f)$ and m will coincide in the first l bits, whence we can recover $f(n)$ from m.

Since the precision to run the experiment is logarithmic in the size of the input, the time taken to obtain the required digits from the wedge vertex will be polynomial, as desired. □

Proposition 2.29. If a set B is decided by an error-free SmSM, clocked in polynomial time and with an exponential schedule, then $B \in \mathtt{P/log^2\star}$.

Proof. Suppose that B is decidable by an error-free SmSM M, clocked in polynomial time and with an exponential schedule T. Since T is exponential and 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, M only queries the oracle with words of size less or equal to $l = a\lceil\log(n)\rceil + b$.[12] Given this, we consider a prefix advice function f such that $f(n)$ encodes the concatenation of boundary numbers needed to answer all the queries of size l. Thus $f(n)$ encodes

$$l_1\downarrow_1 \# r_1\downarrow_1 \# l_2\downarrow_2 \# r_2\downarrow_2 \# \ldots \# l_m\downarrow_m \# r_m\downarrow_m \#$$

The prefix advice function f is such that $|f(n)| = 2l + 2\sum_{i=1}^{l} i \in O(l^2) = O(\log^2(n))$. Therefore, to decide B in polynomial time, with the prefix advice $f \in \log^2$, we simulate M on the input word and use Algorithm 2.3 to replace the calls to $Prot\_IP(z)$ by a comparison with the part of the boundary numbers given by $f(n)$. Thus, given the advice function f, we can decide B in polynomial time, whence $B \in \mathtt{P/log^2\star}$. □

We will now prove that we can fully characterize the computational power of the error-free SmSM, if we *assume* that it has a schedule satisfying $T(k) \in \Omega(2^k)$.[13] The idea is that, under this assumption, successive boundary numbers will be close enough together that we can use r_k and l_k to obtain r_{k+1} and l_{k+1}, using just a constant amount of additional information.

Proposition 2.30. Given a SmSM with time schedule $T(k) \in \Omega(2^k)$, it is possible to define a linear prefix advice function f, such that $f(n)$ encodes all the boundary numbers with size up to n.

[12] This is why we consider a different form for the size of a logarithmic advice (recall the comment at the end of Section 2.1.2).

[13] If, instead of working with the time given by inequality 2.3, we kept the degree n in the expression for the duration of an experiment, the restriction on the schedule would be $T(k) \in \Omega(2^{kn})$.

Proof. If the time schedule associated with a SmSM is $T(k) \in \Omega(2^k)$, there are constants $D, k_0 \in \mathbb{N}$ such that, for all $k \geq k_0, T(k) \geq D2^k$. Let $c \in \mathbb{N}$ be such that $2^c > C/D$. Then, given the time from inequality 2.3 and the definition of boundary numbers, we have that, for each $k \geq k_0$:

$$t(r_k) \leq \frac{C}{|y - r_k|} \Rightarrow T(k) \leq \frac{C}{|y - r_k|} \Rightarrow D2^k \leq \frac{C}{|y - r_k|} \Rightarrow |y - r_k| \leq \frac{C}{D}2^{-k} < 2^{-k+c}$$
$$t(l_k) \leq \frac{C}{|y - l_k|} \Rightarrow |y - l_k| \leq \frac{C}{D}2^{-k} < 2^{-k+c}$$

We will consider truncations of the boundary numbers as the concatenation of two words, i.e., $r_k\downarrow_k = x_k z_k$ and $l_k\downarrow_k = v_k w_k$, where x_k and v_k have size $k - c$, and z_k and w_k have size c. Note that, since r_k cannot have a tail composed solely of 1's, $r_k - 2^{-k+c}$ is smaller than x_k. Therefore, we have that $y - 2^{-k+c} < r_k - 2^{-k+c} < x_k \leq r_k < y + 2^{-k+c}$, whence $|y - x_k| < 2^{-k+c}$ and $|y - v_k| < 2^{-k+c}$.

Now, we will show how to obtain x_{k+1} and v_{k+1} from x_k and v_k, respectively. In the following paragraph, assume l to be a positive integer smaller than $k - c$. If $x_k = \ldots 10^l$, then, as $|y - x_k| < 2^{-k+c}$, $y\downarrow_{k-c}$ can either be $\ldots 10^l$ or $\ldots 01^l$. In this case, since $|y - x_{k+1}| < 2^{-k+c-1}$, x_{k+1} must end in one of the following: $x_{k+1} = \ldots 10^l 1$, $x_{k+1} = \ldots 10^l 0$, $x_{k+1} = \ldots 01^l 1$, or $x_{k+1} = \ldots 01^l 0$. Similarly, if $x_k = \ldots 01^l$, then $y_k\downarrow_{k-c}$ can either be $\ldots 01^l$ or $\ldots 01^{l-1}0$ and x_{k+1} can be one of four possibilities as well. Therefore, even though x_k might not be a prefix of x_{k+1}, the latter can be obtained from x_k by using two bits to determine which of the four cases is the correct one. The result is analogous for the case of v_k and v_{k+1}.

Given this fact, we define the advice function inductively as follows:

- If $n < k_0$, then $f(n) = l_1\downarrow_1 \# r_1\downarrow_1 \ldots l_n\downarrow_n \# r_n\downarrow_n \#$;
- if $n = k_0$, then $f(k_0) = f(k_0 - 1)\#\#x_{k_0}\#z_{k_0}\#v_{k_0}\#w_{k_0}\#$;
- if $n > k_0$, then $f(n) = f(n - 1)\#b_{11}b_{12}\#z_n\#b_{21}b_{22}\#w_n\#$, where b_{ij} are the bits that allow us to obtain x_n and v_n from x_{n-1} and v_{n-1}, respectively.

Therefore, we can always recover r_k or l_k from $f(n)$, for any $k < n$. Before the double marker, the values are written explicitly in $f(n)$; if $n = k_0$, then, by concatenating x_{k_0} with z_{k_0} and v_{k_0} with w_{k_0}, we get the desired values; if $n > k_0$, we can obtain l_n and r_n by concatenating the two bits from the sequence to then z_n or w_n, respectively.

Finally, since z_k and w_k have a constant size, $|f(n)|$ is asymptotically linear in n. □

Proposition 2.31. If a set B is decided by an error-free SmSM, clocked in polynomial time and with an exponential schedule $T(k) \in \Omega(2^k)$, then $B \in \mathtt{P/log}\star$.

Proof. The proof is analogous to that of Proposition 2.29, but using the advice described in Proposition 2.30. Since this function is linear in the size of the queries, and therefore logarithmic in the size of the input, we achieve the desired result. □

Thus, we have proved the following result.

Proposition 2.32. A set B is decided by an error-free SmSM, clocked in polynomial time and with an exponential schedule $T(k) \in \Omega(2^k)$, if and only if $B \in \mathtt{P}/\mathtt{log}\star$.

It is an open problem to know if in the above proposition holds without the schedule restriction.

2.2.3.E Computational power of the error-prone SmSM with unbounded precision

In this section we will characterize the class of sets that can be decided by an error-prone SmSM, clocked in polynomial time, with unbounded precision and with an exponential schedule.

Proposition 2.33. If $B \in \mathtt{BPP}//\mathtt{log}\star$, then there exists an error-prone SmSM, clocked in polynomial time, with unbounded precision and an exponential schedule, that decides B.

Proof. Let $B \in \mathtt{BPP}//\mathtt{log}\star$. Then, there is a prefix advice function $f \in \log$, a probabilistic Turing machine M, clocked in polynomial time p, and a constant $\gamma_1 < 1/2$ such that, for every word w of size $\leq n$, the probability of M making a mistake is smaller than γ_1. We need to prove two things: that the SmSM can be used to simulate the advice function f and the tossing of a fair coin; this simulation can be done in polynomial time in the size of the input, with an error bounded by a constant smaller than $1/2$.

Consider a SmSM with an exponential time schedule $T(k) = D2^k$, for $D > C$, and wedge vertex at $y(f)$, as described in Definition 2.19, and, once more, consider $l = 3(a\lceil\log(n)\rceil + b) + 3(\lceil\log(n)\rceil + 1)$, which we choose due to the use of the markers and the function c in the encoding of $y(f)$. Then, by Propositions 2.15 and 2.18, if we use Algorithm 2.1 with input $l+6$, the outcome will be a dyadic rational m which will will coincide with $y(f)$ in the first l bits.

Now, by Propositions 2.26 and A.6, there is a dyadic rational z that can be used to produce a sequence of fair coin tosses of size $p(n)$ with a probability of failure of γ_2, which we choose to be such that $\gamma_1 + \gamma_2 < 1/2$. Thus, after using the SmSM as an oracle to compute $f(2^{\lceil\log(n)\rceil})$ (recall Proposition 2.20), the SmSM uses the vertex to generate a sequence of fair coin tosses of size $p(n)$, which it uses to guide its computation on $\langle w, f(2^{\lceil\log(n)\rceil})\rangle$. We will prove that this process has an associated error probability bounded by $\gamma_1 + (1-\gamma_1)\gamma_2 \leq \gamma_1 + \gamma_2 < 1/2$.

We can separate the computation done by the SmSM into three steps: The part where we find l digits of $y(f)$, which has no associated error, the part where we do the coin tossing, with an error of γ_2, and the simulation of M, which has an associated error of γ_1. From the possible coin tosses done in step 2, a fraction γ_2 of these will have an error, and the rest will actually be fair coin tosses. Of the computations that follow from the fair coin tosses, a fraction γ_1 will have an error, and, in the worst case, all the computations that follow from the biased coin tosses will have an error as well. Then, the overall fraction of wrong computations is bounded by $\gamma_1 + (1-\gamma_1)\gamma_2 < \gamma_1 + \gamma_2 < 1/2$.

To conclude, note that, by Proposition 2.15, running the experiment takes time $O(lT(l))$, which, since l is logarithmic in the size of the input, amounts to a polynomial time. Encoding $\langle w, f(2^{\lceil \log(n) \rceil}) \rangle$ also takes polynomial time and so does the generation of the fair coin tosses. Therefore, the time for the whole computation is polynomial in the size of the input. □

Now, to prove the upper bound, we will construct an advice function that provides the Turing machine with approximations of the left and right boundary numbers, thus enabling it to simulate the outcome of an experiment. To know how many digits from the boundary numbers are necessary, for the experiment to be simulated up to a given precision, we will use Proposition 2.22.

Proposition 2.34. If B is decidable by an error-prone SmSM, clocked in polynomial time, with unbounded precision and an exponential schedule, then $B \in \mathtt{BPP}//\mathtt{log}^{2}\star$.

Proof. Suppose that B is decidable by a SmSM M, with unbounded precision, in polynomial time $O(n^a)$. Since M on w takes polynomial time in $n = |w|$, and each experiment takes an exponential time to run, the size of each oracle query must be bounded above by $b\lceil \log(n) \rceil + c$, for some constants b and c. The number of queries to the oracle cannot exceed the running time of the machine, so the probabilistic query tree of M has a *depth* of at most αn^a, for some constant α. Let γ be the error probability of M and let $d > 0$ be such that $2^d > 2\alpha/(1/2 - \gamma)$.

We will simulate the behaviour of M using Algorithm 2.3, in which we replace an experimental call with a comparison between the query and a truncation of the boundary numbers.

By Proposition 2.22, if we use $k + d + a\lceil \log(n) \rceil$ bits of l_k and r_k, we can approximate the probabilities of all queries with precision $2^{-d-a\lceil \log(n) \rceil}$, for every $1 \leq k \leq b\lceil \log(n) \rceil + c$. Now, we use Proposition A.5 of Section A.1 to see that, with a precision of $2^{-d-a\lceil \log(n) \rceil}$, the simulation we make of M has an accepting probability differing from M's in, at most, $1/2 - \gamma$.[14] We then have:

$$\begin{aligned}\mathcal{A}_3(\alpha n^a, 2^{-d-a\lceil \log(n) \rceil}) &\leq (3-1) \times \alpha n^a \times 2^{-d-a\lceil \log(n) \rceil} = \frac{2\alpha n^a}{2^d \times 2^{a\lceil \log(n) \rceil}} \\ &\leq \frac{2\alpha n^a}{2^d \times 2^{\log(n^a)}} = \frac{2\alpha n^a}{2^d \times n^a} = \frac{2\alpha}{2^d} < (1/2 - \gamma)\end{aligned}$$

Considering $\beta = \max\{a, b\}$ and $x = \lceil \log(n) \rceil$, we have:

$$\begin{aligned}&b\lceil \log(n) \rceil + c \leq \beta x + c \\ &k + d + a\lceil \log(n) \rceil \leq b\lceil \log(n) \rceil + c + d + a\lceil \log(n) \rceil \leq d + c + 2\beta x\end{aligned}$$

Using the notation $l\rfloor_n^m$ to mean the digits n to m of l, and $l_1 \cdot l_2$ to denote the concatenation of l_1 and l_2, we recursively define the following prefix advice function, which encodes approximations to the boundary

[14] As in Definition A.4, $\mathcal{A}_m(n, s)$ denotes the maximum difference of acceptance probability amongst m-ary trees with depth n, whose branches have probabilities differing in, at most, s. In this case, we have 3-ary query trees.

numbers l_k and r_k, for $1 \leq k \leq \beta x + c$, with a precision of $d + c + 2\beta x$ digits:

- $f(0) = l_1\downarrow_{d+c}\#r_1\downarrow_{d+c}\#l_2\downarrow_{d+c}\#r_2\downarrow_{d+c}\#\ldots\#l_c\downarrow_{d+c}\#r_c\downarrow_{d+c}$
- $f(x+1) = f(x)\cdot\left\{l_k\downarrow_{d+c+2\beta x+1}^{d+c+2\beta(x+1)}\#r_k\downarrow_{d+c+2\beta x+1}^{d+c+2\beta(x+1)}\right\}_{k=1}^{\beta x+c}\cdot\left\{l_k\downarrow_{d+c+2\beta(x+1)}\#r_k\downarrow_{d+c+2\beta(x+1)}\right\}_{k=\beta x+c+1}^{\beta(x+1)+c}$

So $f(x+1)$ contains:

- The value of $f(x)$;
- $2(\beta x + c)$ blocks of size 2β, with approximations of l_k and r_k, for $k = 1, \ldots, \beta x + c$;
- 2β blocks of size $d+c+2\beta(x+1)$, with approximations of l_k and r_k, for $k = \beta x+c+1, \ldots, \beta x+c+\beta$.

This way, each new value of f increases the accuracy of the boundary numbers that have been previously written and adds part of new boundary numbers. Notice that, at $f(x)$, we will have approximations of r_k and l_k, for $k \leq \beta(x-1) + c$, with a precision of $d + c + 2\beta x$. For example:

$$\begin{aligned} f(1) = {} & l_1\downarrow_{d+c}\#r_1\downarrow_{d+c}\#l_2\downarrow_{d+c}\#r_2\downarrow_{d+c}\#\ldots\#l_c\downarrow_{d+c}\#r_c\downarrow_{d+c} \\ & \#l_1\downarrow_{d+c+1}^{d+c+2\beta}\#r_1\downarrow_{d+c+1}^{d+c+2\beta}\#\cdots\#l_c\downarrow_{d+c+1}^{d+c+2\beta}\#r_c\downarrow_{d+c+1}^{d+c+2\beta} \\ & \#l_{c+1}\downarrow_{d+c+2\beta}\#r_{c+1}\downarrow_{d+c+2\beta}\#\cdots\#l_{c+\beta}\downarrow_{d+c+2\beta}\#r_{c+\beta}\downarrow_{d+c+2\beta} \end{aligned}$$

A Turing machine can read from $f(x)$ the $2(\beta x + c)$ (nonsequential) blocks and update, at the same time, the approximations of l_k and r_k, for $k = 1, \ldots, \beta x + c$, with $d + c + 2\beta x$ digits of precision. Analysing the advice function f, we can conclude that:

$$|f(x)| = 2(\beta(x-1)+c)(d+c+2\beta x) + 2\beta(d+c+2\beta x) + \sum_{i=0}^{x}(2(\beta i + c)) - 1 = O(x^2)$$

Since we only consider x at most logarithmic in the input size, n, we have that $|f(\lceil\log(n)\rceil)| = O(\log^2(n))$. Thus, the advice function $g(n) = f(\lceil\log(n)\rceil)$ provides approximations of l_k and r_k, for $1 \leq k \leq b\lceil\log(n)\rceil + c$, with at least $k + d + a\lceil\log(n)\rceil$ 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 M' that simulates M on the input word, but, whenever M queries the oracle with z, M' compares the query z with approximations of the boundary numbers $l_{|z|}$ and $r_{|z|}$, with a precision of $2^{-d-a\lceil\log(n)\rceil}$. Then, M' simulates a path in the probabilistic query tree, representing the oracle consultation, by tossing a coin $d+a\lceil\log(n)\rceil$ times. After simulating a path, the machine M' proceeds as M. Since the difference in the probability of acceptance is bounded by $1/2 - \gamma$, M' gives a wrong answer with probability less than $\gamma + (1/2 - \gamma) = 1/2$.[15]

Since M runs in polynomial time and comparing query words with boundary numbers and computing

[15]Note that the same argument applies to showing that the error probability of M' is indeed bounded above by some constant $\delta < 1/2$.

probabilities can be done in polynomial time as well, we conclude that B is decidable in polynomial time, in the size of the input, with advice $g(n) = f(\lceil \log(n) \rceil)$. □

We will now prove that, if we make the assumption from Proposition 2.24, regarding the duration of an experiment, we can lower the previous upper-bound to $\texttt{BPP//log}\star$, thus achieving a complete characterization.

Proposition 2.35. If B is decidable by an error-prone SmSM, clocked in polynomial time, with unbounded precision and an exponential schedule, then, considering *explicit time* of the form $t(z) = 1/|y - z|$, we have that $B \in \texttt{BPP//log}\star$.

Proof. We resume to the proof of Proposition 2.34, providing now the advice function that solves the problem is in $\log\star$. All variables and constants are as in the proof of Proposition 2.34.

Consider the error-prone SmSM M, clocked in polynomial time and with an exponential schedule T. By Proposition 2.34, we know that B can be decided by a probabilistic Turing machine, M' in polynomial time, if given access to an advice function f of size $O(\log^2(n))$, 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:

- $g(0) = y\rfloor_{d+c}$
- $g(x+1) = g(x) \cdot y\rfloor_{d+c+2\beta x+1}^{d+c+2\beta(x+1)}$

We can use $g(x)$ to get the first $d + c + 2\beta x$ bits of y, whence, by the Proposition 2.24, 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 $d + c + 2\beta x$ bits of precision. We can see that $|g(x)| = (d + c + 2\beta x) + x = O(x)$, which, since x will be, at most, logarithmic in the size of the input, implies $|g(\lceil \log(n) \rceil)| = O(\lceil \log(n) \rceil)$.

Thus, we define a probabilistic Turing machine M'' that, on input w, simulates 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 M' takes in polynomial time as well, we conclude that our Turing machine can decide B in polynomial time. □

Thus, we have proved the following result.

Proposition 2.36. A set B is decidable by a error-prone SmSM, clocked in polynomial time, with unbounded precision and an exponential schedule, with *explicit time* of the form $t(z) = 1/|y - z|$, if and only if $B \in \texttt{BPP//log}\star$.

It is an unknown if the above proposition holds, without the assumption of the explicit time form.

2.2.3.F Computational power of the error-prone SmSM with fixed precision

In this section we will characterize the class of sets that can be decided by an error-prone SmSM, clocked in polynomial time, with fixed precision and with an exponential schedule. The proof of the

upper bound is analogous to the proof for the unbounded precision case (Propositions 2.34 and 2.35), so we will only give the proof for the lower bound.

Proposition 2.37. If $B \in \mathtt{BPP}//\mathtt{log}\star$, then there exists an error-prone SmSM, clocked in polynomial time, with fixed precision and an exponential schedule, that can decide B.

Proof. The proof is similar to that of Proposition 2.33, but using the measurement algorithm for the fixed precision protocol. The main difference is that now there is an error associated with the measurement.

Let $B \in \mathtt{BPP}//\mathtt{log}\star$. Then, there is a prefix advice function $f \in \log$, a probabilistic Turing machine M, clocked in polynomial time p, and a constant $\gamma_1 < 1/2$ such that, for every word w, of size $\leq n$, the probability of M making a mistake is bounded by γ_1. Consider a SmSM with an exponential time schedule $T(k) = D2^l$, for $D > C$, and wedge vertex at $y = 1/2 - \varepsilon + 2y(f)\varepsilon$, as described in Definition 2.19. The different placing of the wedge is due to proposition 2.16. Consider γ_2 and γ_3 such that $\gamma_1 + \gamma_2 + \gamma_3 < 1/2$. The value γ_2 will be associated with the error in the coin tossing and γ_3 with the error in the measurement algorithm.

Define $l = 3(a\lceil\log(n)\rceil + b) + 3(\lceil\log(n)\rceil + 1)$. Then, by Proposition 2.16, using Algorithm 2.2, with input $l + 5$ and an allowed error of γ_3, will yield a dyadic rational m such that $|y(f) - m| < 2^{-l-5}$, with an error smaller than γ_3. Therefore, by Proposition 2.18, we can use m to compute l bits of $f(2^{\lceil\log(n)\rceil})$ with an error of γ_3. By Propositions 2.27 and A.6, there is a dyadic rational z that can be used to produce a sequence of fair coin tosses of size $p(n)$, with a probability of failure of γ_2. Thus, after using the SmSM as an oracle to compute $f(2^{\lceil\log(n)\rceil})$ with an error of γ_3, the SmSM uses the vertex to simulate a sequence of fair coin tosses of size $p(n)$, which it uses to guide its computation on $\langle w, f(2^{\lceil\log(n)\rceil})\rangle$.

The three steps in the computation have an associated error of γ_1, γ_2 and γ_3, respectively, so, in the worst case, the fraction of wrong computations is $\gamma_1 + (1-\gamma_1)\gamma_2 + (1-\gamma)(1-\gamma_2)\gamma \leq \gamma_1 + \gamma_2 + \gamma_3 < 1/2$.

Running the experiment takes $O(2^{2l+h}T(l))$, by Proposition 2.16, which, since l is logarithmic in the size of the input, amounts to a polynomial time. Encoding $\langle w, f(2^{\lceil\log(n)\rceil})\rangle$ also takes polynomial time and so does the generation of the fair coin tosses, whence the time for the whole computation is polynomial in the size of the input. □

Proposition 2.38. If B is decidable by an error-prone SmSM, clocked in polynomial time, with fixed precision $\varepsilon = 2^{-q}$ and an exponential schedule, then $B \in \mathtt{BPP}//\mathtt{log}^2\star$.

Proposition 2.39. A set B is decidable by an error-prone SmSM, clocked in polynomial time, with fixed precision $\varepsilon = 2^{-q}$ and an exponential schedule, with *explicit time* of the form $t(z) = 1/|y - z|$, if and only if $B \in \mathtt{BPP}//\mathtt{log}\star$.

It is an open problem to know whether or not the above proposition holds without the assumption of the explicit time form.

3

Theory of measurement

In this chapter we will analyse the infinite precision measurement algorithms from a *fundamental measurement* point of view. We will start by introducing the notion of fundamental measurement, as seen by Campbell and Hempel, and present the axiomatization of measurement given by Hempel in [44], together with some immediate results in this theory. We will then present the axiomatization of measurement given in [16], where the duration of an experiment is take into account, and Hempel's notion is recovered as we allow the duration of an experiment to tend to infinity.

