TÉCNICO LISBOA

Space Bounded Scatter Machines

João Alves Alírio

Dissertation to obtain the Master of Science Degree in

Mathematics and Applications

Supervisor: Prof. José Félix Gomes da Costa

Examination Committee

Chairperson: Prof. Maria Cristina Sales Viana Serôdio Sernadas
Supervisor: Prof. José Félix Gomes da Costa
Members of the Committee: Prof. André Nuno Carvalho Souto
and Prof. João Filipe Quintas dos Santos Rasga

May 2017

Acknowledgments

I would like to thank my supervisor, Professor José Félix Costa, for introducing me to the subject of Complexity Theory, for all his help and guidance in the elaboration of this work, for his availability and all the hours spent explaining and clarifying several issues, but mostly for all his patience and tolerance through the whole time of the work.

I would also like to thank my family for all the support and help, and my friends who accompanied me through the work and helped and supported me.

Thank you all.

Abstract

This dissertation concerns the computational power of an abstract computation model when bounded in polynomial space. The computation model in study consists of a Turing machine coupled with a physical experiment as oracle (the scatter experiment). This computation model (the scatter machine) is already studied concerning polynomial time restrictions, from where we adapted most of the proof techniques used to this work.

We understood for which functions we are able to build a clock (a specific Turing machine) in polynomial space. There are clocks in polynomial space for functions with at most an exponential growing rate. We also presented a new communication protocol (the generalized protocol) between the analogue and the digital components of the scatter machine which, in this case, allow us to use the proper computational power of the scatter experiment as an oracle.

We introduce a uniform complexity class, which we use as support class for one non-uniform complexity class needed to describe the computational results we obtained. We established the computational results of the scatter machine bounded in polynomial space, for both the sharp wedge and the smooth wedge cases, using in each case both the standard and the generalized communication protocols. We established the results for the three usual precision assumptions (infinite precision, arbitrary precision and fixed precision), and in some cases of the smooth wedge experiment we obtain different results depending on if we use the time