We will propose a new axiomatization of measurement, which arises from making comparisons using only dyadic rationals, and prove that, once more, we can recover Hempel's notion as a limit concept. We will see that the three types of infinite precision measurements identified in [20], two-sided, one-sided and vanishing, can be made to satisfy this axiomatization.

3.1 Introduction to the theory of measurement

"Measurement consists in the assignment of numerals to things or properties. But not every assignment of numerals is measurement" (see [27]).

In this section we will introduce the notion of quantities and the possibility of measuring them. We divide measurement into two types: fundamental measurement, which presupposes no prior metrical scales, and derived measurement, which is the determination of a metrical concept by means of others.[1] Properties that can be measured by means of fundamental measurement are termed fundamental magnitudes, as opposed to derived magnitudes.

We divide quantities into extensive, such as mass or length, and intensive, such as temperature or density. Extensive quantities are those whose measure is proportional to the amount of substance present, under a certain way of combining objects. In 1951 Suppes gave an axiomatization for these quantities, which we present in Section A.5, from the Appendix.

We can also classify a quantity as to being additive or non-additive. A quantity is termed additive if the measured of combined objects is the sum of their respective measure; it is termed non-additive if that is not the case.

Campbell and Jeffreys argue that the only properties that can be measured fundamentally are those that are both additive and independent[2] in combination (see [27]). Hempel also writes "(...) perhaps the only-type of fundamental measurement used in the physical sciences is illustrated by the fundamental measurement of mass, length, temporal duration (...)" (see [44]). All the experiments we consider and that are presented, for example, in [4], are measuring such quantities.

3.1.1 Fundamental measurement

According to Hempel, a physical measurement is divided into three stages: *classification*, *comparison* and *quantification*. During the *classificatory phase*, objects in a domain are sorted according to similarities regarding a given aspect. As Hempel writes in [44], "A classificatory concept represents (...) a characteristic which any object (...) must either have or lack". The chosen characteristic is termed an *attribute* which, should its meaning be precise, will divide the domain into two distinct classes. In a *comparative phase*, the objects in a class are compared as to having *more*, *less* or the *same*, of the chosen attribute. This comparison is done by means of observations of events, such as the tilting of a balance scale, which, in this case, indicates that one object weights more than the other. Finally, during a *quantitative phase*, the attributes are scaled by means of a map M, from objects to a number system.

Hempel gives a brief survey of the advantages offered by non-classificatory concepts, namely by the quantitative concept (see [44] – pag. 56). One advantage is that, by means of metrical concepts, it is often possible to differentiate between object that share the same classification. The introduction of metrical terms also makes it possible to formulate general laws, expressed in the form of functional relationships between different quantities.

[1] Consider, for example, the definition of density in terms of mass and volume.
[2] Independence here means that the measure of an object cannot interfere with the measure of another one.

From now on we will denote by $\mathcal{O}$ a class of physical objects, endowed with some attribute (or property), such as mass or length, and consider our number system to be the real numbers. Thus, a measurement of an attribute, in the sense of Hempel, will be a map $M : \mathcal{O} \to \mathbb{R}$ that satisfies a set of conditions (see Definition 3.2).

To sort and compare attributes, Hempel proposes a *comparative concept*, which is witnessed by an instrument, or *experimental apparatus*. This experiment has to be able to implement two comparative predicates, $\mathcal{E}$ and $\mathcal{L}$, over the set $\mathcal{O}$. These predicates have the semantics that $a\mathcal{E}b$, if a and b are *identical* in their measured attribute, and $a\mathcal{L}b$, if the measurement of a, regarding the chosen attribute, is *less* than that of b.

Definition 3.1. Let $\mathcal{L}$ and $\mathcal{E}$ be two binary relations. We define the following concepts:

- $\mathcal{L}$ is $\mathcal{E}$-irreflexive if, for every $a, b \in \mathcal{O}$, if $a\mathcal{E}b$ holds, then $a\mathcal{L}b$ does not hold;
- $\mathcal{L}$ is $\mathcal{E}$-connected if, for every $a, b \in \mathcal{O}$, if $a\mathcal{E}b$ does not hold, then $a\mathcal{L}b$ or $b\mathcal{L}a$ holds;
- $\mathcal{E}$ and $\mathcal{L}$ are a *comparative concept*, if $\mathcal{E}$ is an equivalence relation and $\mathcal{L}$ is transitive, $\mathcal{E}$-irreflexive, and $\mathcal{E}$-connected.

We can thus use a comparative concept to develop a “schema of ordered strata" (see [28]). We place each object into its equivalence class, regarding the equivalence relation $\mathcal{E}$, and then place these classes into a serial order, according to the relation $\mathcal{L}$.

Now consider, for example, that our domain consist of 6-sided dice, with each side having any integer value, and that we compare two dices according to which one has the highest probability of producing a greater value. Then, Efron’s dice provide a counter example to the transitivity of the relation $\mathcal{L}$ (see, for example, [69]). Thus, we will not be able to use this concept of comparison to assign a numerical value to each dice, in a way that can be considered a *measurement*. Something that is worth remarking is that, with this same domain, we might be able to attain a comparative concept, by sorting dice according to some other criteria, like the weight of a dice; the key aspect here is that we cannot produce a metrical concept, whose value will reflect the notion of one dice beating another one.

Definition 3.1 gives a first set of conditions under which a property is amenable to fundamental measurement. However, as Campbell writes in [26], “(...) the possession of order alone will not enable a property to be measured, except possibly by the use of previously established systems of measurement for other properties." He argues that for a property to be measurable as a fundamental magnitude, “(...) a physical process of addition should be found for it" (see [26]).

The significance of addition is made evident when we introduce the notion of a standard. As Campbell states in [26], “(...) the fixing of the weight of one body fixes uniquely the weight of all others, and yet the process of measuring weight does not involve the measurement of any other magnitude". So it seems that, in order for the measurement of a property not to depend on prior metrical concepts, it must

be an additive property.

If this “first law of addition", as Campbell called it, is satisfied, the *principle of fundamental measurement* is quite simple. Consider the example of mass. We start by fixing a standard object, whose weight we set to 1.[3] Then, given an object a, we say that its weight is n if a balance scale achieves equilibrium with a on one side and n instances of the unit on the other.

This notion of standards, seen also in [45] (pag. 63), when Hempel refers to the International Prototype Kilogram, and in [28] (pag. 64), when Carnap demands that a measurement determines when to assign “zero" and "one" to an object[4], is captured here by identifying a sub-structure of $\mathcal{O}$, consisting of a standard object, called the unit, to which we attribute the measure 1, together with all its multiples and submultiples: this substructure we call the *toolbox of standards*. If the quantity we are measuring satisfies the “first law of addition", we can assign a standard measure to every objects in the *toolbox*, using only the concept of natural numbers and multiplication. If experimental equality is achieved with 3 instances of a and 2 instances of the unit, we say that the measure of a is $2/3$.

Campbell states an extra criteria for fundamental measurement, which he termed the “second law of addition". It says: “The magnitude of a system produced by the addition of bodies A, B, C, $\ldots$ depends only on the magnitude of those bodies and not on the order or method of their addition;" (see [26]). This is a generalization of the following condition: if a_1 and a_2 are experimentally equal to b_1 and b_2, respectively, then so it their respective physical additions. Considering the example of weight, Campbell states a condition under which the second law of addition is implied by the first one: “The second law of addition will be true if the system which will balance a given combination of bodies in a pan is the same however those bodies are placed in the pan and if, when we add to each of the previously balanced pans one of a pair of bodies of equal weight, the pans will remain balanced." So the way we place objects to be compared has to be irrelevant, and the addition of equal objects cannot introduce inequalities in the measurement. This condition seems to depend more on the physical experiment, than on the nature of the quantity being measured, and is always verified by our ideal experiments.

3.1.2 Hempel’s axiomatization of measurement

In the previous section we presented a set of criteria under which a property is amenable to fundamental measurement. Now, we will present Hempel’s definition of when a given assignment from objects to numbers is to be considered a measurement. We will also derive some simple facts of this theory, corresponding to intuitive properties of a measurement.

[3] Note that this choice is, essentially, arbitrary. If we choose a different unit, we will obtain the same measurement, up to a constant.

[4] As Carnap points out, while these values are usually 0 and 1, this is not necessarily the case. As an example, he refers to the centigrade scale, in which the second standard is 100 degrees. Note that, when we measure temperature with a thermometer, fixing these standard will uniquely determine the temperature of any other object, *because* we have already established the metrical concept of length.

Definition 3.2. Let $\mathcal{L}$ and $\mathcal{E}$ be comparative concepts on the set $\mathcal{O}$ of objects. Suppose there is an experimental apparatus to witness these relations. Then, the map $M : \mathcal{O} \to \mathbb{R}$ is a *measurement map* if the following axioms are verified:

- If $a\mathcal{E}b$ holds, then so does $M(a) = M(b)$;
- if $a\mathcal{L}b$ holds, then so does $M(a) < M(b)$.

The previous axiomatization allows to prove simple results. In the following propositions, always assume that $\mathcal{E}$ and $\mathcal{L}$ are a comparative concept in $\mathcal{O}$.

Proposition 3.3. For every $a, b \in \mathcal{O}$, one and only one of the following statements holds: $a\mathcal{E}b$, $a\mathcal{L}b$, or $b\mathcal{L}a$.

Proof. First, to prove that at least one of the events holds. If $a\mathcal{E}b$ holds, then we are done. Otherwise, since $\mathcal{L}$ is $\mathcal{E}$-connected and $a\mathcal{E}b$ does not hold, then $a\mathcal{L}b$ or $b\mathcal{L}a$ holds. Now suppose that $a\mathcal{E}b$. Then, since $\mathcal{E}$ is an equivalence relation we also have that $b\mathcal{E}a$, which, since $\mathcal{L}$ is $\mathcal{E}$-irreflexive, means that $a\mathcal{L}b$ and $b\mathcal{L}a$ do not hold. If $a\mathcal{L}b$ holds, then, since $\mathcal{L}$ is $\mathcal{E}$-irreflexive, $a\mathcal{E}b$ doesn't hold. Also, if $b\mathcal{L}a$ were to hold, then, by the transitivity of $\mathcal{L}$, we would have $a\mathcal{L}a$. Thus, it would contradict the fact that $\mathcal{L}$ is $\mathcal{E}$-irreflexive, since $a\mathcal{E}a$, by $\mathcal{E}$'s reflexive property. The proof for when $b\mathcal{L}a$ is analogous to the previous case. □

Proposition 3.4. The converse of the axioms in Definition 3.2 hold, i.e., for any $a, b \in \mathcal{O}$:

- If $M(a) = M(b)$ holds, then so does $a\mathcal{E}b$;
- if $M(a) < M(b)$ holds, then so does $a\mathcal{L}b$.

Proof. The proof is done by contradiction. Suppose that $a\mathcal{E}b$ doesn't hold. Then, either $a\mathcal{L}b$ or $b\mathcal{L}a$ must hold, which, by the second axiom, implies that either $M(a) < M(b)$ or $M(b) < M(a)$ holds, whence $M(a) = M(b)$ can't hold. The other proof is analogous. □

Proposition 3.5. For every $a, b \in \mathcal{O}$, $a\mathcal{E}b$ if and only if, for every $u \in \mathcal{O}$, $(a\mathcal{L}u \Leftrightarrow b\mathcal{L}u)$ and $(u\mathcal{L}a \Leftrightarrow u\mathcal{L}b)$.

Proof. Suppose that $a\mathcal{E}b$ and let $u \in \mathcal{O}$. We will only prove that $a\mathcal{L}u \Rightarrow b\mathcal{L}u$, since the other implications are analogous. Suppose that $a\mathcal{L}u$, which, since $\mathcal{L}$ is $\mathcal{E}$-irreflexive, implies that $a\mathcal{E}u$ doesn't hold. Hence, given that $\mathcal{E}$ is an equivalence relation, $b\mathcal{E}u$ can't hold either, meaning that, since $\mathcal{L}$ is $\mathcal{E}$-connected, either $b\mathcal{L}u$ or $u\mathcal{L}b$ must hold. However, if $u\mathcal{L}b$ holds, then, since $\mathcal{L}$ is transitive, $a\mathcal{L}b$ would also hold, which would contradict the fact that $a\mathcal{E}b$, whence $b\mathcal{L}u$.

Suppose that, for every $u \in \mathcal{O}$, $(a\mathcal{L}u \Leftrightarrow b\mathcal{L}u)$ and $(u\mathcal{L}a \Leftrightarrow u\mathcal{L}b)$. By Proposition 3.3, we either have that $a\mathcal{E}b$, $a\mathcal{L}b$ or $b\mathcal{L}a$. Suppose that $a\mathcal{L}b$ is the case. Then, $b\mathcal{L}b$ would hold as well, which is a contradiction. Analogously, it is impossible for $b\mathcal{L}a$ to hold, so $a\mathcal{E}b$ must be the case. □

Proposition 3.6. For every $a, b \in \mathcal{O}$, if $a\mathcal{E}b$ and $b\mathcal{L}c$ then $a\mathcal{L}c$.

Proof. Suppose that $a\mathcal{L}c$ is not the case. Then, by Proposition 3.3, we must have that either $a\mathcal{E}c$ or $c\mathcal{L}a$. If $a\mathcal{E}c$, then, since $\mathcal{E}$ is an equivalence relation, $b\mathcal{E}c$, which contradict the fact that $b\mathcal{L}c$. If, on the other hand, $c\mathcal{L}a$, then, given that $\mathcal{L}$ is a transitive relation, we also have that $b\mathcal{L}a$, which contradict the fact that $a\mathcal{E}b$. □

3.2 Measuring quantities with time

In this section we will present the work done in [16], where Hempel's comparative concept is seen as a "limit" of relations with an indexation by a real parameter $t > 0$, which represents the physical duration of an experiment. This axiomatization sees the introduction of a constant K, which arises form the non-linearity of the duration of an experiment, as one approaches the unknown quantity, and the assumption that two objects a and b are compared by running an experiment with query a and unknown object b. In [16], the authors give the example of the CME[5] as an experiment that can satisfy this axiomatization.

Definition 3.7. A collection of relations $\{\mathcal{E}_t\}_{t>0}$ in $\mathcal{O} \times \mathcal{O}$ is said to be a timed equivalence relation if there is a non-negative constant $K \leq 1$ such that:

- For every $t > 0$, $\mathcal{E}_t$ is reflexive;
- $\{\mathcal{E}_t\}_{t>0}$ is timed symmetric, i.e., for every $t > 0$ and $a, b \in \mathcal{O}$, if $a\mathcal{E}_t b$, then $b\mathcal{E}_{tK}a$;
- $\{\mathcal{E}_t\}_{t>0}$ is timed transitive, i.e., for every $t > 0$ and $a, b, c \in \mathcal{O}$, if $a\mathcal{E}_t b$ and $b\mathcal{E}_t c$, then $a\mathcal{E}_{tK}c$;
- for every $t, t' > 0$ if $t < t'$, then $a\mathcal{E}_{t'}b \Rightarrow a\mathcal{E}_t b$.

Definition 3.8. Two collections of binary relations $\{\mathcal{E}_t\}_{t>0}$ and $\{\mathcal{L}_t\}_{t>0}$ determine a timed comparative concept for the elements of $\mathcal{O}$, if:

- For every $t > 0$ and $a, b \in \mathcal{O}$, exactly one of $a\mathcal{E}_t b$, $a\mathcal{L}_t b$, $b\mathcal{L}_t a$ holds;
- $\{\mathcal{E}_t\}_{t>0}$ is a timed equivalence relation;
- $\{\mathcal{L}_t\}_{t>0}$ is timed transitive, i.e., for every $t > 0$, there is a constant $K \leq 1$ such that, for every $a, b, c \in \mathcal{O}$, if $a\mathcal{L}_t b$ and $b\mathcal{L}_t c$, then $a\mathcal{L}_{tK}c$;
- for every $t, t' > 0$, if $t < t'$, then $a\mathcal{L}_t b$ implies that $a\mathcal{L}_{t'}b$.

From now on we will use the notation $\mathcal{E}_t$ and $\mathcal{L}_t$ to mean both the class of relations and a particular relation in the class. The meaning will be clear in the context.

We can assign any labeling to the possible outcomes of an experiment. For the case of the SmSM, for example, the outcomes are labeled q_l, q_t and q_r.

[5] The specification of the experiment can be found in [16]. We will give a brief description of the CME in Section 3.3.1.

Definition 3.9. We say that an experimental apparatus, with a given labeling of events, *witnesses* the relations $\mathcal{E}_t$ and $\mathcal{L}_t$ if, whenever the experiment is done with arbitrary objects $a, b \in \mathcal{O}$, the label of each outcome, in time t, implies one and only one of the relations $a\mathcal{L}_t b$, $b\mathcal{L}_t a$, or $a\mathcal{E}_t b$.

Definition 3.10. Let $\mathcal{E}_t$ and $\mathcal{L}_t$ be timed comparative relations on a set $\mathcal{O}$, of objects, witnessed by some experimental apparatus. Then, $M : \mathcal{O} \to \mathbb{R}$ is a *measurement map* if, for any time $t > 0$, whenever $a\mathcal{L}_t b$ holds, so does $M(a) < M(b)$.

Definition 3.11. An apparatus, witnessing a timed comparative concept $\mathcal{E}_t$ and $\mathcal{L}_t$, satisfies the *separation property* for a measurement map $M : \mathcal{O} \to \mathbb{R}$ if, for every objects $a, b \in \mathcal{O}$, whenever $M(a) < M(b)$ holds, then so does $a\mathcal{L}_t b$, for some time bound $t > 0$.

Note the relation between Definitions 3.10 and 3.11. Together, they say that $M(a) < M(b)$ if and only if there is a time t, such that the relation $a\mathcal{L}_t b$ is satisfied.

We will now see how to recover Hempel's notion of a comparative concept and a measurement map, as a "limit" of a timed comparative concept and a map satisfying Definitions 3.10 and 3.11.

Definition 3.12. Given a timed comparative concept $\mathcal{E}_t$ and $\mathcal{L}_t$, we define the relations $\mathcal{E}_{lim}$ and $\mathcal{L}_{lim}$ as follows:

- $a\mathcal{E}_{lim}b$ if, for every time bound t, $a\mathcal{E}_t b$;
- $a\mathcal{L}_{lim}b$ if there is a time bound t such that $a\mathcal{L}_t b$.

Proposition 3.13. Suppose there is physical apparatus witnessing a timed comparative concept $\mathcal{E}_t$ and $\mathcal{L}_t$ and that a measurement map is defined, in the sense of Definition 3.10, such that the apparatus satisfies the separation property. Then, in the sense of Hempel, $\mathcal{E}_{lim}$ and $\mathcal{L}_{lim}$ are a comparative concept and M is a measurement map.

Proof. We will prove that Hempel's axiomatization holds for these "limit" relations:

- $\mathcal{E}_{lim}$ is reflexive: Trivial, since each $\mathcal{E}_t$ is reflexive;
- $\mathcal{E}_{lim}$ is symmetric: If $a\mathcal{E}_{lim}b$, then $a\mathcal{E}_t b$ for every time bound t. Then, for every time bound, $b\mathcal{E}_{tK}a$, which implies that $b\mathcal{E}_t a$, for every time bound, whence $b\mathcal{E}_{lim}a$;
- $\mathcal{E}_{lim}$ is transitive: Same argument as before, since each $\mathcal{E}_t$ is timed transitive;
- $\mathcal{L}_{lim}$ is transitive: If $a\mathcal{L}_{lim}b$ and $b\mathcal{L}_{lim}c$, then $a\mathcal{L}_t b$ and $b\mathcal{L}_{t'}c$, for some time bounds t, t'. We can assume that $t < t'$, which implies that $a\mathcal{L}_{t'}b$. From here, we conclude that $a\mathcal{L}_{t'K}c$, which concludes the proof;
- $\mathcal{L}_{lim}$ is $\mathcal{E}_{lim}$-irreflexive: If $\mathcal{E}_{lim}$ holds, then for every time t, $a\mathcal{E}_t b$ holds, whence we can never have one of the events that determines $\mathcal{L}_{lim}$;

- $\mathcal{L}_{lim}$ is $\mathcal{E}_{lim}$-connected: If $a\mathcal{E}_{lim}b$ doesn't hold, then there is t such that $a\mathcal{E}_t b$ doesn't hold. The proof is completed by remarking that each $\mathcal{L}_t$ is $\mathcal{E}_t$ connected;
- If $a\mathcal{E}_{lim}b$ then $M(a) = M(b)$: If $M(a) < M(b)$, then, by the separation property, there is a bound t such that $a\mathcal{L}_t b$ holds, meaning that $a\mathcal{E}_t b$ doesn't hold, which implies that $a\mathcal{E}_{lim}b$ doesn't hold;
- If $a\mathcal{L}_{lim}b$ then $M(a) < M(b)$: If $a\mathcal{L}_{lim}b$, then there is a bound t such that $a\mathcal{L}_t b$. Since M is a measurement map, we will have $M(a) < M(b)$.

□

3.3 Limit measurement

In this section we will consider a different way of performing comparisons. Instead of comparing two object $a, b \in \mathcal{O}$ directly, we will compare them by running two experiments with the same query: one with unknown quantity a and the other with unknown quantity b.[6] This procedure is inspired by the SmSM, in which one of the objects in an experiment, the query, is necessarily in a dyadic position. Therefore, unlike the case of the CME, where we can compare the mass of any two objects by colliding one with the other, we can't set the position of the cannon to an arbitrary real value. Note that an axiomatization of measurement for this second type of experiment, in which the attribute of one of the objects is a dyadic rational, is also applicable to experiments of the first type; we can always ignore the fact that we can compare objects directly and perform comparisons using only dyadic rationals.

For this axiomatization, the time parameter t will have a discrete set of possible values, which are determined by the time the experimental apparatus waits to receive an answer, i.e., by the machine's schedule. We will assume that we are always working with machines defined with an increasing schedule.[7] Given this particular way of performing a comparison, we can assume that the constant K, introduced in the previous section, is equal to one. We will support this claim by showing that the CME can be made to satisfy this axiomatization. We will skip the initial formalism from the previous section and use the notation $\mathcal{E}_t$ and $\mathcal{L}_t$, straightaway, to denote both a relation indexed in time and a collection of relations.

Definition 3.14. Two collections of binary relations $\mathcal{E}_t$ and $\mathcal{L}_t$ determine a *limit* timed comparative concept for the elements of $\mathcal{O}$, if, for every $t > 0$ and $a, b, c \in \mathcal{O}$, the following hold:

- Exactly one of $a\mathcal{E}_t b$, $a\mathcal{L}_t b$, $b\mathcal{L}_t a$ holds;
- $\mathcal{E}_t$ is reflexive and symmetric;

[6] This methodology is in conformity with the example given by Carnap to compare the temperature of two objects: "We simply place the thermometer in contact with body a, wait until there is no longer any change in the height of the test liquid, then mark the level of the liquid. We apply the thermometer in the same way to object b." (see [28] pag. 64)

[7] Recall Definition 3.12. If we worked with a non-increasing schedule, we would not be able to let the parameter t increase arbitrarily.

- $\mathcal{L}_t$ is transitive;
- if $a\mathcal{L}_t c$ holds, there is a time T such that, if $a\mathcal{E}_T b$ holds, then $b\mathcal{L}_T c$ holds as well;
- if $a\mathcal{L}_t b$ holds, then there is an order T after which $a\mathcal{L}_{t'} b$ holds, for any $t' \geq T$.

We define a measurement map and the relations $\mathcal{E}_{lim}$ and $\mathcal{L}_{lim}$ as in Definition 3.12.

Proposition 3.15. Suppose there is physical apparatus witnessing a *limit* timed comparative concept $\mathcal{E}_t$ and $\mathcal{L}_t$ and that a measurement map is defined, in the sense of Definition 3.10, such that the apparatus satisfies the separation property. Then, in the sense of Hempel, $\mathcal{E}_{lim}$ and $\mathcal{L}_{lim}$ are a comparative concept and M is a measurement map.

Proof. The proof is very similar to that of Proposition 3.13, the only differences being in showing that $\mathcal{E}_{lim}$ and $\mathcal{L}_{lim}$ are transitive.

Suppose that $a\mathcal{E}_{lim} b$ and $b\mathcal{E}_{lim} c$ hold, and that there is some time bound t, such that $a\mathcal{L}_t c$ holds as well. Then, there is a time T such that, if $a\mathcal{E}_T b$ holds, then $b\mathcal{L}_T c$ holds as well. The contradiction is achieved by noticing that, by definition of $\mathcal{E}_{lim}$, $a\mathcal{E}_T b$ will necessarily hold, regardless of how large T is.

Now suppose that $a\mathcal{L}_{lim} b$ and $b\mathcal{L}_{lim} c$ are verified. Then, there are time bounds t, t', such that both $a\mathcal{L}_t b$ and $b\mathcal{L}_{t'} c$ hold. This means that there are orders T, T', after which these relations are always satisfied. Then, for $t'' \geq \max\{T, T'\}$, both $a\mathcal{L}_{t''} b$ and $b\mathcal{L}_{t''} c$ will hold, whence so will $a\mathcal{L}_{t''} c$. By definition, this implies that $a\mathcal{L}_{lim} c$ is verified. □

One might wonder if the last condition of Definition 3.14 can be replaced with "for every $t > 0$, if $a\mathcal{L}_t b$ holds, then $a\mathcal{L}_{t'} b$ holds as well, for some $t' > t$." This definition would just ensure that $a\mathcal{L}_{t'} b$ holds infinitely often, after some bound T. However, if that were the case, we wouldn't be able to ensure that a t'' exists, such that both $a\mathcal{L}_{t''} b$ and $b\mathcal{L}_{t''} c$ are satisfied, in the proof that $\mathcal{L}_{lim}$ is a transitive relation.

As a final remark, we clarify the thought process behind defining a limit timed comparative concept.

The idea is to start with an *ideal comparative concept*, in the sense of Hempel, through which we wish to assign a numerical value to the elements of our domain, and an experimental apparatus, design to compare objects according to this concept (consider for example, the balance scale, which compares weights). Our relations indexed in time t, will then be defined through the ability of the experimental apparatus to witness that two objects are different, according to that ideal concept, in time t. We will say that $a\mathcal{L}_t b$ if, using our experimental apparatus, we are able to tell that $a\mathcal{L}b$, in time t, and that $a\mathcal{E}_t b$, if that is not the case. In the limit, i.e., in the sense of Definition 3.12, we recover the ideal comparative concept we wanted to define.

3.3.1 The CME as an example

In this section, following the work from [16], we prove that the CME can be used to define a *limit* timed comparative concept, when comparisons are performed by running two experiments with the same

query. Figure 3.1 contains a schematic depiction of the CME.

Figure 3.1: Schematic representation of the CME.

We start with a particle a, at rest and with unknown mass m_a, at which we fire another particle b, with a dyadic mass m_b. The collision between the two particles is assumed to be elastic and one dimensional. We can have the following outcomes: if $m_a > m_b$, b moves backwards after the collision; if $m_a < m_b$, b moves forward; if $m_a = m_b$, b comes to rest and a is projected forward with speed v_b. We setup detectors to the left and to the right of the masses, which signal when the test mass b crossed them. Note that, the more m_b approaches m_a, the more time it will take for b to cross a flag. As for the case of the SmSM, we define the CME with a time constructible schedule T, which determines how much time the machine waits for an experiment to be completed. We will always assume that this schedule is increasing, i.e., that a greater input precision will result in a greater waiting time.

To perform a comparison between a and b we will run two experiments with a single dyadic query of mass z. Therefore, there will be 9 possible overall responses, 3 from each experiment, depending on whether or not the test particle crossed a detector and, if so, which one. We will identify the responses q_l, q_t and q_r with the mass at which a particle was fired. For example, q_t^a will mean that we fired a particle with mass z towards a and the outcome was a time out.

Definition 3.16. We say that $a\mathcal{E}_t b$ if, for every dyadic mass z, firing a particle with mass z yields the same result from both experiment, in time t. We say that $a\mathcal{L}_t b$ if there is a dyadic rational z such that, when a particle with mass z is fired, we get one of the following results in time t: (q_r^a, q_l^b), (q_r^a, q_t^b), or (q_t^a, q_l^b). In this case, we say that z witnesses the relation $a\mathcal{L}_t b$.

When we say that the outcome of an experiment was in time t, we mean that an experiment was ran with a query z such that $t = T(|z|)$.

Proposition 3.17. If there is $t > 0$, such that $a\mathcal{L}_t b$ holds, then a is lighter than b.

Proof. Let z be the dyadic rational that witnesses the relation $a\mathcal{L}_t b$. If the outcome is (q_r^a, q_l^b), then $m_a < z$ and $z < m_b$, whence $m_a < m_b$; if we get (q_r^a, q_t^b), then $m_a < z$ and m_b is closer to z than m_a, whence $m_a < m_b$; if the outcome is (q_t^a, q_l^b), the argument is analogous to the previous case. □

We will now prove that $\mathcal{E}_t$ and $\mathcal{L}_t$ are a *limit* timed comparative concept.

Proposition 3.18. Given a time t and $a, b \in \mathcal{O}$, one and only one of the relations $a\mathcal{L}_t b$, $b\mathcal{L}_t a$, or $a\mathcal{E}_t b$ will hold.

Proof. If $a\mathcal{E}_t b$ doesn't hold, then there is a dyadic mass z such that the result of the two experiments is different in time t. This implies that one of the events that determine $a\mathcal{L}_t b$ or $b\mathcal{L}_t a$ will happen.

Now, to see that only one of the events can happen. Suppose $a\mathcal{E}_t b$ is the case. Then, every firing yields the same result in time t, whence none of the cases that determine $a\mathcal{L}_t b$ or $b\mathcal{L}_t a$ can happen. If, for example, $a\mathcal{L}_t b$ holds, $a\mathcal{E}_t b$ is excluded by definition and, since a is lighter than b, $b\mathcal{L}_t a$ can't happen. □

Proposition 3.19. For every $t > 0$, $\mathcal{E}_t$ is reflexive and symmetric.

Proof. To see that $\mathcal{E}_t$ is *reflexive*, we just have to observe that, if we setup both experiments with the same unknown mass a, then we will always get the same answer from both experiments, for every time t. The fact that $\mathcal{E}_t$ is *symmetric* comes from the fact that we can juts change the order of the experiments and the results will stay the same. □

Proposition 3.20. For every $t > 0$, $\mathcal{L}_t$ is transitive.

Proof. Suppose that $a\mathcal{L}_t b$ and $b\mathcal{L}_t c$ hold, and let z be the witness of the relation $a\mathcal{L}_t b$. Then, if we setup two experiments, with unknown vertices a and c, z will be a witness of $a\mathcal{L}_t c$. If the answer from firing with mass z towards a was q_r^a, the answer from firing towards b must have been q_t^b or q_l^b. Then, since c is heavier than b, the result of running the experiment with a and c can only be either (q_r^a, q_l^c) or (q_r^a, q_t^c); if, on the other hand, we got (q_t^a, q_l^b), when firing with mass z, we will have (q_t^a, q_l^c) when firing wit mass z, since c is heavier than b. □

Proposition 3.21. For every $t > 0$ and $a, b, c \in \mathcal{O}$, if $a\mathcal{L}_t c$ holds, there is a time T such that, if $a\mathcal{E}_T b$ holds, then $b\mathcal{L}_T c$ holds as well.

Proof. Suppose that $a\mathcal{L}_t c$ holds, which implies that $m_a < m_c$. Suppose that $a\mathcal{E}_T b$ holds, for $T = T(k)$, where k is the precision required to perform firings towards a and b, with queries in the interval $(m_a, (m_a + m_c)/2)$, with at least one answer different from a time out. Then, since the queries are heavier than m_a, we must have gotten the answer (q_r^a, q_r^b), for some query $z \in (m_a, (m_a + m_c)/2)$. This means that, if we setup two experiments, with b and c, firing with mass z will yield the answer (q_r^b, q_l^c), whence the relation $b\mathcal{E}_T c$ will hold. □

Proposition 3.22. If $a\mathcal{L}_t b$ holds, for $t > 0$, there is an order T after which $a\mathcal{L}_{t'} b$ holds, for every $t' \geq T$.

Proof. Consider a dyadic rational z, that witnesses the relation $a\mathcal{L}_t b$. Then, since a lighter than b, there is another dyadic rational z' such that $m_a < z' < m_b$. Since the experimental time of firing with mass z' towards a and b is finite, we can pad z' with enough zeros[8], obtaining a query z'', such that $T = T(|z''|)$ is greater than both experimental times. We will thus have a witness for the relation $a\mathcal{L}_T b$. For any time greater than T, we can just use the same query, but padded with more zeros. □

[8]Note that there doesn't have to be a computable way of knowing the number of zeros that have to be padded to z'; we just require that such a number exists.

Remark 3.23. Note that this order T is achieved, at most, when there is a query z for which the outcome is (q_r^a, q_l^b). For any time order above T, an experiment with that same query will always yield the answer (q_r^a, q_l^b).

These propositions, together, imply the following result.

Proposition 3.24. The relations $\mathcal{E}_t$ and $\mathcal{L}_t$, as defined in 3.16, are a *limit* timed comparative concept.

3.4 Three types of measurement

In this section we will prove that the three types of experiments, two-sided, one-sided and vanishing, can be used to satisfy the axiomatization we presented. As pointed out in [20], these three types of experiments lead to three forms of comparison: signed comparison, if both $a\mathcal{L}b$ and $b\mathcal{L}a$ can be tested, threshold comparison, if either $a\mathcal{L}b$ or $b\mathcal{L}a$ can be tested, and vanishing comparison, if we can only test the predicate ($a\mathcal{L}b$ or $b\mathcal{L}a$).

To represent each type of experiment, we will use a SmSE with some alterations in the collecting boxes. We will assume that a SmSM is always defined with an increasing schedule. The idea to define the relations $\mathcal{E}_t$ and $\mathcal{L}_t$ is to run two experiments, each with a vertex in a different position. We will represent the two experiments as a single one, by introducing the TSmSE, whose schematic representation is presented in figure 3.2. The experiment works as the regular SmSE, with the difference that the cannon, placed between the two curves, shoots a particle to the left and another to the right.

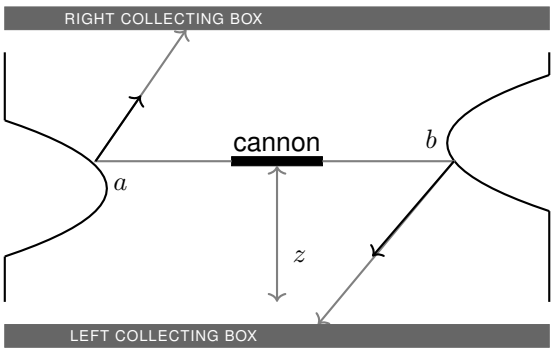

Figure 3.2: Schematic representation of the TSmSE.

In this case, the set $\mathcal{O}$, of objects, is the wedge vertices that can be used to setup the TSmSE, and the time parameter t will mean the possible waiting times of an experiment, which, determined by the machine's schedule, is assumed to be increasing. Unless explicitly stated otherwise, assume that the TSmSE is setup with the vertex a on the left and b on the right.

3.4.1 Two-sided type measurement

For the case of two-sided type measurement, the TSmSE is built from two regular SmSE. Instead of the TSmSE having four collecting boxes, its two collecting boxes are assumed to be able to distinguish if the particle that crossed them was shot towards a or b. We will index the possible oracle responses (q_l, q_t, q_r) in the vertex where the response came from.

Definition 3.25. We say that $a\mathcal{E}_t b$ if, for every dyadic position z, firing the cannon at z yields the same result from both sides, in time t. We say that $a\mathcal{L}_t b$ if there is a dyadic rational z such that, when the cannon fires at z, we get one of the following results in time t: (q_r^a, q_l^b), (q_r^a, q_t^b), or (q_t^a, q_l^b). In this case, we say that z witnesses the relation $a\mathcal{L}_t b$.

Since the CME is a two-sided type experiment, the proof that the relations defined in 3.25 are a *limit* timed comparative concept is analogous to the proof presented in Section 3.3.1. Thus, we will only state the final result.

Proposition 3.26. The relations $\mathcal{E}_t$ and $\mathcal{L}_t$, as defined in 3.25, are a *limit* timed comparative concept.

To measure a vertex, we successively compare it with vertices in dyadic positions. Hence we can fix one of the vertices of the TSmSE and set the other to a dyadic position. However, since we perform comparisons by setting the cannon to a dyadic position, this experiment is reduced to running a measurement algorithm in the regular SmSE.

Definition 3.27. We define the measurement map M as the function which, given a vertex y, successively runs Algorithm 2.1, with increasing input precision, and returns the limit of the sequence of generated dyadic rationals.

Recall the notion of the toolbox of standards, presented in Section 3.1. In the case of the SmSM, the standards correspond to vertices in dyadic positions, which we represent by dyadic positions of the cannon. We are thus measuring an arbitrary vertex by successively comparing it with standard objects.

We will consider that Algorithm 2.1 has been changed so that, when a time out occurs, instead of setting z_0 and z_1 to z, the value of the variables is saved. This means that the next iteration of the algorithm can start with this same assignment, instead of setting z_0 and z_1 to their default values again. Note that, for the case of a wedge vertex in a dyadic position, this will mean that, after a certain point, all the approximations experimental calls with result in a time out.

We will now prove that the map M converges to the exact position of any vertex. Since this will be the case for the three types of measurements, we will state the result in a more general manner, so that it applies to every situation.

Proposition 3.28. Let P be an algorithm for the SmSM, for which a greater input precision corresponds to an output with a smaller distance to the wedge vertex y.[9] Suppose that a map $M : \mathcal{O} \to \mathbb{R}$ is defined as in Definition 3.27, as a "limit" of P. Then, given a vertex y, $M(y)$ will converge to the exact position of y.

Proof. If, during the measurement procedure, there is always an iteration such that a time out does not occur, each such successive call to P will result in a better approximation of y, so the outputted queries will converge to the exact position of the vertex.

Now suppose that, after a given iteration with a query z, all the experimental calls yield a time out. If y is not in the dyadic position z, the experimental time of firing at z towards y is finite. This means that, for a given input precision l, the waiting time of the machine $T(l)$ will surpass the time of the experiment, whence a time out will not occur. Thus, if, at a given time, all the queries result in a time out, y must be in the dyadic position given by the queries (which differ only in the number of padded zeros.) □

Proposition 3.29. The map M is a measurement map.

Proof. Suppose that there is $t > 0$ such that $a\mathcal{L}_t b$ is the case. Then, since a is to the left of b, the fact that $M(a) < M(b)$ holds is implied by Proposition 3.28. □

Proposition 3.30. The measurement map satisfies the separation property 3.11.

Proof. Suppose that $a, b \in \mathcal{O}$ are such that $M(a) < M(b)$.

Consider the case where a is a dyadic rational (the other case is solved analogously). Then, after a certain point, the approximations z_i^a, given by the measurement map defined in 3.27, will be constantly equal to a. Then, since $M(a) < M(b)$, the result of firing at $z = a$ towards b will have to be q_l, for a z padded with enough zeros. This z will be a witness of the relation $a\mathcal{L}_t b$, for $t = T(|z|)$.

Now consider the case where neither a nor b is a dyadic rational and let z be a dyadic rational, such that $M(a) < z < M(b)$. Then, for a certain query z', equal to z, but padded with enough zeros, the result of firing at z' towards a and b must be q_r and q_l, respectively. This z' will be a witness of the relation $a\mathcal{L}_t b$, for $t = T(|z'|)$. □

3.4.2 One-sided type measurement

For the case of one-sided, or threshold, type measurement, we can only make one type of comparison: either "less than" or "more than". This type of experiment was studied in depth in [13].

Consider, for example, the photoelectric effect experiment, depicted in Figure 3.3, in which we wish to find the minimum frequency (threshold) after which a photon, shot towards a metallic surface, will cause

[9] Since we only consider increasing schedules, this condition is verified by each type of experiment's measurement algorithm.

an electron to be ejected. Since we only get a response whenever the light beam frequency exceeds the threshold frequency, the photoelectric experiment is a one-sided type experiment.

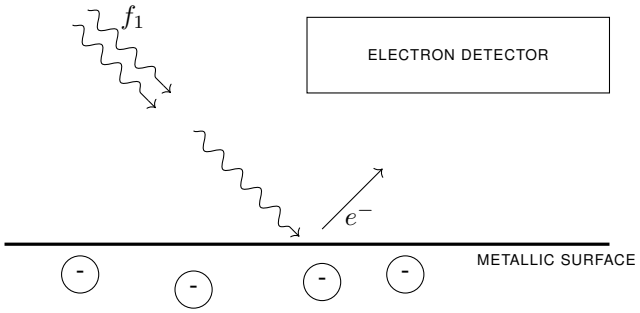

Figure 3.3: Schematic representation of the photoelectric effect experiment.

To better visualize this type of measurement, consider the BBE, shown in Figure 3.4, in which a pressure sensitive stick signals that the balance scale has tilted to one side, but a rigid block prevents the other side of the balance from moving downwards.

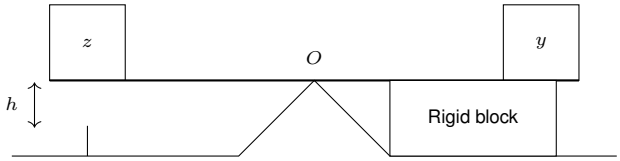

Figure 3.4: Schematic representation of the broken balance experiment.

Clearly, we can only tell if $z > y$, which is indicated by the pressure sensitive stick being triggered.

The computational power of this type of experiment, when running in polynomial time and with an exponential schedule, is the same as for the case of the two-sided type measurement and is summarized in Table 5.1 (see [13]).

In terms of the SmSE, we have a one-sided type measurement if only one of the collecting boxes is functional. We will study the case where the left one works, which means that the oracle responses will either be q_l, meaning that $z < y$, or q_t, which implies that y is either very close to z, or smaller than z. We will call this model one-sided SmSE.

Analogously to the measurement algorithm performed by the regular SmSM, we have Algorithm 3.1 to approximating the binary expansion of an unknown position y.

To define the semantic of $\mathcal{E}_t$ and $\mathcal{L}_t$ we will setup a one-sided TSmSE using two one-sided SmSE. This means that the right collecting box of the TSmSE is not functional and the left collecting box is able to tell if the particle that crosses it was shot towards a or b. We thus have four possible outcomes from an experiment: (q_l^a, q_l^b), (q_l^a, q_t^b), (q_t^a, q_l^b),or (q_t^a, q_t^b). The first case, for example, happens when we get a signal from both a's and b's left collecting box.

Algorithm 3.1: Measurement algorithm for one-sided type measurement.

Data: Positive integer l, representing the desired accuracy;
$z_0 = 0$; $z_1 = 1$; $z = 0$;
while $z_1 - z_0 > 2^{-l}$ **do**
    $z = (z_0 + z_1)/2$;
    $s =$ Prot_IP($z\restriction_l$);
    if $s == q_l$ **then**
        $z_0 = z$;
    else
        $z_1 = z$;
return z;

Definition 3.31. We say that $a\mathcal{E}_t b$ if, for every dyadic position z, firing the cannon at z yields the same result from both sides, in time t. We say that a $b\mathcal{L}_t a$ if there is a dyadic rational z such that, when the cannon fires at z, we get the result (q_t^a, q_l^b) in time t. Clearly, if $a\mathcal{L}_t b$ holds, a is to the left of b.

We will now see that $\mathcal{E}_t$ and $\mathcal{L}_t$ are a *limit* timed comparative concept. The proofs are analogous to the two-sided case.

Proposition 3.32. Given a time t and $a, b \in \mathcal{O}$, one and only one of $a\mathcal{L}_t b$, $b\mathcal{L}_t a$ and $a\mathcal{E}_t b$ will hold.

Proof. If $a\mathcal{E}_t b$ doesn't hold, then there is a dyadic rational z such that the result of the experiment is either (q_l^a, q_t^b) or (q_t^a, q_l^b), whence $a\mathcal{L}_t b$ or $b\mathcal{L}_t a$ will happen.

Now, to see that only one of the events can happen. Suppose $a\mathcal{E}_t b$ is the case. Then, every firing yields the same result in time t, whence none of the cases that determine $a\mathcal{L}_t b$ or $b\mathcal{L}_t a$ can happen. If, for example, $a\mathcal{L}_t b$ holds, $a\mathcal{E}_t b$ is excluded by definition and, since a is to the left of b, $b\mathcal{L}_t a$ can't happen. □

Proposition 3.33. For every $t > 0$, $\mathcal{E}_t$ reflexive and symmetric.

Proof. See the proof of Proposition 3.19. □

Proposition 3.34. For every $t > 0$, $\mathcal{L}_t$ is transitive.

Proof. Suppose that $a\mathcal{L}_t b$ and $b\mathcal{L}_t c$ hold, and let z be the witness of the relation $a\mathcal{L}_t b$. Then, if we setup two experiments, with unknown vertices a and c, z will be a witness of $a\mathcal{L}_t c$. Since the result of firing at z towards a and b must have been (q_t^a, q_l^b) and c is farther from z than b is, the result of firing at z towards a and c must be (q_t^a, q_l^c), whence $a\mathcal{L}_t c$ will hold. □

Proposition 3.35. For every $t > 0$ and $a, b, c \in \mathcal{O}$, if $a\mathcal{L}_t c$ holds, there is a time T such that, if $a\mathcal{E}_T b$ holds, then $b\mathcal{L}_T c$ holds as well.

Proof. Suppose that $a\mathcal{L}_t c$ holds, and let z be the dyadic rational that produced the outcome (q_t^a, q_l^c).

Suppose that $a\mathcal{E}_T b$ holds, for $T = T(k)$, where k is the precision required to perform firings in the interval $(a, (a+c)/2)$, such that the result from c is q_l, and let $z \in (a, (a+c)/2)$. Then, since z is to the right of a, we will have the outcome q_t^a, since the right collector isn't functional, whence, since $a\mathcal{E}_t b$ holds, we will also have the outcome q_t^b from firing at z towards b. Since we got q_l^c from the query z, we have the outcome (q_t^b, q_l^c) in time T, whence $b\mathcal{E}_T c$ will hold. □

Proposition 3.36. If $a\mathcal{L}_t b$ holds, for $t > 0$, there is an order T after which $a\mathcal{L}_{t'} b$ holds, for every $t' \geq T$.

Proof. Consider a dyadic rational z, that witnesses the relation $a\mathcal{L}_t b$. Then, since a is to the left of b, there is another dyadic rational z' such that $a < z' < b$. Since the experimental time of firing at z' towards a and b is finite, we can pad z' with enough zeros, obtaining a query z'', such that $T = T(|z''|)$ is greater than both experimental times. We will thus have a witness for the relation $a\mathcal{L}_T b$. For any time greater than T, we can just use the same query, but padded with more zeros. □

These propositions, together, imply the following result.

Proposition 3.37. The relations $\mathcal{E}_t$ and $\mathcal{L}_t$, as defined in 3.31, are a *limit* timed comparative concept.

Definition 3.38. We define the measurement map M as the function which, given a vertex y, successively runs Algorithm 3.1, with increasing input precision, and returns the limit of the sequence of generated dyadic rationals.

Proposition 3.39. The map M is a measurement map.

Proof. See the proof of Proposition 3.29. □

Proposition 3.40. The measurement map satisfies the separation property 3.11.

Proof. Suppose that $a, b \in \mathcal{O}$ are such that $M(a) < M(b)$.

Consider the case where a is a dyadic rational (the other case is solved analogously). Then, after a certain point, the approximations z_i^a, given by the measurement map defined in 3.27, will be constantly equal to a. Then, since $M(a) < M(b)$, the result of firing at $z = a$ towards b will have to be q_l, for a z padded with enough zeros. This z will be a witness of the relation $a\mathcal{L}_t b$, for $t = T(|z|)$.

Now consider the case where neither a nor b is a dyadic rational and let z be a dyadic rational, such that $M(a) < z < M(b)$. Then, for a certain query z', equal to z, but padded with enough zeros, the result of firing at z' towards a and b must be q_t and q_l, respectively. This z' will be a witness of the relation $a\mathcal{L}_t b$, for $t = T(|z'|)$. □

3.4.3 Vanishing type measurement

Vanishing type measurement occurs when we can test the predicate $z \neq y$, but can't distinguish $z < y$ from $y < z$. This type of measurement was studied in [14].

Consider, for example, the Brewster angle experiment, first suggested as an example of a vanishing type experiment in [18]. The goal of this experiment is to measure Brewster's angle, the angle of incidence of polarized light on an insulating surface, at which the electric vector of the reflected light has no component in the plane of incidence (see [23]).[10]

The experimental apparatus consists of a surface, a projector and a light detector, that reacts when it has absorbed energy above a threshold limit. We send a light beam into the surface, with a certain incidence angle φ, and await a response from the light detector (see figure 3.5).

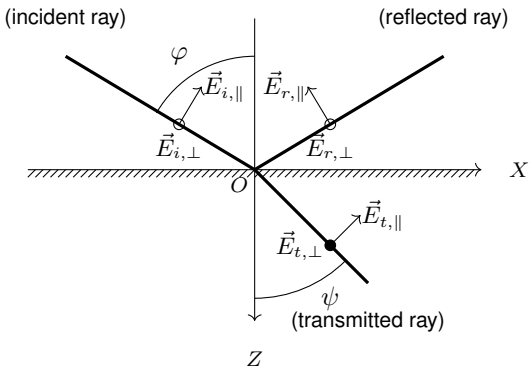

Figure 3.5: Schematic representation of the Brewster angle experiment.

The electric field is perpendicular to the direction of propagation and can be decomposed into components parallel (subscript $\parallel$) and perpendicular (subscript $\perp$) to the plane of incidence. Let $E_{i,\parallel}$ and $E_{i,\perp}$ denote the components of the incident ray, $E_{r,\parallel}$ and $E_{r,\perp}$ the components of the reflected ray, and $E_{t,\parallel}$ and $E_{t,\perp}$ the components of the transmitted ray. The black circle denotes the normal component pointing forward and the white circle denotes the normal component pointing backwards

In this type of experiment, we cannot infer any information about the Brewster angle φ_B, simply by sending a ray with desired angle φ. This is because, as φ approaches φ_B, the intensity of the electric field decreases to 0, either if $\varphi < \varphi_B$, or if $\varphi > \varphi_B$.

Consider the simpler example of the VBE, depicted in Figure 3.6, in which the pressure sensitive sticks signal that the balance has tilted to one side, but not to which one. Clearly, we can tell if $z \neq y$, if the pressure sensitive is triggered, but have no way of distinguishing $z < y$ from $z > y$.

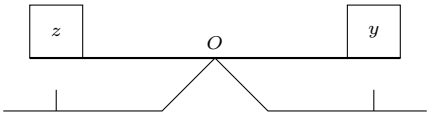

Figure 3.6: Schematic representation of the vanishing balance experiment.

[10] A visualization of the Brewster angle experiment can be found in https://micro.magnet.fsu.edu/primer/java/polarizedlight/brewster.

In terms of the SmSE, we have a vanishing type measurement if both collecting boxes are functional, but the SmSE doesn't distinguish between a signal that arrived from the left or from the right collecting box. This means that the oracle responses will either be that the particle has crossed one of the collecting boxes or that there was a time out. We will call this model vanishing SmSM.

The idea behind measuring an unknown quantity, testing only the predicate of inequality, is to run the experiment with two queries, to see which of them took the longest to return an answer. This will be the one that is the closest to the unknown value.

Two ways of performing vanishing type measurement are introduced in [14]: on one hand, we can setup two experiments in *parallel*, with the two possible queries, and wait to see which one took the longest; on the other hand, we can use just one experiment and count the experimental time for each query, performing a comparison afterwards. Their respective computational power, when running in polynomial time, is presented in Tables 3.1 and 3.2 (see [14]).

	Infinite	Unbounded	Bounded
Lower bound	$\mathtt{P/poly}$	$\mathtt{P/poly}$	$\mathtt{BPP//log}\star$
Upper bound	$\mathtt{P/poly}$	$\mathtt{P/poly}$	$\mathtt{BPP//log}\star$

Table 3.1: Computational results for the parallel implementation of vanishing type experiments.

	Infinite	Unbounded	Bounded
Lower bound	$\mathtt{P/log}\star$	$\mathtt{BPP//log}\star$	$\mathtt{BPP//log}\star$
Upper bound	$\mathtt{P/poly}$	$\mathtt{P/poly}$	$\mathtt{BPP//log}\star$
Upper bound Exponential T	–	$\mathtt{BPP//log}\star$	–

Table 3.2: Computational results for the time-counting implementation of vanishing type experiments.

3.4.3.A Parallel implementation

For this implementation, we assume that we can always distinguish the two events from one another. Although an unfeasible assumption, "we are willing to consider it because it will provide us with some interesting results later on" (see [14]). We thus have the Protocol 3.7, which, given a time constructible schedule T, attempts to signal the machine which particle arrived first.

We use Algorithm 3.2 to approximate the position of the unknown vertex with a given precision l.

PROTOCOL COMPARE_IP(z_1, z_2)

Receive as input two dyadic rationals $z_1 < z_2$ of size n;
Run two experiments, one for each query, and wait constructive $T(n)$ units of time;
Check which experiment finished first;
If it's the first experiment, return "first";
If it's the second, return "second";
If neither instance calls back, return "time out".

Figure 3.7: Protocol for the parallel implementation of the infinite precision vanishing type measurement.

Algorithm 3.2: Measurement algorithm for parallel vanishing type measurement.

Data: Positive integer l, representing the desired accuracy;
$z_0 = 0$; $z_4 = 1$; $z_2 = (z_0 + z_4)/2$;
while $z_4 - z_0 > 2^{-l}$ **do**
  $z_1 = (z_0 + z_2)/2$; $z_3 = (z_2 + z_4)/2$;
  $s1 =$ compare_IP$(z_1\downarrow_l, z_2\downarrow_l)$; $s2 =$ compare_IP$(z_2\downarrow_l, z_3\downarrow_l)$;
  if $s_1 ==$ *"second"* **then**
    $z_2 = z_1$; $z_4 = z_2$;
  else
    if $s_2 ==$ *"first"* **then**
      $z_0 = z_2$; $z_2 = z_3$;
    else
      if $s_1 ==$ *"first"* **then**
        if $s_2 ==$ *"second"* **then**
          $z_0 = z_1$; $z_4 = z_3$;
        else
          return z_2;
      else
        return z_2;
return z_2;

To define the semantic of $\mathcal{E}_t$ and $\mathcal{L}_t$ we will setup two parallel vanishing TSmSE, one for each query, using four vanishing SmSE. This *apparatus* is depicted in Figure 3.8.

In the example form Figure 3.8, the query z_1 takes the longest to arrive at a collecting box, for the case of a, but, for the case of b, is the first query to return an answer. Thus, the outcome of the experiment will be ("second","first"). To avoid a heavy notation, we will not index the outcomes in the vertex from which they were obtained. The first term in the outcome will refer to the vertex a and the second to the vertex b.

Definition 3.41. We say that $a\mathcal{E}_t b$ if, for every pair of dyadic rationals $z_1 < z_2$, running the protocol compare_IP(z_1, z_2) yields the same answer in time t. We say that $a\mathcal{L}_t b$ if there is a pair of dyadic rationals $z_1 < z_2$ such that, when we run the protocol compare_IP(z_1, z_2), we get one of the following results in time t: ("second","first"), ("second", "time out"), or ("time out","first").

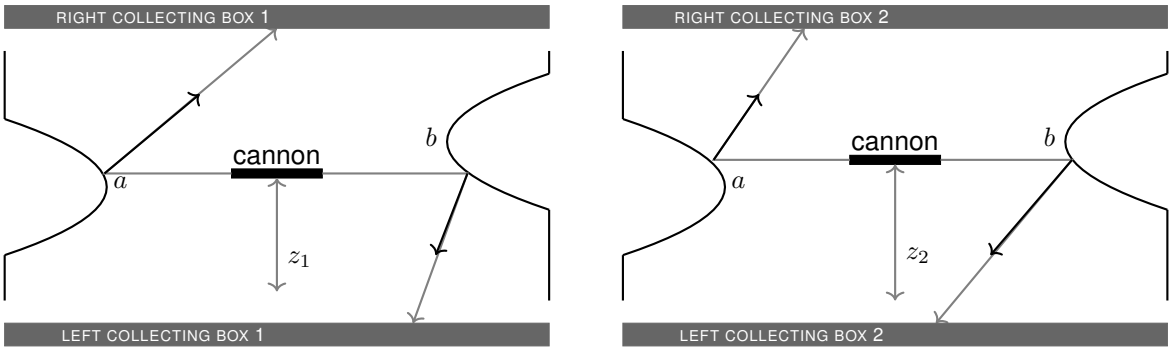

Figure 3.8: Schematic representation of the parallel vanishing TSmSE.

Proposition 3.42. If there is $t > 0$, such that $a\mathcal{L}_t b$ holds, then a is to the left of b.

Proof. Let (z_1, z_2) be the queries that witness the relation $a\mathcal{L}_t b$. If the result was ("second","first"), then a is closer to z_1 and b is closer to z_2; if the result was ("time out","first"), then b is farther from z_1 than a is; finally, if the outcome is ("second", "time out"), then a is farther from z_2 than b is. □

We will now prove that $\mathcal{E}_t$ and $\mathcal{L}_t$ are a *limit* timed equivalence relation.

Proposition 3.43. Given a time t and $a, b \in \mathcal{O}$, one and only one of $a\mathcal{L}_t b$, $b\mathcal{L}_t a$ and $a\mathcal{E}_t b$ will hold.

Proof. Suppose that $a\mathcal{E}_t b$ doesn't hold. Then, there is a pair of queries (z_1, z_2) that yield a different answer from a and b, whence one of the options that define the relations $a\mathcal{L}_t b$ or $b\mathcal{L}_t a$ will hold.

If $a\mathcal{E}_t b$ holds, then the other relations are trivially not verified. If $a\mathcal{L}_t b$ holds, then $a\mathcal{E}_t b$ is excluded by definition and $b\mathcal{L}_t a$ can't happen due to the fact that a is to the left of b (Proposition 3.42). □

Proposition 3.44. For every $t > 0$, $\mathcal{E}_t$ reflexive and symmetric.

Proof. See the proof of Proposition 3.19. □

Proposition 3.45. For every $t > 0$, $\mathcal{L}_t$ is transitive.

Proof. Suppose that $a\mathcal{L}_t b$ and $b\mathcal{L}_t c$, and let (z_1, z_2) be the witnesses of the relation $a\mathcal{L}_t b$. Then, if we setup the experiment with vertices a and c, (z_1, z_2) will witness $a\mathcal{L}_t c$ as well, since c could only be further away from a than b is. Suppose, for example, that the answer to compare_IP(z_1, z_2), between a and b, was ("time out","first"). Then, since c is to the right of b, the experimental time of shooting at z_1 towards c will be smaller than that of shooting at z_1 fired towards b. Thus, the outcome of compare_IP(z_1, z_2), between a and c, can only be ("time out","first") as well. □

Proposition 3.46. For every $t > 0$ and $a, b, c \in \mathcal{O}$, if $a\mathcal{L}_t c$ holds, there is a time T such that, if $a\mathcal{E}_T b$ holds, then $b\mathcal{L}_T c$ holds as well.

Proof. Suppose that $a\mathcal{L}_t c$ holds, whence, by Proposition 3.42, a is to the left of b. Let $T = T(k)$ be the precision required to fire both z_1 and z_2 in the interval $(a, (a+c)/2)$. With enough zeros padded, the answer from firing at c will be “first" and the answer from a will be “second". Thus, if $a\mathcal{E}_T b$, the answer from firing at b will also be “second", whence, if we setup the experiment with b and c and fire at z_1 and z_2, we will have the outcome (“second",“first"). Thus, if $a\mathcal{E}_T b$ holds, $b\mathcal{L}_T c$ will hold as well. □

Proposition 3.47. If $a\mathcal{L}_t b$ holds, for $t > 0$, there is an order T after which $a\mathcal{L}_{t'} b$ holds, for every $t' \geq T$.

Proof. If the relation $a\mathcal{L}_t b$ is witnessed by some queries (z_1, z_2), then a is to the left of b. Then, we can find two queries (z_1', z_2') within a and b, for which a sufficient padding will yield the answer (“second",“first") from an experimental call. This will also be true for any time above $T = T(|z_1'|) = T(|z_2'|)$. □

These proposition, together, imply the following result.

Proposition 3.48. The relations $\mathcal{E}_t$ and $\mathcal{L}_t$, as defined in 3.41, are a *limit* timed comparative concept.

Definition 3.49. We define the measurement map M as the function which, given a vertex y, successively runs Algorithm 3.2, with increasing input precision, and returns the limit of the sequence of generated dyadic rationals.

The approximations given by M, in this case, are given by the middle query z_2. Note that, if a vertex is in a dyadic position, it can be that the two queries will arrive at the exact same time to a collecting box. In this case, we could stop the algorithm and return an answer right away, but we will treat this event as if it where a time out and proceed with the measurement.

Proposition 3.50. The map M is a measurement map.

Proof. See the proof of Proposition 3.29. □

Proposition 3.51. The measurement map satisfies the separation property 3.11.

Proof. Suppose that $a, b \in \mathcal{O}$ are such that $M(a) < M(b)$.

Consider the case where b is a dyadic rational (the other case is solved analogously), and consider $z_1 \in (a, b)$. Then, since $M(a) < M(b)$, the result of firing at (z_1, z_2) towards a will have to be “second", for a z_2 equal to b with enough zeros padded. This pair (z_1, z_2) will be a witness of the relation $a\mathcal{L}_t b$, for $t = T(|z_1|) = T(|z_2|)$, since it will produce the outcome (“second",“time out").

Now consider the case where both a and b are non-dyadic and let $z_1 < z_2$ be dyadic rationals, such that $M(a) < z_1 < z_2 < M(b)$. Then, with $z_1' = z_1$ and $z_2' = z_2$, but padded with enough zeros, the result of firing at (z_1', z_2'), towards a and b, must be “second" and “first", respectively. These queries will be witnesses of the relation $a\mathcal{L}_t b$, for $t = T(|z_1'|) = T(|z_2'|)$. □

3.4.3.B Time-counting implementation

For this implementation, we have to use the assumption that it is possible for two experiments to consume the same number of machine steps. We thus introduce Protocol 3.9, which, for a time constructible schedule T, attempts to signal which query took the least amount of discrete time to return an answer.

PROTOCOL COMPARE_IP(z_1, z_2)

Receive as input two dyadic rationals $z_1 < z_2$ of size n;
Run an experiment with the query z_1 and wait constructive $T(n)$ units of time, counting the number of steps it takes to receive an answer;
Store this value in a variable T_1. If the result was a time out, let $T_1 = T(n) + 1$;
Do the same for the query z_2, defining a variable T_2;
If $T_1 < T_2$, return "first";
If $T_1 > T_2$, return "second";
If $T_1 = T_2 > T(n)$, return "time out";
If $T_1 = T_2 \leq T(n)$, return "indistinguishable".

Figure 3.9: Protocol for the time-counting implementation of the infinite precision vanishing type measurement.

If the result is "first", then y is closer to z_2 than to z_1; if it's "second", then y is closer to z_1 than to z_2; if it's "time out", then both z_1 and z_2 are too close to y to get an answer within the specified waiting time; finally, the outcome "indistinguishable" can occur either because the time of both experiments was the same, or because the difference between them was to small to be distinguished by the machines discrete measure of time.

As pointed out in [14], this implementation has a different notion of imprecision associated with it, which originates from the way we count time. If we don't make the assumption that all machine transitions take the same amount of physical time, then, when we count T_1 steps, the actual time taken for the experiment may not be in $(T_1 - 1, T_1]$. To formalize this consideration, a new kind of precision was defined: *time precision*. Given a map $g : \mathbb{N} \to \mathbb{N}$ and a query z, the number of machine transitions counted, when an experiment is ran with z, is a natural value T_1, uniformly sampled in $[\lceil t \rceil - g(|z|), \lceil t \rceil + g(|z|)]$, where t represents the actual experimental time. If $g(n) = 0$, we have a full precision, which is the only case that we will consider. The complexity classes from Table 3.2 were obtained using a time precision g, computable in polynomial time.

To measure with this implementation, we have to deal with the possibility of getting the answer "indistinguishable", which happens if the physical time of a machine's step is greater than the time difference between the two queries. To get around this problem, we will only consider time functions that satisfy the inequality $t'(z) \geq f(z)/(y - z)$, for some function f that increases as z approaches y.[11] We thus have the following result, which ensure that, if the two queries lie on the same side relative to the unknown vertex, the difference in times will increase as they approach the unknown vertex.

[11] Note that this condition is satisfied, for example, by the time form $t(z) = 1/|y - z|$, given in Proposition 2.24.

Proposition 3.52. Consider a SmSM with vertex y and queries z_1, z_2. Suppose that z_1 and z_2 are on the same side relative to y, that both $|y - z_1|$ and $|y - z_2|$ are bounded by 2^{-k} and that $|z_2 - z_1| = 2^{-k-2}$. Then, we have that $|t(z_2) - t(z_1)| \geq f(z_1)/4$.

Proof. Suppose that $z_1 < z_2 \leq y$ (the other case is analogous). Then, applying the mean value theorem, we have that, for some $\xi \in (z_1, z_2)$:

$$|t(z_2) - t(z_1)| = |z_2 - z_1||t'(\xi)| \geq \frac{|z_2 - z_1|}{|y - \xi|} f(\xi) \geq \frac{|z_2 - z_1|}{|y - z_1|} f(z_1) \geq \frac{2^{-k-2}}{2^{-k}} f(z_1) = \frac{f(z_1)}{4}$$

□

Thus, we know that, after some point, the answer “indistinguishable" will only occur when the wedge vertex is in between the two queries. To deal with the problem of obtaining “indistinguishable" in the first iterations, even when y is not within the queries, we can begin the measurement in a sub-interval of $(0, 1)$, small enough so that this does not happen. The specification of this sub-interval only requires a finite amount of information, so it can be hard-wired into the machine. We can thus measure with a variation of Algorithm 3.2, by replacing the first two if statements, $s_1 ==$ “second" and $s_2 ==$ “first", with $s_1 ==$ “second" or “indistinguishable" and $s_2 ==$ “first" or “indistinguishable", respectively.

To define the semantic of $\mathcal{E}_t$ and $\mathcal{L}_t$ we will setup a vanishing TSmSE using two vanishing SmSE. This means that both collecting boxes of the TSmSE work, but only to say which particle crossed them, not which collecting box was crossed.

Note that, at step k, we have y within an interval of size 2^{-k}, which we divide in four, thus having queries whose distance to each other is 2^{-k-2}. Given this fact, together with Proposition 3.52, and the subsequent discussion on hardwiring an interval into the machine, we will assume that the answer “indistinguishable" will only occur if the vertex is in between the two queries.[12] Moreover, it will only occur if the distance from the vertex to both queries is so similar, that the machine isn't able to tell them apart. This enables us to exclude the case where the answer is “indistinguishable", from one side, and “time out", from the other. If, for example, shooting at (z_1, z_2) towards a yields the response “indistinguishable" and z_2 produces a time out when shot towards b, then b is closer to z_2 than a is, whence the we must have the outcome “first" from firing at (z_1, z_2) towards b.

Definition 3.53. We say that $a\mathcal{E}_t b$ if, for every pair of dyadic rationals $z_1 < z_2$, running the protocol compare_IP(z_1, z_2) yields the same answer from both sides, in time t. We say that $a\mathcal{L}_t b$ if there is a pair of dyadic rationals $z_1 < z_2$ such that, when we run the protocol compare_IP(z_1, z_2), we get one of the following results in time t: (“second","first"), (“second", “time out"), (“second", “indistinguishable"), (“time out","first"), or (“indistinguishable","first").

[12] Note that we are essentially assuming that we have prior information about the position of each vertex, which allows us to specify the mentioned interval.

Proposition 3.54. If there is $t > 0$, such that $a\mathcal{L}_t b$ holds, then a is to the left of b.

Proof. We just have to justify the cases where we have “indistinguishable", since the others have been covered in Proposition 3.42. Let (z_1, z_2) be the queries that witness the relation $a\mathcal{L}_t b$. If the result was (“indistinguishable",“first"), then a's distance to both z_1 and z_2 is very similar, but b is closer to z_2 than to z_1; if the result was (“second", “time out"), then a is farther from z_2 than b is; if the result was (“second", “indistinguishable"), then b's distance to both z_1 and z_2 is very similar, but a is closer to z_1 than to z_2. In every case, we can conclude that a is to the left of b. □

Proposition 3.55. Given a time t and $a, b \in \mathcal{O}$, one and only one of $a\mathcal{L}_t b$, $b\mathcal{L}_t a$ and $a\mathcal{E}_t b$ will hold.

Proof. Suppose that $a\mathcal{E}_t b$ doesn't hold. Then, there is a pair of queries (z_1, z_2) that yield a different answer from a and b, whence one of the options that define the relations $a\mathcal{L}_t b$ or $b\mathcal{L}_t a$ will hold.

If $a\mathcal{E}_t b$ holds, then the other relations are trivially not verified. If $a\mathcal{L}_t b$ holds, then $a\mathcal{E}_t b$ is excluded by definition and $b\mathcal{L}_t a$ can't happen due to the fact that a is to the left of b (Proposition 3.54). □

Proposition 3.56. For every $t > 0$, $\mathcal{E}_t$ is reflexive and symmetric.

Proof. See the proof of Proposition 3.19. □

Proposition 3.57. For every $t > 0$, $\mathcal{L}_t$ is transitive.

Proof. Suppose that $a\mathcal{L}_t b$ and $b\mathcal{L}_t c$, and let (z_1, z_2) be the witnesses of the relation $a\mathcal{L}_t b$. Then, if we setup the experiment with vertices a and c, (z_1, z_2) will witness $a\mathcal{L}_t c$ as well, since c could only be further away from a than b is. Suppose, for example, that the answer to compare_IP(z_1, z_2), between a and b, was (“second",“indistinguishable"). Then, since c is to the right of b, the experimental time of shooting at z_1 towards c will be smaller than that of shooting at z_1 fired towards b. Thus, the outcome of compare_IP(z_1, z_2), between a and c, can only be (“second",“indistinguishable") or (“second",“first"), whence the pair (z_1, z_2) will be a witness of the relation $a\mathcal{L}_t c$, for $t = T(|z_1|) = T(|z_2|)$. □

Proposition 3.58. For every $t > 0$ and $a, b, c \in \mathcal{O}$, if $a\mathcal{L}_t c$ holds, there is an time T such that, if $a\mathcal{E}_T b$ holds, then $b\mathcal{L}_T c$ holds as well.

Proof. Suppose that $a\mathcal{L}_t c$ holds, whence, by Proposition 3.54, a is to the left of b. Then, there are dyadic rationals $z_1 < z_2$ in the interval $(a, (a + c)/2)$, for which firing towards c will yield the result “first" and firing towards a will result in “second". Thus, if $a\mathcal{E}_T b$, firing at b will also yield be “second", whence, setting up the experiment with vertices b and c, and queries z_1 and z_2, we will have the outcome (“second",“first"). □

Proposition 3.59. If $a\mathcal{L}_t b$ holds, for $t > 0$, there is an order T after which $a\mathcal{L}_{t'} b$ holds, for every $t' \geq T$.

Proof. If the relation $a\mathcal{L}_t b$ is witnessed by some queries (z_1, z_2), a is to the left of b. Then, we can find two queries (z_1', z_2') within a and b and with a sufficient padding of zeros, which will yield the answer ("second","first") from an experimental call. This will also be true for any time above $T = T(|z_1'|)$. □

These proposition, together, imply the following result.

Proposition 3.60. The relations $\mathcal{E}_t$ and $\mathcal{L}_t$, as defined in 3.53, are a *limit* timed comparative concept.

Definition 3.61. We define the measurement map M as the function which, given a vertex y, successively runs Algorithm 3.2, the time-counting version, with increasing input precision, and returns the limit of the sequence of generated dyadic rationals.

Proposition 3.62. The map M is a measurement map.

Proof. See the proof of Proposition 3.29. □

Proposition 3.63. The measurement map satisfies the separation property 3.11.

Proof. See the proof of Proposition 3.51. □

4

Measurable numbers

In 1986, Geroch and Hartle introduced the notion of a *measurable number*, which, as a computable number arises in mathematics, would arise from a physical theory (see [40]). These numbers were defined as the real values y, for which a technician, given an abundance of *unprepared raw materials*, and an allowed error ε, would be able to perform an experiment, under a finite set of instructions, yielding ultimately a rational number within ε of y.

Comparing the definitions of measurable and computable numbers, the *technician* is analogous to the *Turing machine*, the *instructions* to the machine's *code*, and the *abundance of unprepared raw materials* to demanding that the memory tape of the Turing machine should start blank.

However, the definition of a measurable number didn't take into account the *time* it would take for the experiment to run, which was only done by Beggs, Costa and Tucker in [17]. It also didn't require that an actual *measurement* be performed; one could ask the technician to build a computer and write a program to successively output the digits of any computable number.

4.1 Numbers that can be measured by the error-free SmSM

In this section we will present, in greater detail, the work done in [17], where the class of measurable numbers is defined with a Turing machine as the *technician*[1] and the two-sided error-free SmSE as the experiment. We will give a precise characterization of this set of numbers, explore its density in the interval $(0, 1)$ and provide two results regarding the undecidability of asserting whether or not a number is measurable.

4.1.1 A characterization of measurable numbers

We will define a measurable number as one that can be measured by the SmSM. Note that this is a universal definition, amongst measurement experiments defined with a schedule, regardless of the validity of the BCT conjecture. If a real number y is measurable by an experiment with a sub-exponential experimental time, should such an experiment exist, we can always measure y with the SmSM, using a greater schedule.

Definition 4.1. We say that a real number $y \in (0, 1)$ is *measurable* if there exists a SmSM M, with vertex y and a time constructible schedule T, such that M, knowing $n - 1$ digit of y, outputs its nth digit in time $T(n)$, i.e., without timing out.[2]

Recall the measurement maps from previous section (see, for example Definition 3.27), which were defined as an “extension" of a linear search algorithm.[3] To have a measurement map, according to Hempel, we had to allow them to keep running, even when the outcome of an experiment was a time out. This means that we have no way of knowing when we will actually attain another bit of the binary expansion of the wedge vertex; in the worst possible case, we can have a time out forever.

We thus make a distinction between the numbers that we can *obtain* using the SmSM, in the limit, and the numbers that we can *measure*. The difference lies in the fact that, in the second definition, we know how much time it will take to obtain the next digits of y. Note that, if the number of time outs is finite, we can replace the schedule with another one, still time constructible, such that no time out occurs.

We consider time constructible schedules, instead of just computable, to capture the idea of a physical clock that counts the time of an experiment as it’s being performed. As any computable function is bounded above by some time constructible one (see, for example, Lemma 2.3 of [7]), this assumption doesn’t restrict our class of measurable numbers.

Definition 4.2. A Turing machine M is said to be a universal measuring procedure for the SmSE if, for

[1] We thus encode, in our model of measurement, the recursive structure of experimental actions.

[2] We assume that, for each output z, an experiment is run with the cannon aimed at z. This is to ensure that the *measurement* comes from the physical experiment itself and not from a calculation done internally by the machine.

[3] We will drop the expression “extension" and just refer to the measurement map as linear search algorithm.

every measurable number y, there exists a time constructible schedule T, such that M, equipped with T, measures y.

Note the difference between this definition and the one from a measurable number. In Definition 4.1, every measurable number has an associated Turing machine and schedule that witnesses its measurability. In Definition 4.2, however, it is demanded that a single Turing machine measures every measurable real number, by *only* changing its schedule.

Remark 4.3. For the results that follow, it will be useful to consider the following presentation for a non-dyadic real $y \in (0,1)$, where $u_1 \geq 0$ and $u_i \geq 1$, for every $i \geq 2$:

$$y = 0 \cdot \underbrace{1 \ldots 1}_{u_1} \underbrace{0 \ldots 0}_{u_2} \underbrace{1 \ldots 1}_{u_3} \ldots \tag{4.1}$$

If y is a dyadic rational, this form is valid up to a certain point, after which all the digits of y are 0.

Proposition 4.4. Consider an error-free SmSM M, with a time constructible schedule T and unknown wedge vertex position at a non-dyadic y, written according to Pattern 4.1. If y is measurable by some program, then the sequence u_k is bounded by a computable function.

Proof. Define $a_k = u_1 + \cdots + u_k$. Then, the digit at the end of the block labelled by u_k is in the a_kth position. Denote by $y_k^{\pm}$ the first a_k digits of y, but one having the last digit equal to 0 and the other having the last digit equal to 1. Then, to determine all digits up to the a_kth digit, any program must have successfully run the experiment with a query z in the intervals $[y_k^-, y[$ and $]y, y_k^+]$.

If $y\downarrow_{a_k} = y_k^+$, then u_k is a block of 1's and $y - y_k^-$ has $a_k - 1$ zeros, followed by a one and the rest of y. Hence, $|y - y_k^-| \geq 2^{-a_k}$ and $|y - y_k^-| \leq 2^{-a_k+1}$. Analogously, we can prove a similar inequality for y_k^+, using the fact that the next block is a block of zeros. We then have:

$$\begin{aligned} 2^{-a_k} &\leq |y - y_k^-| \leq 2^{-a_k+1} \\ 2^{-a_{k+1}-1} &\leq |y - y_k^+| \leq 2^{-a_{k+1}} \end{aligned} \tag{4.2}$$

If, on the other hand, $y\downarrow_{a_k} = y_k^-$, we have the following inequalities:

$$\begin{aligned} 2^{-a_k-1} &\leq |y - y_k^-| \leq 2^{-a_k} \\ 2^{-a_{k+1}} &\leq |y - y_k^+| \leq 2^{-a_{k+1}+1} \end{aligned} \tag{4.3}$$

To determine the first a_k digits of y, the strictest bounds occur in the interval $]y, y_k^+]$. Since no time

out can ever occur, the schedule will have to, *at least*, satisfy the following (see inequalities 4.3 and 2.3):

$$T(a_k) \geq t(y_k^+) \geq \frac{A}{|y - y_k^+|} \geq A2^{a_{k+1}-1}$$

We then have:

$$\begin{aligned} 2^{u_{k+1}} = 2^{a_{k+1}-a_k} = 2^{-a_k}2^{a_{k+1}} \leq 2^{-a_k+1}T(a_k)/A \\ \Rightarrow u_{k+1} \leq \left\lceil \log\left(\frac{T(u_1 + \cdots + u_k)}{A}\right)\right\rceil - (u_1 + \cdots + u_k) + 1 \end{aligned}$$

This inequality establishes a computable relation between u_{k+1} and its previous terms, so u_k is bounded by a computable function. □

Proposition 4.5. Consider a SmSM with unknown wedge vertex position at a non-dyadic y, written according to Pattern 4.1. If the sequence u_k is bounded by a computable function, then y is measurable by the linear search algorithm.

Proof. Suppose that the sequence u_k is bounded by a computable function and, as in the proof of Proposition 4.5, define the sequence a_k. Then, since the strictest bound is in the interval $]y, y_k^+]$, when $y\downarrow_k = y_k^+$ (see Inequality 4.2), the longest experimental time is, at most, $C2^{a_{k+1}+1}$.

Since the sequence u_k is bounded by a computable function, so is the sequence a_k, whence we can find a schedule that will satisfy $T(k) \geq C2^{a_{k+1}+1}$. □

We then have the following result.

Proposition 4.6.

- A non-dyadic $y \in (0, 1)$, written according to Pattern 4.1, is measurable if and only if the sequence u_k is bounded by a computable function.
- The Turing machine equipped with the linear search algorithm is a universal measuring procedure.

4.1.2 Measurability, measure theory and decidability

In this section we examine the class of measurable numbers and show that, in the sense of *measure theory*, almost all numbers are measurable. We also show, however, that there is an uncountable number of non-measurable numbers and that measurable numbers are, in a finite amount of time, indistinguishable from non-measurable ones.

We start by introducing Algorithm 4.1, which, for input $k > 0$, attempts to find y to k binary places.

Proposition 4.7. Let $y \in (0, 1)$ and $k > 0$. Then, there is a set F_k, of measure 2^{-k}, such that, if $y \notin F_k$, the SmSM can measure y to k binary places.

Algorithm 4.1: Sweeping measurement algorithm.

Data: Positive integer k, representing the desired accuracy;
$p = 1$;
while $p \leq 2^k$ **do**
  $s =$ Prot_IP($p/2^k$);
  if $s == q_r$ **then**
    return $p/2^k$;
  $p = p + 1$;
return $p/2^k$;

Proof. Let $k \in \mathbb{N}$ and define the following set:

$$F_k = \bigcup_{0 \leq p \leq 2^k} \left[\frac{p}{2^k} - \frac{1}{2^{2k+1}} , \frac{p}{2^k} + \frac{1}{2^{2k+1}} \right] \cap (0,1)$$

Then, for every $y \notin F_k$, and $z = p/2^k$, for $1 \leq p \leq 2^k$, we have that $|z - y| \geq 1/2^{2k+1}$, so $t(z) \leq C2^{2k+1}$. Therefore, Algorithm 4.1, with input k and schedule $T(k) = D2^{2k+1}$, for $D \geq C$, will never produce a time out, whence it will be able to determine the first k binary places of y. The proof is completed by observing the following:

$$|F_k| = \sum_{0 \leq p \leq 2^k} \left(\frac{p}{2^k} + \frac{1}{2^{2k+1}} \right) - \left(\frac{p}{2^k} - \frac{1}{2^{2k+1}} \right) = \sum_{0 \leq p \leq 2^k} \frac{1}{2^{2k}} = \frac{1}{2^k}$$

□

Proposition 4.8. With probability 1, a real numbers in $(0,1)$ is measurable.

Proof. Define the sets $B = \bigcap_{n \geq 0} \bigcup_{k \geq 1} F_{n+k}$ and $A = (0,1) \setminus B$, where each F_{n+k} is defined as in the proof of Proposition 4.7. Since the measure of each F_{n+k} is 2^{-k-n}, the measure of $\bigcup_{k \geq 1} F_{n+k}$ is 2^{-n}, whence B is a measure zero set and A has *measure one*.

Now, let $y \in A$. Then there is $n \geq 0$ such that $y \notin F_{n+k}$, for every $k \geq 1$. Therefore, if we setup the SmSM with the schedule $T(k) = D2^{2(k+n)+1} = D2^{2k+(2n+1)}$, for $D \geq C$, Algorithm 4.1, on input k, will not fail to measure y to k binary places. Then, y is measurable by the program P, which, defined with the schedule $T(k) = D2^{2k+(2n+1)}$, successively calls Algorithm 4.1, with increasing precision k.[4] □

Note that the set A, from the proof of Proposition 4.8 does not contain any dyadic rationals. Indeed, let $a = p/2^r$ be a dyadic rational, with $0 < p < 2^r$, and let $k \geq 0$. If $k < r$, then $a \in F_{n+k}$, where $n = r - k$; if $k \geq r$, then $a \in F_k$.[5]

Proposition 4.9. There are uncountably many values $y \in (0,1)$ that are not measurable by the SmSM.

[4] It is important to note that, for the measurement to succeed, we would need to know *a priori* the order n, required to setup the appropriate schedule.

[5] For every $t \leq r$ and $q < 2^t$, it holds that $q/2^t \in F_r$. Indeed, by taking $p = q2^{r-t} < 2^r$, we have that $q/2^t = p/2^r \in F_r$.

Proof. Proposition 4.6 implies that any real number $y \in (0,1)$, written according to Pattern 4.1, such that the sequence u_k is not bounded by a computable function, is not measurable by any Turing machine using the SmSE as an oracle. Then, any $y \in (0,1)$, written according to Pattern 4.1, for which u_k is either $BB(k)$ or $BB(k+1)$, where BB is the *Busy Beaver* function (which is not bounded by any computable function – see Section A.4), is not measurable by the SmSM. The proof is completed by remarking that there is an uncountable number of real values in the above condition. □

We conclude by asserting the undecidability of the property of measurability.

Proposition 4.10. There is no program running on a Turing machine, using the SmSE as an oracle, that can decide, in a finite amount of time, if a number is measurable.

Proof. Recall that, by Proposition 4.6, a real number, written according to Pattern 4.1, is measurable if and only if u_k is bounded by a computable function.

In a finite amount of time, a SmSM can only find a finite number of digits of y. Then, a continuation of the already calculated digits, using the *Busy Beaver* construction from Proposition 4.9, is indistinguishable from a continuation that is bounded by a computable function. Thus, a finite number of digits is not enough to distinguish a measurable from a non-measurable number. □

4.2 Measurement as a means to classify real numbers

Consider a real number $y \in (0,1)$, written according to Pattern 4.1, and let $a_k = u_1 + \cdots + u_k$. From the proof of Propositions 4.4 and 4.5, we have:

- If y is measurable with a schedule T, the sequence u_k satisfies

$$u_{k+1} \leq \left\lceil \log\left(\frac{T(u_1 + \cdots + u_k)}{A}\right)\right\rceil - (u_1 + \cdots + u_k) + 1 \tag{4.4}$$

- If the sequence u_k is bounded by a computable function, y is measurable with a schedule satisfying

$$T(k) \geq C 2^{a_{k+1}+1}$$

This relationship allows us classify a real number according to its *measurement complexity*, by considering the time complexity of the schedule with which we can measure it (if it exists).

Recall Proposition 2.9, where we showed that, for a query z close enough to y, the duration of an experiment call satisfies the inequality

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

When considering the complexity class of a schedule, the constants A and C will become irrelevant, so we will set them to 1. The degree n would also introduce a multiplicative constant in the schedules, so we will set it to 2, thus considering that the experimental time is given by

$$t(z) = \frac{1}{|y - z|}$$

We can interpret this assumption as a classification of real numbers, when being measured with a scatter machine whose curve has the shape of a parabola, or forget about these physical interpretations and regard the conversion between the sequence u_k and the schedule T as an abstract operator.

We will characterize numbers that can be measured with two types of exponential schedules and prove that there is an infinite number of complexity classes, which make up the Grzegorczyk hierarchy, that are closed for the operation of, given a bound on u_k, obtaining a schedule to measure a vertex y. The reciprocal property is only achieved when we reach the class of primitive recursive functions.

To obtain a bound on the expansion of a vertex y, knowing the schedule with which we can measure it, we will consider a different way of constructing the schedule, than the form given in Proposition 4.5.

Suppose that we have a function g, which, for an input n, representing a binary place, returns the block u_k where n is. Then, since every binary place in the same block will yield the same experimental duration, we can measure y with any schedule satisfying $T(n) \geq 2^{a_{g(n)+1}+1}$.

However, since the sequence u_k is not necessarily computable, but bounded by a computable function, we might not be able to compute the function g, as defined above. Instead, we will consider that g gives the block where n would be, should each block be given by the computable function that bounds u_k. It's expression can be thus given by $g(n) = \lfloor b_n^{-1} \rfloor + 1$, where b^{-1} is the inverse of b, the function that bounds a, when its expression is thought of as that of a real valued function. We will thus use a schedule T such that, for each $k \geq 1$,

$$T(k) \geq 2^{b_{g(k)+1}+1} \tag{4.5}$$

Before moving on, we will present some results regarding time constructibility. Together with the examples from Section A.6, from the Appendix, these will imply the time constructibility of the schedules from the next section. The first two results can be found, for example, in [33], and the last one is an adaptation of Example 8 from [52].

Proposition 4.11. The class of time constructible functions is closed under composition.

Proof. Let M_f and M_g be two deterministic Turing machines that witnesses the time constructibility of two function f and g.

We start by constructing a machine M', which incorporates the finite control of M_g and M_f and contains an extra tape. It starts by simulating M_g, writing a symbol for every transition that it performs

(first, the symbol $\dot{0}$, and then a 0 for each transition).

When M' reaches the accepting state of M_g, the machine's head writes in the additional tape the last 0 and the simulation of M_f begins, giving it, as input, the sequence of $g(n)$ 0's, from right to left.

However, constructed as above, M' writes its $g(n)$ first steps, simulating M_g, and then $f(g(n))$ steps, by simulating M_f, i.e., it perform $g(n) + f(g(n))$ transitions. To solve this problem, we interchange the simulation of M_g with the simulation of M_f. When M_f is supposed to read the input, we simulate another step of M_g and account it a the input being given to M_f.

Thus altered, M' witnesses the time constructibility of the function $f \circ g$. □

Proposition 4.12. The class of time constructible functions is closed under sum and product.

Proof. Let M_f and M_g be two deterministic Turing machines that witnesses the time constructibility of two function f and g. In both cases, we construct a machine M' which writes, in a new tape, in parallel with the computation of the original machine, a copy of the input in the form $\dot{0}0\dots0$ of size n, where n is the size of the input.

In the case of addition, the accepting state of M_f is fused with the initial state of M_g. The simulation of M_g is done by reading $\dot{0}0\dots0$ as the input, from right to left. M' will thus perform $f(n)+g(n)$ transitions.

For the case of the product, note that $f(n) \times g(n) = f(n) + g(n) + (f(n) - 1) \times (g(n) - 1) - 1$. Thus, M' starts by calculating $f(n)$ and $g(n)$, in $f(n) + g(n)$ transitions, to two different tapes, marking the first and last three symbols in each of them. Then, it performs $(f(n)-1)\times(g(n)-1)$ transitions, by traversing the tape with $f(n) - 1$ symbols $g(n) - 1$ times, halting one transition before the end. □

Proposition 4.13. For $p \in (0, 1)$ a rational number and $c \geq 1$, the function $f(k) = k + c\lfloor k^p \rfloor$ is time constructible.

Proof. Let $p \in (0, 1)$ be a rational number and $c \geq 1$. Kobayashi's proof that $f(k) = k + \lfloor k^p \rfloor$ is time constructible makes use of Theorem 5.2, also from [52], and implies that $g(k) = k + h(\lfloor k^p \rfloor)$ is time constructible, when h is as well.[6] Then, to achieve the desired result, we only have to consider $h(n) = cn$, which, by Proposition A.11, from the Appendix, is time constructible. □

4.2.1 Measuring with an exponential schedule

4.2.1.A Real numbers with a polynomially bounded expansion

In this section we will characterize the complexity of measuring a real number, written according to Pattern 4.1, whose sequence u_k is bounded above by a polynomial.

[6] Note that, according to our definition, $h(n) = n$ is not, indeed, time constructible, since a machine requires, at least, $n + 1$ transitions to read the input and move to the accepting state. We can solve this problem by marking the last bit of the input, but this is not necessary, as the schedules we construct in the proof of Proposition 4.16 are built with a constant c greater than 1.

Proposition 4.14. Consider a real number $y \in (0,1)$, written according to Pattern 4.1. Then, the sequence u_k is bounded by a constant, if and only if y is measurable with a schedule $T(k) \in O(2^k)$.

Proof. "$\Rightarrow$"

Suppose that the sequence u_k is bounded by a constant $c \in \mathbb{N}$. Then, for every $k \geq 1$, $a_k \leq ck$, whence the function g is given by $g(k) = \lfloor k/c \rfloor + 1$. Then, given Inequality 4.5, fixing $d = 2c+1$, we can measure y with the following schedule:

$$T(k) = 2^{k+d} \geq 2^{c\lfloor k/c \rfloor + 2c + 1} = 2^{c(\lfloor k/c \rfloor + 2) + 1} = 2^{b_{g(k)+1}+1}$$

"$\Leftarrow$"

Now, if the vertex y is measurable with a schedule of the form $T(k) = 2^{k+d}$, for some constant d, then, given Inequality 4.4, the sequence u_k will satisfy

$$\begin{aligned} u_{k+1} &\leq \lceil \log\left(T(u_1 + \cdots + u_k)\right) \rceil - (u_1 + \cdots + u_k) + 1 \\ &= (u_1 + \cdots + u_k) + d - (u_1 + \cdots + u_k) + 1 = d + 1 \end{aligned}$$

which concludes the result. □

Note that, even with this restriction on the expansion of u_k, of it being bounded by a constant, we still need an exponential schedule to measure the vertex y.

Proposition 4.15. There is an uncountable numbers of values that can be measured with a schedule $T(k) \in O(2^k)$.

Proof. Consider the set of Cantor numbers, introduced in Section 2.2.3.A. Since, for any $y \in \mathcal{C}_3$, the sequence u_k is bounded above by $c = 4$, we can measure any number in $\mathcal{C}_3$ with a schedule of the form $T(k) = 2^{k+2\cdot 4+1} = 2^{k+9}$. □

Another way of obtaining this result is by using the fact that the distance by between a dyadic rational z, of size k, and any cantor number, is greater than 2^{-k-10} (Proposition 6.1 of [13]). We thus have that, for every query z, of size k, $t(z) \leq 1/|y - z| < 2^{k+10}$, which defines, once again, a schedule in $O(2^k)$.

We will now present two results, which, together with the previous one, characterize the schedules with which we can measure a real number with a polynomially bounded expansion.

Proposition 4.16. Let $y \in (0,1)$ be a real number, written according to Pattern 4.1. Then, if u_k is bounded by a polynomial of degree $m-1$, for $m \geq 2$, y is measurable with a schedule $T(k) \in O(2^{k+O(\lfloor k^{(m-1)/m} \rfloor)})$.

Proof. Suppose that the sequence u_k is bounded by a polynomial βk^{m-1}, for some constant $\beta \in \mathbb{N}$ and $m \geq 2$. Then, there is some $\alpha \in \mathbb{N}$, such that, for every $k \geq 1$,

$$a_k = u_1 + \cdots + u_k \leq \beta + \ldots \beta k^{m-1} \leq {}^{7}\alpha k^m$$

Thus, the function g is given by $g(k) = \lfloor \sqrt[m]{k/\alpha} \rfloor + 1$, so we can measure y with any schedule satisfying

$$\begin{aligned} T(k) &\geq 2^{b_{g(k)+1}+1} = 2^{\alpha(\lfloor \sqrt[m]{k/\alpha} \rfloor + 2)^m + 1} \\ &= 2^{\alpha \lfloor \sqrt[m]{k/\alpha} \rfloor^m + \alpha \sum_{i=0}^{m-1} \binom{m}{i} \lfloor \sqrt[m]{k/\alpha} \rfloor^i 2^{m-i} + 1} \end{aligned}$$

Note that, for any $1 \leq i \leq m-1$, $\lfloor \sqrt[m]{k/\alpha} \rfloor^i 2^{m-i}$ is bounded above by $\lfloor \sqrt[m]{k} \rfloor^{m-1} 2^{m-1}$. Then, by setting

$$c = \alpha 2^{m-1} \sum_{i=1}^{m-1} \binom{m}{i} = \alpha 2^{m-1}(2^m - 2)$$

and $e = 2^m + 1$, we can measure y with the following schedule:

$$T(k) = 2^{k + c\lfloor k^{1-1/m} \rfloor + e} \in O(2^{k + O(\lfloor k^{1-1/m} \rfloor)})$$

□

Proposition 4.17. If a real number $y \in (0, 1)$, written according to Pattern 4.1, is measurable with a schedule $T(k) \in O(2^{k+O(\lfloor k^{(m-1)/m} \rfloor)})$, for $m \geq 2$, the sequence u_k is bounded by a polynomial of degree $m - 1$.

Proof. Let $T(k) = 2^{k + c\lfloor k^{1-1/m} \rfloor + e}$, for some constants $c, e \in \mathbb{N}$ and $m \geq 2$, and let $q = (m-1)/m$. Then, setting $d = e + 1$ and $q = (m-1)/m$, the sequence u_k is bounded by the (real) sequence v_k, defined as

$$\begin{aligned} v_{k+1} &= \log\left(2^{k + c(v_1 + \cdots + v_k)^q + e}\right) - (v_1 + \cdots + v_k) + 1 \\ &= (v_1 + \cdots + v_k) + c(v_1 + \cdots + v_k)^q + e - (v_1 + \cdots + v_k) + 1 \\ &= c(v_1 + \cdots + v_k)^q + d \end{aligned}$$

Now, let $\alpha = 1/cm^m$, $k \geq 2$, and set $s_k = v_1 + \cdots + v_k$. We will consider two cases.

Suppose that $s_i \geq \alpha(i+1)^m$, for every $i \leq k-1$. Then, by the mean value theorem, there is $\xi_i \in (s_{i-1}, s_i)$, for each $i \geq 2$, such that,

$$\delta v_i = v_{i+1} - v_i = c(v_1 + \cdots + v_i)^q - c(v_1 + \cdots + v_{i-1})^q = cs_i^q - cs_{i-1}^q = cq\xi_i^{-1/m} v_i$$

[7] Summing over a polynomial of degree $m-1$ gives a polynomial of degree m.

Now, since $\ln(x+1) \leq x$, for any $x > -1$, we have that

$$\delta \ln(v_i) = \ln(v_{i+1}) - \ln(v_i) = \ln\left(\frac{v_{i+1}}{v_i}\right) = \ln\left(\frac{v_{i+1} - v_i}{v_i} + 1\right) = \ln\left(\frac{\delta v_i}{v_i} + 1\right) \leq \frac{\delta v_i}{v_i} = cq\xi_i^{-1/m}$$

Summing over the previous expressions, we get

$$\ln(v_{k+1}) - \ln(v_1) = \sum_{i=1}^{k} \delta \ln(v_i) \leq \sum_{i=1}^{k} cq\xi_i^{-1/m} \leq cq \sum_{i=1}^{k} \frac{1}{s_{i-1}^{1/m}}$$

Then, we have that

$$\begin{aligned}\ln(v_{k+1}) &\leq cq \sum_{i=1}^{k} \frac{1}{s_{i-1}^{1/m}} + \ln(d) \leq cq \sum_{i=1}^{k} \frac{1}{(\alpha i^m)^{1/m}} + \ln(d) = \frac{cq}{\alpha^{1/m}} \sum_{i=1}^{k} \frac{1}{i} + \ln(d) \\ &\leq^{8} \frac{c(m-1)}{m\alpha^{1/m}} (\ln(k)+1) + \ln(d) = (m-1)(\ln(k)+1) + \ln(d)\end{aligned}$$

whence

$$v_{k+1} \leq de^{(m-1)}k^{(m-1)}$$

Now suppose that there is some $i \leq k-1$, such that $s_i < \alpha(i+1)^m$. Then, since $s_{i+1} = s_i + v_i$, the value of s_{i+1} is smaller than it would be, if $s_i \geq \alpha(i+1)^m$. This will also be true for all the subsequent s_i's, for $i \leq k-1$. Thus, v_{k+1}, in this scenario, is necessarily smaller than the v_{k+1} from the previous case, which we already showed to be bounded by $de^{(m-1)}k^{(m-1)}$. □

Together, these propositions imply the following result.

Proposition 4.18. Let $y \in (0,1)$ be a real number, written according to Pattern 4.1, and $m \geq 1$. Then, $u_k \in O(k^{m-1})$, if and only if y is measurable with a schedule $T(k) \in O(2^{k+O(\lfloor k^{(m-1)/m} \rfloor)})$.

4.2.1.B Real numbers with an exponentially bounded expansion

In this section we will characterize the complexity in measuring a real number, written according to Pattern 4.1, whose sequence u_k is bounded above by an exponential function.

Proposition 4.19. Let $y \in (0,1)$ be a real number, written according to Pattern 4.1, and $\beta \geq 1$.

- If $u_k \in O(2^{\beta k})$, then y is measurable with a schedule $T(k) \in O(2^{2^{2\beta}k})$;
- if y is measurable with a schedule $T(k) \in O(2^{2^{\beta}k})$, then $u_k \in O(2^{\beta k})$.

[8] Recall that $\ln(k) \leq S_k \leq \ln(k)+1$, where S_k is the kth partial sum of the harmonic series.

Proof. For the first statement, suppose that $u_k \leq \alpha 2^{\beta k}$, for some constants $\alpha, \beta \in \mathbb{N}$. Then,

$$a_k \leq \alpha 2^{\beta} \left(1 + \cdots + 2^{\beta(k-1)}\right) = \frac{\alpha 2^{\beta}}{2^{\beta} - 1} \left(2^{\beta k} - 1\right)$$

In this case, the function g is given by

$$g(k) = \left\lfloor \log_2 \left(\frac{(2^{\beta} - 1)k}{\alpha 2^{\beta}} + 1 \right) / \beta \right\rfloor + 1$$

Then, we can measure y with any schedule satisfying

$$T(k) \geq 2^{b_{g(k)+1}+1} = 2^{\alpha 2^{\beta\left(\left\lfloor \log_2\left(\frac{(2^{\beta}-1)k}{\alpha 2^{\beta}}+1\right)/\beta\right\rfloor+2\right)}+1}$$

Therefore, we can use the following schedule to measure y:

$$T(k) = 2^{\alpha 2^{\beta\left(\log_2\left(\frac{(2^{\beta}-1)k}{\alpha 2^{\beta}}+1\right)/\beta+2\right)}+1} = 2^{\alpha 2^{2\beta}\frac{(2^{\beta}-1)k}{\alpha 2^{\beta}}+\alpha 2^{2\beta}+1} = 2^{2^{\beta}(2^{\beta}-1)k+\alpha 2^{2\beta}+1} \in O(2^{2^{2\beta}k})$$

For the second statement, let y be measurable with a schedule of the form $T(k) = 2^{2^{\beta}k+e}$, for some constants β and e. Then, the sequence u_k will bounded by v_k, defined by

$$\begin{aligned} v_{k+1} &= \left\lceil \log \left(2^{2^{\beta}(v_1+\cdots+v_k)+e}\right) \right\rceil - (v_1 + \cdots + v_k) + 1 \\ &= 2^{\beta}(v_1 + \cdots + v_k) + e - (v_1 + \cdots + v_k) + 1 \\ &= (2^{\beta} - 1)(v_1 + \cdots + v_k) + e + 1 \end{aligned}$$

Let $k \geq 1$, $d = e + 1$ and set $s_k = v_1 + \cdots + v_k$. Then,

$$v_{k+1} - v_k = (2^{\beta} - 1)s_k + d - (2^{\beta} - 1)s_{k-1} - d = (2^{\beta} - 1)v_k$$

Thus, $v_k = d2^{\beta(k-1)} \in O(2^{\beta k})$. □

We thus have the following characterization, which expresses the closure of the exponential class to the operations of obtaining a schedule and an expansion on u_k.

Proposition 4.20. Let $y \in (0, 1)$ be a real number, written according to Pattern 4.1. Then $u_k \in 2^{O(k)}$, if and only if y is measurable with a schedule $T(k) \in 2^{O(k)}$.

Moreover, Propositions 4.19 and 4.8 imply the following result.

Proposition 4.21. The set of real numbers in $(0, 1)$, written according to Pattern 4.1, whose binary expansion u_k is bounded by $(n + 1)2^k$, for some $n \geq 0$, is a measure one set.

Proof. By the proof of Proposition 4.8, a real number $y \in (0,1)$, with binary expansion given by u_k, can be measured with the schedule $T(k) = 2^{2k+2n+1}$ with probability one, for some $n \geq 0$. As in the proof of the second statement of Proposition 4.19, set $d = (2n+1)+1 = 2(n+1)$. Then, with probability one, the sequence u_k is bounded by $v_k = 2(n+1) \cdot 2^{k-1} = (n+1)2^k$, which concludes the result. □

4.2.2 The Grzegorczyk hierarchy

In this section we will present an infinite number of classes of functions, which make up the Grzegorczyk hierarchy, and prove that a sequence u_k, bounded by a function in the nth level of the hierarchy, implies a schedule, at most, in that same level. We will only list some properties of these classes of functions. The reader is directed to [34], [67], or [59], for a more detailed exposition to this topic.

In the base of the hierarchy we find the elementary functions, first introduced by Kálmar in 1943.

Definition 4.22. The class $\mathcal{E}$ of elementary functions is the smallest class containing zero, sucessor, projections and cut-off subtraction, which is closed under composition, bounded sum and bounded product.

Consider the m-times iterated exponential, denote by $2^{[m]}(x)$, defined recursively as $2^{[0]}(x) = x$ and $2^{[m+1]}(x) = 2^{2^{[m]}(x)}$. If $f(x)$ is an elementary function, then there is some $k \in \mathbb{N}$ such that, for every x, $f(x) \leq 2^{[k]}(x)$. Moreover, we can characterize the elementary functions as

$$\mathcal{E} = \bigcup_{n \geq 1} \mathsf{DTIME}(2^{[n]}(k)) = \mathsf{DTIME}(2^k) \cup \mathsf{DTIME}(2^{2^k}) \cup \cdots$$

Due to this fact, it is argued that $\mathcal{E}$ contains all effectively computable functions (see [25]).

To construct the Grzegorczyk hierarchy, originally defined in [43], we will consider the function $h_n(x)$, defined recursively as $h_0(x) = x+1$ and $h_{n+1}(x) = h_n^{[x]}(x)$.

Definition 4.23. For $n \geq 3$,[9] we define the nth level of the Grzegorczyk hierarchy, denoted by $\mathcal{E}^n$, as the smallest class containing zero, successor, projections, cut-off subtraction and the function h_{n-1}, which is closed under composition, bounded sum, and bounded product.

We list three properties of this hierarchy in the following proposition.

Proposition 4.24.

- Each level of the Grzegorczyk hierarchy is properly contained in the next one;
- the union of all the levels of the Grzegorczyk hierarchy gives the class PR, of primitive recursive functions;[10]

[9] Note that $h_0(x) = x+1$, $h_1(x) = 2x$ and $h_2(x) = x2^x$ are all elementary functions.
[10] This class of functions will be defined ahead.

- a function is in $\mathcal{E}^n$, if and only if it can be computed in time $\mathcal{E}^n$.

Given the closure properties that each level of the hierarchy satisfies, we have the following result.

Proposition 4.25. Let $n \geq 3$ and let $y \in (0,1)$ be a real number, written according to Pattern 4.1, whose sequence u_k is bounded by a function in $\mathcal{E}^n$. Then, y is measurable with a schedule in $\mathcal{E}^n$.

Proof. Suppose that u_k is bounded by a function in $\mathcal{E}^n$. Then, given that $\mathcal{E}^n$ is closed for bounded sum, the sequence b_k, that bounds above a_k, is also in $\mathcal{E}^n$, whence so is $T(k) = 2^{b_{k+1}+1}$. Thus, there is a schedule in $\mathcal{E}^n$ with which we can measure y. □

Recall the class of elementary functions. If one considers that $\mathcal{E}$ contains all effectively computable functions, then this result implies that, if u_k is bounded by an *effectively computable* function, then we can measure y with an *effectively computable* schedule.

Now, to introduce the class of primitive recursive functions, we need to introduce the operator of primitive recursion. Consider two functions g and h, of arity n and $n+2$, respectively. Then, we define a function f, through primitive recursion, as

$$\begin{cases} f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \end{cases}$$

We thus define the class of primitive recursive functions as the smallest class containing zero, successor and projections, which is closed for composition and primitive recursion.

Indeed, the class of primitive recursive function is closed for a sort of primitive recursion where f is defined to depend on all its previous values. The proof of closure, which can be found, for example, in Lemma 3.1.8 of [73], is done using the technique of course-of-values recursion.

Proposition 4.26. A real number $y \in (0,1)$, written according to Pattern 4.1, is measurable with a primitive recursive schedule if and only if the sequence u_k is bounded by a primitive recursive function.

Proof. The first implication is a consequence of Proposition 4.25 and the fact that each level of the Grzegorczyk hierarchy is contained in the class of primitive recursive functions.

For the second implication, suppose that y is measurable with a primitive recursive schedule T. Then, Inequality 4.4 imposes a primitive recursive bound on the sequence u_k. □

Note that the reasoning above does not apply for a specific layer of the Grzegorczyk hierarchy. Indeed, we have the following results.

Proposition 4.27. If a real number $y \in (0,1)$ is measurable with the elementary schedule $T(k) = 2^{2^k}$, then Inequality 4.4 *is not enough* to guarantee that the sequence u_k is bounded by an elementary function.

Proof. Given the schedule $T(k) = 2^{2^k}$, Inequality 4.4 implies that u_k is bounded by the sequence v_k, defined as

$$v_{k+1} = \left\lceil \log\left(2^{[2]}(v_1 + \cdots + v_k)\right)\right\rceil - (v_1 + \cdots + v_k) + 1 = 2^{v_1+\cdots+v_k} - (v_1 + \cdots + v_k) + 1$$

We will prove by induction that, for each $k \geq 2$, $v_k \geq 2^{[k-2]}(2)$. The inequality is trivially true for $k = 2$, so consider an arbitrary $k \geq 2$.

$$\begin{aligned} v_{k+1} &= 2^{v_1+\cdots+v_k} - (v_1 + \cdots + v_{k-1} + 2^{v_1+\cdots+v_{k-1}} - (v_1 + \cdots + v_{k-1}) + 1) + 1 \\ &= 2^{v_1+\cdots+v_k} - 2^{v_1+\cdots+v_{k-1}} \geq 2^{v_k} + 2^{v_1+\cdots+v_{k-1}} - 2^{v_1+\cdots+v_{k-1}} \\ &= 2^{v_k} \geq 2^{2^{[k-2]}(2)} = 2^{[k-1]}(2) \end{aligned}$$

Thus, since $h(k) = 2^{[k-2]}(2)$ is not an elementary function[11], so isn't the sequence v_k. Therefore, using only the relationship given by Inequality 4.4, we cannot guarantee that that there is elementary bound on the sequence u_k. □

Proposition 4.28. Let $y \in (0, 1)$ be a real number, written according to Pattern 4.1, whose sequence u_k is bounded by $2^{[k-2]}(2)$.[12] Then, y is measurable with a schedule $T(k) \in 2^{O(2^k)}$.

Proof. If u_k is bounded above by $2^{[k-2]}(2)$, then

$$a_k = u_1 + \cdots + u_k \leq 1 + 2^{[0]}(2) + \cdots + 2^{[k-2]}(2) =: b_k$$

First, we will show that $b_k \leq 2^{[k-1]}(2) - 1$. For $k = 2$ the inequality is trivially verified, so let $k \geq 2$. Then,

$$b_{k+1} = b_k + 2^{[k-1]}(2) \leq 2^{[k-1]}(2) - 1 + 2^{[k-1]}(2) = 2 \times 2^{[k-1]}(2) - 1 \leq 2^{[k]}(2) - 1$$

Let $k = b_m$, for some $m \geq 1$. Then,

$$\begin{aligned} 2^{b_{g(k)+1}+1} &= 2^{b_{m+1}+1} = 2^{b_m+2^{[m-1]}(2)+1} \leq 2^{2^{[m-1]}(2)+2^{[m-1]}(2)} \\ &= 2^{2\times 2^{[m-1]}(2)} = 2^{2\times 2^{2^{[m-2]}(2)}} \leq 2^{2\times 2^k} \end{aligned}$$

Note that, if k is such that $b_m \leq k < b_{m+1}$, then $g(k)$ will also be m, whence the above inequality will hold for any k. Thus, we can measure y with the schedule

$$T(k) = 2^{2\times 2^k} \in 2^{O(2^k)}$$

[10] This equality is only valid for $k \geq 2$, as v_0 is not defined.
[11] Recall the statements after Definition 4.22.
[12] For $k = 1$, set $2^{[k-2]}(2)$ to 1.

□

We can thus find an elementary schedule for which we cannot guarantee an elementary bound on the sequence u_k, and a non-elementary bound on u_k for which we cannot find an elementary schedule. This shows that the reciprocal result from the one in Proposition 4.25 is not verified for the class of elementary functions. It is also interesting to remark that, unlike the cases from section 4.2.1, in which the bound on u_k is smaller than the schedule which which we can measure the vertex, the bounds in the previous cases are greater than the schedules.

5

Conclusion

5.1 Summary

5.1.1 Computational results

Following the work from [5], we started this thesis by presenting a model of computation, the Smooth Scatter Machine (SmSM), in which a Turing machine is coupled with a physical experiment, the Smooth Scatter Experiment (SmSE), exchanging data with it according to a protocol, that determines the experimental precision, and a schedule, a time constructible function that determines the amount of time that the Turing machine waits for an answer from the physical experiment. This information exchange is akin to the one between a Turing machine and an oracle, the difference lying in the fact that, in this case, the answer from the physical oracle takes some amount of *physical time* to be obtained (see Section 2.2.1).

Considering three possible protocols, error-free, error-prone with unbounded precision and error-prone with fixed precision, we constructed an algorithm that the Turing machine could run to approximate the position of a wedge vertex (see Section 2.2.2), thus obtaining information to use as an advice to carry out some computational task. We then classified the class of sets that can be decided by

the SmSM, when clocked in polynomial time and with an exponential schedule, achieving a complete characterization when making an assumption on the schedule, for the error-free case (see Section 2.2.3.D), and on the physical duration of an experiment, for the error-prone cases (see Sections 2.2.3.E and 2.2.3.F).

The lower bounds were obtained by encoding the information given by an advice function in a real number, corresponding to the position of the wedge vertex (see Section 2.2.3.A), and the upper bounds were obtained by introducing the concept of a boundary number, which, when given to a Turing machine by means of an advice function, can be used to simulate (or approximate) the outcome of the physical experiment (see Section 2.2.3.B). We summarize the results obtained in Table 5.1. The first row represents the protocol that rules the data exchange.

	Infinite	Unbounded	Bounded
Lower Bound	$\mathtt{P}/\log\star$ Proposition 2.28	$\mathtt{BPP}/\log\star$ Proposition 2.33	$\mathtt{BPP}/\log\star$ Proposition 2.37
Upper Bound	$\mathtt{P}/\log\star$ Schedule in $\Omega(2^k)$ Proposition 2.31	$\mathtt{BPP}//\log^2\star$ Proposition 2.34	$\mathtt{BPP}//\log^2\star$ Proposition 2.38
Upper Bound Explicit time	-	$\mathtt{BPP}/\log\star$ Proposition 2.35	$\mathtt{BPP}/\log\star$ Proposition 2.39

Table 5.1: Computational results

Note that, if $\mathtt{P}$ is strictly contained in $\mathtt{BPP}$, the error in the cannon's precision leads to an increase in computational power (see Section 2.1.2), which happens because we can explore the error-prone protocols to simulate the tossing of a fair coin, thus being able to simulate probabilistic algorithms (see Sections 2.2.3.C and 2.1.1).

5.1.2 Fundamental measurement

Having established a measurement procedure, the linear search algorithm, we looked at the error-free SmSM from the perspective of *fundamental measurement*. We introduced the notion of a fundamental magnitude, which differs from a derived one in that it can be measured without the need for prior metrical concepts[1], and presented Campbell and Jeffreys' work from [27], on determining which quantities are fundamental, and Hempel's work from [44], on the axiomatization of a fundamental measurement procedure (see Section 3.1). If an attribute is extensive (see Section A.5), additive and independent, it is *amenable* to fundamental measurement. Introducing the notion of a *comparative concept*, represented by two relation, $\mathcal{E}$ and $\mathcal{L}$, which assert equality and innequality over objects of a domain $\mathcal{O}$, respectively,

[1] A derived magnitude, such as density, is obtained by first measuring other quantities, such as volume and mass, and performing some algebraic operation on the results.

we get a first workable definition of when a map from object to numbers, is a *measurement*. An application $M : \mathcal{O} \to \mathbb{R}$ is a measurement map if, for any objects $a, b \in \mathcal{O}$, $M(a) = M(b)$, whenever $a\mathcal{E}b$, and $M(a) < M(b)$, whenever $a\mathcal{L}b$ (see Section 3.1.2).

Now, recall that, in the case of the SmSM, we determine the outcome of an experiment by the observant of an event, which indicates that one quantity is greater than another (see Section 2.2.2). The absence of an event can either mean that the quantities being compared are equal, or that the duration of the experiment exceeded the time given by the machine's schedule. This notion of the *duration* of an experiment, which, taking the BCT conjecture, increases, at least, exponentially, as our approximations approach the unknown quantity we want to measure (see Sections 1.1 and 2.2.1), is not present in Hempel's axiomatization. Thus, after the work from [16], we presented a new axiomatization, which, considering the ability of an experiment to witness innequality between two objects, in time t, recovers Hempel's notion of a measurement procedure as time is allowed to approach infinity (see Sections 3.2 and 3.3).

Following [20], we introduced and exemplified three types of physical measurement: two-sided, if we can test when $x < y$ and $x > y$, one-sided, if we can only test whether $x < y$ or $x > y$, but not both, and vanishing, if we can only assert that $x \neq y$, but can't distinguish which is the greatest. We presented alterations that could be made to the error-free SmSM, so that it would represent each form of measurement, and proved that, in each case, our axiomatization could be satisfied (see Section 3.4).

5.1.3 Measurable numbers

Considering the two-sided error-free SmSM, we continued the work from [17], introducing the notion of a number that can be *obtained* and that of a number that can be *measured*. In the second case, we demand that a time constructible schedule exists, such that, when obtaining successive approximations of the position of the wedge vertex (or whatever object we are measuring), no time out occurs (see Section 4.1.1). We gave a precise characterization of these *measurable numbers*, presented two results regarding their density in the interval $(0, 1)$ and showed that, in a finite amount of time, a measurable number is indistinguishable from a non-measurable one (see Section 4.1.2). A real number in $(0, 1)$, written according to Pattern 4.1, is measurable if and only if the sequence u_k is bounded by a computable function. As a consequence, no dyadic rational can be measured by the SmSM. This makes our definition of measurability more strict than Geroch and Hartle's, for whom any computable number should be measurable. The difference lies in the fact that, in Geroch and Hartle's definition, the technician didn't actually have to *measure* an object; it was only required that he preformed an experiment, yielding ultimately an approximation of that number.

We finished by characterizing the measurement complexity of classes of real numbers, by determining the time complexity of the schedule with which we can measure them (see Section 4.2). We

considered real numbers with a polynomial and exponentially bounded expansion (see Section 4.2.1) and with an expansion bounded by some function in a given layer of the Grzegorczyk hierarchy (see Section 4.2.2). For a real number $y \in (0,1)$, written according to Pattern 4.1, we summarize the results obtained (Propositions 4.18, 4.20 and 4.26) in Table 5.2. The first row represents the order of the function bounding u_k; the second row represents the time complexity of the schedule with which we can measure the vertex y.

Bound on u_k	$O(k^{m-1})$	$2^{O(k)}$	PR
Time complexity of $T(k)$	$O(2^{k+O(\lfloor k^{(m-1)/m} \rfloor)})$	$2^{O(k)}$	PR

Table 5.2: Characterization of measurable numbers with a fixed schedule complexity.

Moreover, if u_k is in the nth level of the Grzegorczyk hierarchy, then y can be measured with a schedule, at most, in that same level. The reverse property is not verified, which we showed by giving an elementary schedule with which we can measure a real number, whose expansion is not bounded by any elementary function (see Section 4.2.2).

5.2 Future research

In this section we will present future topics of research, which arise from unresolved problems presented in this thesis.

5.2.1 Analogue-digital computation

It is still an open problem to achieve a complete characterization for the computational power of the SmSM, without making an assumption on the time complexity of the schedule, for the error-free case, and on the physical duration of an experiment, for the error-prone cases (see Sections 2.2.3.E and 2.2.3.F). These problems were posed, originally, in [13].

Recall that the results from Figure 5.1 are only applicable to measurement experiments, which gives a strong restriction on the behaviour of an analogue component (see, for example [84]), that interact with a Turing machine under specific protocols (these were studied in a more general setting in [10]). We argue that, in the end, reading the information from a physical experiment will always involve some form of measurement, which has to be accounted for when classifying the complexity of an algorithm (see, for example, the end of Section 2.2 of [22]), but we still don't have a full knowledge on how an analogue component might boost computation.

The third topic relates to the duration of a measurement experiment. Consider, for example, the experiment depicted in Figure 5.1, in which the cannon shoots a particle towards a metal rod with length y. If the cannon is able to shoot *adimensional* particles, we are able to tell, for a query z, in a fixed amount of time, whether $z > y$ or $z \leq y$. The experiment would then break the BCT conjecture. Moreover, it wouldn't fall into any of the three forms of physical measurement we presented, of signed, threshold, or vanishing type comparison. We can thus investigate what (possibly "unreasonable") assumptions can be made in order for the BCT conjecture to be violated and how the experiments we obtain fit into the axiomatization we have developed.

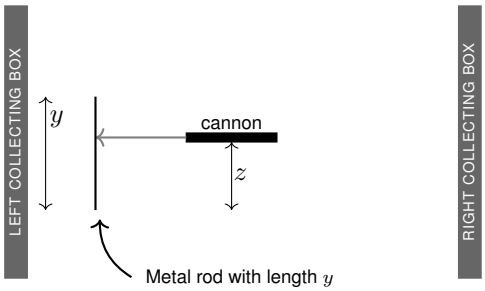

Figure 5.1: Schematic representation of the rod machine experiment.

5.2.2 Fundamental measurement

We still don't know if there is an intensive quantity that is amenable to fundamental measurement. Measurement, as we have constructed here (see the end of Section 3.1.1) seems to only apply to extensive quantities, but we don't know if this is a fundamental property, or just a consequence of our procedure. Hempel writes: "Obviously, an intensive characteristic (...) is not capable of fundamental measurement by reference to some mode of combination which is governed by simple theoretical principles; but it may well be amenable to some alternative type of fundamental measurement" (see [44]).

Regarding the time counting implementation of the vanishing type measurement, we made the assumption that we could hardwire into the machine some finite amount of information, which would allow us to get around the problem of getting the answer "indistinguishable" (see Section 3.4.3.B). We don't know if we can drop this assumption and still have a measurement satisfying our axiomatization, or if a generalization has to be made to also include the case of the time counting implementation.

Regarding measurement with errors, we don't know if it's possible to satisfy, in the limit, Hempel's axiomatization, when we use the error-prone protocols and, if not, if there is probabilistic theory of measurement[2] that the error prone cases can satisfy, or even a more general theory, that would encompass both the error-free and the error-prone cases.

[2] Probabilistic theories of measurement have been proposed, for example, in [38], [68] and [55].

5.2.3 Measurable numbers

First, we have not yet found a way to classify the measurement complexity of real numbers whose expansion is bounded by an arbitrary (possibly non-computable) function – we only achieved a complete characterization for polynomial, exponential and primitive recursive bounds (see Section 4.2). Such a classification would result in a hierarchy of real numbers, according to their measurement complexity. It would also enable the identification of complexity classes that are closed for the operations of obtaining a schedule from a bound and a bound from a schedule. We can see, from Table 5.2, that the class of exponential and of primitive recursive functions are examples of such classes, but we don't know if these are two of a kind of if more examples can be found. Next, although we showed that, with probability one, a real number has a measurement complexity bounded above by $O(2^{2k})$, we still have no idea behind measuring sets of real numbers with different measurements complexities. Still in the topic of measurement complexity, recall Proposition 4.28, where we gave an example of a vertex, with an expansion given by u_k, which we could measure with a schedule $T(k)$ smaller than u_k.[3] This seems to be because, as the duration of an experiment is the same for every binary places in the same block u_k, the schedule only has to grow fast enough to "catch up" to $2^{b_g(k+1)+1}$ in the beginning of a block (recall Inequality 4.5). However, we don't know if this property will be verified for greater bounds, or if we can find a smallest and greatest bound on u_k for which this "swap" is verified.

Secondly, is still unknown whether or not we can define the class of (non) measurable numbers in an inductive manner. We know, from [4], that this set is not closed, for example, under addition, so one would first have to find operations for which a closure property is verified.

The final topic is about the possibility of defining the "smallest" quantity that *cannot* be measured (or, analogously, the greatest quantity that can be measured). Recall that the real number y, whose expansion u_k is given by the busy beaver function, is not measurable by the SmSM (see Proposition 4.9). However, this would also be true if we considered an expansion of the form $u_k = BB(k) - 1$, so there are non-measurable real numbers with a smaller expansion than y's. Then, considering some ordering of functions, what is the smallest expansion u_k that yields a non-measurable number? The Busy beaver gives an upper bound for this expansion, but we still don't know if a minimum can be found. If it can, it would define a sort of *digital quantum*.[4]

[3] Recall that the k from u_k has a different meaning than the k in $T(k)$. In the first case, it refers to the kth block; in the second, to the kth binary place of the expansion of y.

[4] Quantum: "The minimum amount by which certain properties, such as energy or angular momentum, of a system can change. Such properties do not, therefore, vary continuously, but in integral multiples of the relevant quantum." (see [54])

Bibliography

[1] S. Aaronson. Complexity zoo. https://complexityzoo.net/Complexity_Zoo, 2012. Accessed: 2021-05-05.

[2] S. Aaronson. The Busy Beaver frontier. *ACM SIGACT News*, 51:32–54, 09 2020.

[3] L. M. Adleman and K. L. Manders. Reducibility, randomness, and intractability (abstract). In J. E. Hopcroft, E. P. Friedman, and M. A. Harrison, editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 151–163. ACM, 1977.

[4] T. Ambaram. Theory of two-sided experiments. Master's thesis, Instituto Superior Técnico, 2014.

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

[6] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[7] J. Balcázar, J. Días, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, 2nd edition, 1988, 1995.

[8] J. Balcázar and M. Hermo. The structure of logarithmic advice complexity classes. *Theoretical Computer Science*, 207(1):217–244, October 1998.

[9] J. Balcázar and U. Schöning. Logarithmic advice classes. *Theor. Comput. Sci.*, 99:279–290, 06 1992.

[10] E. Beggs, P. Cortez, J. F. Costa, and J. V. Tucker. Classifying the computational power of stochastic physical oracles. *Old City Publishing, Inc.*, 2018.

[11] E. Beggs, J. F. Costa, B. Loff, and J. V. Tucker. Computational complexity with experiments as oracles. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 464(2098):2777–2801, 2008.

[12] E. Beggs, J. F. Costa, B. Loff, and J. V. Tucker. Computational complexity with experiments as oracles II. Upper bounds. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 465(2105):1453–1465, 2009.

[13] E. Beggs, J. F. Costa, D. Poças, and J. V. Tucker. Oracles that measure thresholds: The Turing machine and the broken balance. *Journal of Logic and Computation*, 23(6):1155–1181, 2013.

[14] E. Beggs, J. F. Costa, D. Poças, and J. V. Tucker. Computations with oracles that measure vanishing quantities. *Mathematical Structures in Computer Science*, 2017.

[15] E. Beggs, J. F. Costa, D. Poças, and J. V. Tucker. An Analogue-Digital Church-Turing Thesis. *International Journal of Foundations of Computer Science*, 25(4):373–389, 2014.

[16] E. Beggs, J. F. Costa, and J. V. Tucker. Computational Models of Measurement and Hempel's Axiomatization. In A. Carsetti, editor, *Causality, Meaningful Complexity and Knowledge Construction*, volume 46 of *Theory and Decision Library A*, pages 155–184. Springer, 2010.

[17] E. Beggs, J. F. Costa, and J. V. Tucker. Limits to measurement in experiments governed by algorithms. *Mathematical Structures in Computer Science*, 20(06):1019–1050, 2010.

[18] E. Beggs, J. F. Costa, and J. V. Tucker. The Turing machine and the uncertainty in the measurement process. In H. Guerra, editor, *Physics and Computation, P&C 2010*, pages 62–72. CMATI – Centre for Applied Mathematics and Information Technology, University of Azores, 2010.

[19] E. Beggs, J. F. Costa, and J. V. Tucker. The impact of models of a physical oracle on computational power. *Mathematical Structures in Computer Science*, 22(5):853–879, 2012.

[20] E. Beggs, J. F. Costa, and J. V. Tucker. Three forms of physical measurement and their computability. *The Review of Symbolic Logic*, 7(4):618–646, 2014.

[21] E. Beggs and J. V. Tucker. Experimental computation of real numbers by Newtonian machines. *Proceedings of the Royal Society, Series A (Mathematical, Physical and Engineering Sciences)*, 463(2082):1541–1561, 2007.

[22] E. Blakey. Factorizing RSA keys, an improved analogue solution. *New Gener. Comput.*, 2009.

[23] M. Born and E. Wolf. *Principles of Optics. Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Pergamon Press, 1964.

[24] O. Bournez and A. Pouly. A survey on analog models of computation. *CoRR*, abs/1805.05729, 2018.

[25] W. S. Brainerd and L. H. Landweber. *Theory of computation*. John Wiley & Sons, Inc., 1974.

[26] N. R. Campbell. *Physics: The Elements*. Cambridge University Press, 1920.

[27] N. R. Campbell and H. Jeffreys. Symposium: Measurement and its importance for philosophy. *Proceedings of the Aristotelian Society, Supplementary Volumes*, 17:121–151, 1938.

[28] R. Carnap. *Philosophical Foundations of Physics*. Basic Books, 1966.

[29] P. Cockshott, L. Mackenzie, and G. Michaelson. Physical constraints on hypercomputation. *Theoretical Computer Science*, 394(3):159–174, 2008.

[30] S. A. Cook. The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[31] S. B. Cooper. *Computability Theory*. Chapman & Hall, 2004.

[32] B. J. Copeland and D. Proudfoot. Alan Turing's Forgotten ideas in Computer Science. *Scientific American*, 280(4):98–103, 1999.

[33] J. F. Costa. Turing machines as clocks, rulers and randomizers. *Boletim da Sociedade Portuguesa de Matemática*, 67:121–153, 2012.

[34] N. Cutland. *Computability*. Cambridge University Press, 1980.

[35] M. Davis. The myth of hypercomputation. In C. Teuscher, editor, *Alan Turing: the life and legacy of a great thinker*, pages 195–212. Springer, 2006.

[36] M. Davis. Why there is no such discipline as hypercomputation. *Applied Mathematics and Computation*, 178(1):4–7, July 2006.

[37] H. B. Enderton. *Computability theory: An introduction to recursion theory*. Academic Press, 2010.

[38] J.-C. Falmagne. A probabilistic theory of extensive Measurement. *Philosophy of Science*, 47(2):277–296, 1980.

[39] D. Fowler and E. Robson. Square root approximations in old babylonian mathematics: Ybc 7289 in context. *Historia Mathematica*, 25(4):366–378, 1998.

[40] R. Geroch and J. B. Hartle. Computability and physical theories. *Foundations of Physics*, 16(6):533–550, 1986.

[41] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6:675–695, 1977.

[42] O. Goldreich. *Computational Complexity*. Cambridge University Press, 2008.

[43] A. Grzegorczyk. Some classes of recursive functions. *Rozprawy Matematyczne*, 4, 1953.

[44] C. G. Hempel. *Fundamentals of Concept Formation in Empirical Science*. International Encyclopedia of Unified Science II, 7, 1952.

[45] C. G. Hempel. Fundamentals of concept formation in empirical science. *International Encyclopedia of Unified Science*, 2(7), 1952.

[46] F. Herbert. *Dune Messiah*. Putnam Publishing, 1969.

[47] D. Hilbert and W. Ackermann. *Principles of mathematical logic*, volume 69. Chelsea Publishing Company, 1950.

[48] S. Homer and A. L. Selman. *Probabilistic Complexity Classes*, pages 225–246. Springer US, 2011.

[49] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium onTheory of computing (STOC 1980)*, pages 302–309. ACM Press, 1980.

[50] V. J. Katz and A. Imhausen. *The Mathematics of Egypt, Mesopotamia, China, India, and Islam: A Sourcebook*. Princeton University Press, 2007.

[51] R. B. Kellogg. *CRC Standard Mathematical Tables and Formulae (Daniel Zwillinger,ed.)*, volume 38. Chapman & Hall, 1996.

[52] K. Kobayashi. On proving time constructibility of functions. *Theoretical Computer Science*, 35:215–225, 1985.

[53] P. Kropitz. Problém Busy Beaver. Master's thesis, Univerzita Karlova, Matematicko-fyzikální fakulta, 2011.

[54] J. Law and R. Rennie. *A dictionary of physics*. OUP Oxford, 2015.

[55] R. Michelini and G. Rossi. Measurement uncertainty: a probabilistic theory for intensive entities. *Measurement*, 15(3):143–157, 1995.

[56] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and Systems Science*, 13:300–317, 1976.

[57] J. W. Mills. The nature of the extended analog computer. *Physica D: Nonlinear Phenomena*, 237(9):1235–1256, 2008.

[58] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[59] P. Odifreddi. *Classical Recursion Theory II*. Studies in Logic and the Foundations of Mathematics. North Holland, 1999.

[60] T. Ord. Hypercomputation: computing more than the Turing machine. Master's thesis, The University of Melbourne, 2002.

[61] T. Ord and T. D. Kieu. On the existence of a new family of diophantine equations for ω. *Fundam. Informaticae*, 56(3):273–284, 2003.

[62] R. Penrose. *The Emperor's New Mind*. Oxford University Press, 1989.

[63] E. Post. Recursively enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society*, 50(5):284–316, 1944.

[64] E. Post. Degrees of recursive unsolvability. *Bulletin of the American Mathematical Society*, 54:641–642, 1948.

[65] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.

[66] T. Radó. On non-computable functions. *Bell System Tech. J.*, 41(3):877–884, 1962.

[67] H. Rose. *Subrecursion - function and hierarchies.* Oxford University Press, 1984.

[68] G. B. Rossi. A probabilistic theory of measurement. *Measurement*, 39(1):34–50, 2006.

[69] C. M. Rump. Strategies for rolling the Efron dice. *Mathematics Magazine*, 74(3):212–216, 2001.

[70] O. Shagrir. Supertasks do not increase computational power. *Natural Computing*, 11(1):51–58, 2012.

[71] H. T. Siegelmann. Computation Beyond the Turing Limit. *Science*, 268(5210):545–548, April 1995.

[72] M. Sipser. *Introduction to the Theory of Computation*. Thomson, Course Technology, 1996, 2006.

[73] C. Smorynski. The incompleteness theorems. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 821–865. Elsevier, 1977.

[74] R. I. Soare. Computability and recursion. *Bull. Symb. Log.*, 2(3):284–321, 1996.

[75] R. I. Soare. Turing computability and information content. *Philosophical Transactions of the Royal Society A*, 370:3277–3304, 2011.

[76] J. Stillwell. Emil Post and his anticipation of Gödel and Turing. *Mathematics Magazine*, 77(1):3–14, 2004.

[77] P. Suppes. A set of independent axioms for extensive quantities. *Portugaliæ Mathematica*, 10(2):163–172, 1951.

[78] A. Syropoulos. *Hypercomputation: computing beyond the Church-Turing barrier*. Springer Science & Business Media, 2008.

[79] C. Teuscher and M. Sipper. Hypercomputation: hype or computation? *Comunications of the ACM*, 45(8):23–24, 2002.

[80] P. Thagard. *Computational philosophy of science*. MIT press, 1993.

[81] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[82] A. Turing. On computable numbers. *Proceedings of the London Mathematical Society*, 43:544–546, 1937.

[83] A. Turing. *Systems of Logic Based on Ordinals*. PhD thesis, Princeton University, NJ, USA, 1939.

[84] B. Ulmann. *Analog and Hybrid Computer Programming*. De Gruyter Oldenbourg, 05 2020.

[85] T. van Gelder. What might Cognition be if not Computation? *The Journal of Philosophy*, 92, 07 1995.

[86] P. Verbaan. *The Computational Complexity of Evolving Systems*. PhD thesis, Utrecht University, Netherlands, 2006.

[87] R. Whyman. Physical computation, p/poly and p/log. In A. A. Abbott and D. C. Horsman, editors, *Proceedings of the 7th International Workshop on Physics and Computation, PC 2016, Manchester, UK, 14 July 2016*, volume 214 of *EPTCS*, pages 41–52, 2016.

A

Additional proofs

A.1 Probabilistic Trees

In this section we present the work from [10], regarding the relationship between a probabilistic tree and an error-prone SmSM. The idea is that a run over an error-prone SmSM can be represented by a rooted tree, where each edge represents a deterministic part of the computation and each vertex represents the execution of a query over the SmSE. As the oracle is stochastic, there is a probability associated to each outcome of the oracle. Thus, we can assign probabilities to the edges of the query tree. The main result of this section, proposition A.5, will tell us how much we have to bound the edge difference between two probabilistic trees, so that their respective acceptance probabilities aren't too far apart.

Definition A.1. A probabilistic query tree is a rooted tree (V, E, ν), where:

- V is the set of vertices, representing the configurations of a SmSM during the query state;
- E is the set of edges, representing deterministic parts of the computation;
- $\nu \in V$ is the configuration of the machine in the first time it reaches the query state.

Vertices with no children represent accepting or rejecting configurations.

Definition A.2. An m-ary query tree is a probabilistic query tree where each inner node has exactly m children. Note that the behaviour of an SmSM can be modeled by a 3-ary query tree, since, after the shooting state, there are only three possible options: q_r, q_l, or q_t. We denote by $T_{m,n} = (V_{m,n}, E_{m,n}, \nu_{m,n})$ an m-ary probabilistic query tree with depth n.

Let $L_{m,n} \subset V_{m,n}$ be the set of inner nodes of $T_{m,n}$. Every $u \in L_{m,n}$ has exactly m children $u_1, \ldots, u_m$, to which it is connected through m edges, $e_1, \ldots, e_m$. Probability assignments for the the edges of a probabilistic query tree $T_{m,n}$ are total functions $\sigma : E_{m,n} \to [0,1]$, such that, for every $u \in L_{m,n}$, the probability of the computations following to one of it's children adds up to 1, i.e., $\sigma(e_1) + \cdots + \sigma(e_m) = 1$.

The set of all assignments over $T_{m,n}$ is denoted by $\rho(T_{m,n})$. We denote by $T^{\sigma}_{m,n}$ a query tree $T_{m,n}$ with assignment σ. If π is a path over $T^{\sigma}_{m,n}$, the probability of the SmSM following this path is given by the product of the probability of each edge, i.e., $P(\pi) = \Pi_{i=1}^{n} \sigma(\pi[i])$, where $\pi[i]$ is the ith edge of the path π. The sum over the probabilities of each accepting path gives the acceptance probability of the m-ary tree with depth n and assignment σ. We denotes this probability by $P(T^{\sigma}_{m,n})$.

Definition A.3. Take $m, n \in \mathbb{N}$ and $\sigma_1, \sigma_2 \in \rho(T_{m,n})$. The maximum distance between the two probabilistic query trees $T^{\sigma_1}_{m,n}$ and $T^{\sigma_2}_{m,n}$ is defined as $D(\sigma_1, \sigma_2) = \max\{|\sigma_1(e) - \sigma_2(e)| \, : \, e \in E_{m,n}\}$.

Definition A.4. We will denote by $\mathcal{A}_m(n,s)$ the maximum possible difference between acceptance probabilities of two m-ary trees with depth n, whose maximum distance is bounded above by, s, i.e., $\mathcal{A}_m(n,s) = \max\left\{|P(T^{\sigma_1}_{m,n}) - P(T^{\sigma_2}_{m,n})| \, : \, \sigma_1, \sigma_2 \in \rho(P(T_{m,n})) \wedge D(\sigma_1, \sigma_2) \leq s\right\}$.

Proposition A.5. For any $m, n \in \mathbb{N}$ and $s \in [0,1]$, $\mathcal{A}_m(n,s) \leq (m-1)ns$.

Proof. The proof follows by induction on n. For $n = 0$, the tree has depth zero, which means that the machine does not consult the oracle. We then have that $P(T^{\sigma}_{m,0}) = P(T^{\sigma'}_{m,0}) = 0$, if the leaf in rejecting, and $P(T^{\sigma}_{m,0}) = P(T^{\sigma'}_{m,0}) = 1$, if it is an accepting one. Therefore, $\mathcal{A}_m(n,s) = 0$.

Now suppose the result is true for n and consider the probabilistic tree $T_{m,n+1}$ with m outgoing edges $e_1, \ldots, e_m$ and depth $m+1$. Each edge e_i is incident in a node $T_{m,n}(i)$, for $i = 1, \ldots, m$ respectively. Consider probability assignments $\sigma, \sigma' \in \rho(T_{m,n+1})$ such that $D(\sigma, \sigma') \leq s$. We then have:

$$P(T^{\sigma}_{m,n+1}) = \sigma(e_1)P(T^{\sigma}_{m,n}(1)) + \cdots + \sigma(e_m)P(T^{\sigma}_{m,n}(m))$$
$$P(T^{\sigma'}_{m,n+1}) = \sigma'(e_1)P(T^{\sigma'}_{m,n}(1)) + \ldots \sigma'(e_m)P(T^{\sigma'}_{m,n}(m))$$

As $\sigma(e_m) = 1 - \sigma(e_1) - \cdots - \sigma(e_{m-1})$, we can write:

$$\begin{aligned}
|P(T^{\sigma}_{m,n+1}) - P(T^{\sigma'}_{m,n+1})| &= |\sigma(e_1)P(T^{\sigma}_{m,n}(1)) + \cdots + \sigma(e_m)P(T^{\sigma}_{m,n}(m)) \\
&\quad - \sigma'(e_1)P(T^{\sigma'}_{m,n}(1)) - \cdots - \sigma'(e_m)P(T^{\sigma'}_{m,n}(m))| \\
&= |\sigma(e_1)P(T^{\sigma}_{m,n}(1)) + \cdots + (1 - \sigma(e_1) - \cdots - \sigma(e_{m-1}))P(T^{\sigma}_{m,n}(m)) \\
&\quad - \sigma'(e_1)P(T^{\sigma'}_{m,n}(1)) - \cdots - \sigma'(e_m)P(T^{\sigma'}_{m,n}(m))| \\
&= |\sigma(e_1)(P(T^{\sigma}_{m,n}(1)) - P(T^{\sigma}_{m,n}(m))) \\
&\quad + \cdots + \sigma(e_{m-1})(P(T^{\sigma}_{m,n}(m-1)) - P(T^{\sigma}_{m,n}(m))) + P(T^{\sigma}_{m,n}(m))) \\
&\quad - \sigma'(e_1)P(T^{\sigma'}_{m,n}(1)) - \cdots - \sigma'(e_m)P(T^{\sigma'}_{m,n}(m)| \\
&= |(\sigma(e_1) - \sigma'(e_1))(P(T^{\sigma}_{m,n}(1)) - P(T^{\sigma}_{m,n}(m)) \\
&\quad + \cdots + (\sigma(e_{m-1}) - \sigma'(e_{m-1}))(P(T^{\sigma}_{m,n}(m-1)) - P(T^{\sigma}_{m,n}(m)) \\
&\quad + P(T^{\sigma}_{m,n}(m)) + \sigma'(e_1)(P(T^{\sigma}_{m,n}(1)) - P(T^{\sigma'}_{m,n}(1)))) \\
&\quad + \cdots + \sigma'(e_{m-1})(P(T^{\sigma}_{m,n}(m-1)) - P(T^{\sigma'}_{m,n}(m-1)) \\
&\quad - (\sigma'(e_1) + \cdots + \sigma'(e_{m-1}))P(T^{\sigma}_{m,n}(m)) - \sigma'(e_m)P(T^{\sigma'}_{m,n}(m))|
\end{aligned}$$

Now, as $\sigma'(e_1) + \cdots + \sigma'(e_{m-1}) = 1 - \sigma'(e_m)$, we have:

$$\begin{aligned}
|P(T^{\sigma}_{m,n+1}) - P(T^{\sigma'}_{m,n+1})| &= |(\sigma(e_1) - \sigma'(e_1))(P(T^{\sigma}_{m,n}(1)) - P(T^{\sigma}_{m,n}(m))) \\
&\quad + \cdots + (\sigma(e_{m-1}) - \sigma'(e_{m-1}))(P(T^{\sigma}_{m,n}(m-1)) - P(T^{\sigma}_{m,n}(m))) \\
&\quad + \sigma'(e_1)(P(T^{\sigma}_{m,n}(1)) - P(T^{\sigma'}_{m,n}(1)))) \\
&\quad + \cdots + \sigma'(e_m)((P(T^{\sigma}_{m,n}(m)) - P(T^{\sigma'}_{m,n}(m)))|
\end{aligned}$$

Using the induction hypothesis, and since the difference of the probabilities lies within $[-1, 1]$ and each $|\sigma(e_i) - \sigma'(e_i)|$ is bounded by s, we conclude that:

$$\begin{aligned}
P(T^{\sigma}_{m,n+1}) - P(T^{\sigma'}_{m,n+1})| &\leq |\sigma(e_1) - \sigma'(e_1)| + \cdots + |\sigma(e_{m-1}) - \sigma'(e_{m-1})| \\
&\quad + \sigma'(e_1)\mathcal{A}_m(n, s) + \cdots + \sigma'(e_m)\mathcal{A}_m(n, s) \\
&\leq (m-1)s + \mathcal{A}_m(n, s) \leq (m-1)s + (m-1)ns = (m-1)(n+1)s
\end{aligned}$$

Hence, $\mathcal{A}_m(n+1, s) \leq (m-1)(n+1)s$. □

A.2 Random sequences

In Section 2.2.3.C we proved that the error-prone SmSM can be used to simulate the tossing of a biased coin. In this section, we show that this is enough to simulate the tossing of a fair one, up to a given probability of error.

Proposition A.6. Given a biased coin with probability of heads $q \in (\delta, 1-\delta)$, for some $0 < \delta < 1/2$, and a real number $\gamma \in (0,1)$, we can simulate, with a probability of failure smaller than γ, a sequence of independent fair coin tosses of length n, by doing a linear number of biased coin tosses.

Proof. The method to simulate one fair toss is to toss the biased coin twice. If the result is HT, we output H; if the result is TH, we output T; otherwise, we toss the coin twice again and repeat the process. The probability of the process halting in one step is $r = q(1-q) + (1-q)q = 2q(1-q)$ and the probability of having to toss again is $s = 1 - r$. This process is repeated until one of the first two cases occur, and then the whole method is repeated n times. We will denote the total number of tosses by T_n, which is a random variable given by negative binomial distribution, with mean[1] $\mu = ns/r + n = n/r$ and variance $\sigma^2 = ns/r^2$.

Now, using Chebyshev's inequality, with $t = \eta n/\sigma$, for some value η, we have:

$$P(|T_n - \mu| \geq \eta n) \leq \frac{\sigma^2}{\eta^2 n^2} = \frac{s}{\eta^2 n r^2} \leq \frac{1}{\eta^2 n r^2}$$

If we take $\eta \geq 1/r\sqrt{\gamma}$, we will have that $P(|T_n - \mu| \geq \eta n) \leq \gamma/n \leq \gamma$, for any n. Then, T_n will only be greater than $\mu + \eta n$ with probability bounded above by γ, whence, up to a probability of failure of γ, the total number of times the algorithm runs is

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

Since $r = 2q(1-q)$, and, for each time the algorithm runs, we toss a coin twice, we get that the total number of coin tosses is

$$2 \cdot \frac{n}{2q(1-q)}\left(1 + \frac{1}{\sqrt{1-\gamma}}\right) = \frac{n}{q(1-q)}\left(1 + \frac{1}{\sqrt{1-\gamma}}\right)$$

which is linear in n. □

[1] We add n to the mean of T_n for the variable to count the time until n successful tosses, not just the standard waiting time until success.

A.3 Error propagation

In this section we present two results regarding error propagation, which are used when proving the upper bounds for the SmSM (see Section 2.2.3).

Proposition A.7. Let x be a quantity we measure and Δx the absolute error associated with its measurement. We then have:

- If $x = a \pm b$, $|\Delta x| \leq |\Delta a| + |\Delta b|$;
- if $x = a \times b$, $|\Delta x| \leq a|\Delta b| + b|\Delta a| + |\Delta a \Delta b|$.

Proof. Suppose that $x = a + b$. Then,

$$\begin{aligned} x + \Delta x = (a + \Delta a) + (b + \Delta b) = (a + b) + (\Delta a + \Delta b) \\ \Rightarrow \Delta x = \Delta a + \Delta b \\ \Rightarrow |\Delta x| \leq |\Delta a| + |\Delta b| \end{aligned}$$

Now suppose that $x = a \times b$. Then,

$$\begin{aligned} x + \Delta x = (a + \Delta a)(b + \Delta b) = ab + a\Delta b + b\Delta a + \Delta a \Delta b \\ \Rightarrow \Delta x = a\Delta b + b\Delta a + \Delta a \Delta b \\ \Rightarrow |\Delta x| \leq a|\Delta b| + b|\Delta a| + |\Delta a \Delta b| \end{aligned}$$

□

A.4 Busy Beaver

In this section we present a non-computable function, the Busy Beaver, first introduced by Radó in 1962 (see [66]), which is used in section 4.1.2, when proving the existence of non-measurable numbers. The interested reader is directed to [2], for a further discussion on the *Busy Beaver*.

We define the *Busy Beaver* function as the one which, given a non-zero input n, returns the biggest finite number of transitions that a Turing machine with n states can perform, when running on input 0. We will denote this function by BB and define $BB(0) = 0$.

Proposition A.8. The *Busy Beaver* is not computable.

Proof. Suppose that BB is a computable function. Then, we can decide if a Turing machine M with n states halts on input 0, by simulating it for $BB(n)$ steps. We will prove that this implies that we can solve

the halting problem.

Given a machine M and an input w, we can create another machine M_w, which, on input 0, simulates the behaviour of M on input w. Then, denoting by $|M_w|$ the number of states of M_w, we can use Algorithm A.1 to decide the Halting set. Therefore, since the Halting set is undecidable, the *Busy Beaver* cannot be a computable function. □

Algorithm A.1: Decider for the Halting set

Input: z;
if z *is not the pairing of a specification of a Turing machine* M *and a word* w **then**
  Reject z;
Assemble M_w and run it for $BB(|M_w|)$ steps;
if M_w *is in a halting state* **then**
  Accept z;
else
  Reject z;

Proposition A.9. The *Busy Beaver* is not bounded above by any total computable function.

Proof. If there existed such a total computable function f, we could remake Algorithm A.1, using f instead of BB, for the number of transition to run. This would give a computable decider of the Halting set. □

In fact, it can be proven that the *Busy Beaver* dominates every total computable function. Consider, for example, that the current champion for $BB(6)$, when defined using 1-tape, 2-symbol Turing machines, was found by Kropitz in 2011 (see [53]), who presented a 6-state Turing machine that runs for more than 7.4×10^{36534} steps, a value greater than the estimated number of protons in the observable universe.

A.5 Extensive quantities

"Philosophers have divided quantities (...) into two kinds. Intensive quantities are those which can merely be arranged in a serial order; extensive quantities are those for which a natural operation of addition or combination can also be specified" (see [77]).

In this section we will present Patrick Suppes' axiomatization for extensive quantities. The proof that the quantities in this thesis, such as mass and length, satisfy this axiomatization is straightforward.

To axiomatize extensive quantities, Suppes considers an ordered triple $\langle K, Q, * \rangle$, where K is an non-empty set of object with some attribute, Q is a binary relation that compares objects according to that attribute and $*$ is a binary function that combines two objects in K into one.

Definition A.10. We say that $\langle K, Q, * \rangle$ is a system of extensive quantities if the following axioms are verified:

- If $x, y \in K$, then $x * y \in K$;
- if $x, y, z \in K$, xQy and yQz, then xQz;
- if $x, y, z \in K$, then $(x * y) * zQx * (y * z)$;
- if $x, y, z \in K$ and xQy, then $x * zQz * y$;
- if $x, y \in K$ and not xQy, then there is a $z \in K$ such that $xQy * z$ and $y * zQx$;
- if $x, y \in K$, then not $x * yQx$;[2]
- if $x, y \in K$ and xQy, then there is a positive integer n such that $yQnx$.[3]

In the case of mass, for example, we can define K as the *toolbox of standards*, $*$ as the “gluing" of two objects and Q as $\mathcal{L}$, the relation that asserts that an object has less mass than another. Thus defined, the triple $\langle K, Q, * \rangle$ is a system of extensive quantities. Moreover, mass is clearly an additive and independent property (see Section 3.1.1).

A.6 Time constructible functions

In this section we will show how to construct deterministic Turing machines that witness the time constructibility of two functions and of their composition. These constructions, which serve to provide an intuition for Proposition 4.11 and 4.12, rely on “dance" cycles, in which the machine traverses the input from left to right and right to left as many times as necessary, to halt after the right amount of transitions.

Recall that, by definition, we only have to prove that the machine halts in the right amount of transitions for inputs of size bigger than a given order.

Proposition A.11. For $c \geq 2$, the function $g(k) = ck$ is time constructible.

Proof. We present the following Turing machine which, for any input of size $k \geq c$, halts in exactly ck steps. Between p_1 and p_{k-1}, we repeat the transition from p_{k-2} to p_{k-1}.

The Turing machine has three tapes: the first receives the input; the second is a working tape, which will do the machine’s “dances"; the third tape, also a working tape, will store the value of c.

The value stored in the third tape, before entering the dance cycle, is, in reality, $c - 1$, and not c. This is because part of the first k transitions of $ck = (c - 1)k + k$ is used to read the input.

Thus, the machine writes $k - 1$ symbols in the third tape, while it copies the input to the second tape,[4]

[2] This condition excludes elements with measure 0. For the case of mass, we are excluding the existence of an object with zero weight.

[3] For $x \in K$, we define $1x := x$ and $nx := (n - 1)x * x$, for an integer $n > 1$.

[4] This is the part where it is required that the input size is larger than c.

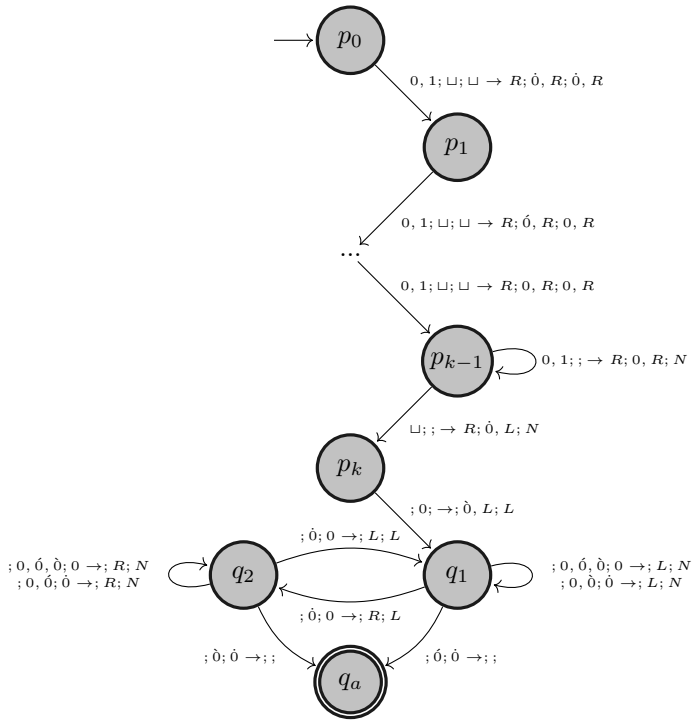

Figure A.1: Clock for the function $g(k) = ck$.

which takes up $k + 1$ transitions, and then enters the dance cycle, moving the reading head of the third tape one step to the left every time it completes a dance.

Each cycle, except the last, consumes k transitions, so, when the machine enters the last cycle, it will have performed $k+1+(c-2)k = (c-1)k+1$ transitions. In the last cycle, due to the marking of the penultimate symbol, which is either $\grave{0}$ or $\acute{0}$, the machine only performs $k - 1$ transitions, which makes up a total of $(c-1)k+1+k-1 = ck$ transitions overall. □

Proposition A.12. The function $f(k) = 2^k$ is time constructible.

Proof. To prove the result, we present the following Turing machine:

In this case, we have setup the Turing machine for it to halt in 2^k transitions, for any k. The particular cases of $k \in \{0, 1, 2\}$ are handled by the states p_0 to q_1.

For inputs of size greater or equal than 3, the machine performs a cycle of $k - 2$ steps. Note that, for $k \geq 2$, we have that

$$2^k = 2^0 + 2^1 + \sum_{i=2}^{k-1} 2^i + 1$$

The first two terms account for the transitions from p_0 to q_1; the last term accounts for the transition to the accepting state.

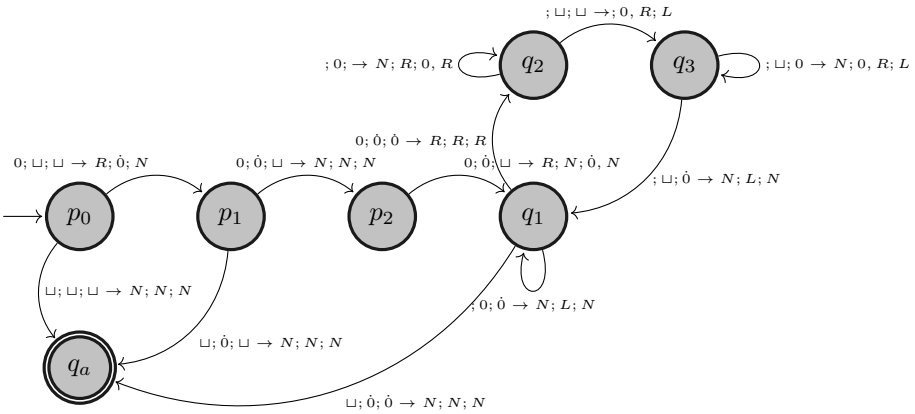

Figure A.2: Clock for the function $f(k) = 2^k$.

The sum is calculated by the cycle. The machine copies the second tape into the third (rewriting the values written in it) and then copies the third tape to the right of the last symbol in the second one, thus duplicating, in each step of this cycle, the previous value.

The halting condition is the reading of the first cell with $\sqcup$ in the input tape, simultaneously with the reading of $\dot{0}$ in the second and third tapes. Before the step is executed for the first time, two of the symbols of the input have already been read.

Consider, for example, that the machine is run with input 3. Before the cycle, 3 transitions are performed and all the symbols of the input are read; when the machine reaches the state q_1, the beginning of the cycle, there is a $\dot{0}$ in each working tape; during the cycle a 0 is written to the right of the $\dot{0}$, in the second tape, and its reading head goes back 2 steps; overall, the cycle performs 4 transitions. When the cycle ends, the reading head is in the beginning of the second tape, marked with a $\dot{0}$, and 1 transition to the accepting state is performed. The machine thus performs a total of 8 transitions.

□

Proposition A.13. For $c \geq 1$, the function $h(k) = 2^{ck}$ is time constructible.

Proof. For $c = 1$, the result is reduced to the previous proposition. For $c \geq 2$ we will follow the steps given in Proposition 4.11 to construct a machine that witnesses the time constructibility of $f \circ g = h$. Let M_g and M_f be the Turing machines given in the previous propositions and consider a Turing machine M', with 7 tapes. M' starts by simulating M_g, writing a symbol in the last two tapes for each transition that M_g performs (first a $\dot{0}$ in each tape and then 0's for each transition). While this simulation is going on, M' simulates M_f, from p_1 to q_1 to setup the conditions to simulate the exponential. Now, each time the simulation of M_f is in q_1, M' simulates another step from M_g and performs a dance of M_f. When this process ends, M' performs an extra 3 transitions.

M' will thus simulate the exponential as if it had received as input ck, the number of transition that

M_g performs, which, since the transitions from p_0 to q_1 are performed with the simulation of M_g, amounts to $2^{ck} - 3$ steps. With 3 final transitions M' performs a total of $h(k)$ steps. □

A.7 Analogue computation

Throughout this thesis we considered only the analogue computation performed by a measurement experiment. In this section we provide a small example of an analogue computer that is capable of performing integration. The interested reader is directed to [84] for a further reading on analogue and hybrid computation.

An analogue computer is one whose structure creates an *analogy* of a given problem. The oldest known device of this sort is the Antikythera mechanism, built around 100 B.C., and discovered in 1900, which produced a mechanical analogue for the study of celestial mechanics. In the early 20th century, analogue computers, such as the Oslo Analyzer, built in 1938, were developed to perform integration and solve differential equations. Figure A.3 (adapted from [84]) contains a schematic depiction of a simple mechanical integrator.

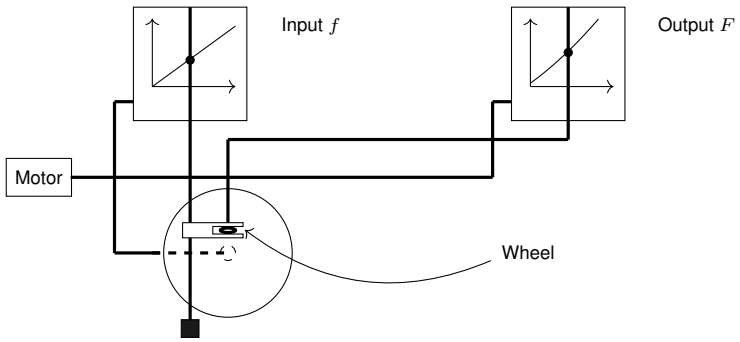

Figure A.3: A simple mechanical integrator.

The mechanism contains a motor that drives all the mechanical parts. A function f is read in the input panel and its vertical component determines the radial position of a wheel, placed on the surface of a rotating disk. If we set the disk to have angular velocity ω, the velocity of the wheel, at time τ, is then given by $f(\tau) \cdot \omega$. The vertical component of $F(x)$ is determined by the turning distance of the wheel, so, in the interval $[0, T]$, under ideal conditions, the device implements the operation

$$\omega \int_0^T f(\tau) d\tau$$

Analogue computation thus provides an efficient way of performing tasks that might be complicated to carry out analytically, which gives a very interesting aspect to explore in *hybrid computation*: the ability to divide a task in two parts; one to be solved by the analogue and the other by the digital component.

Index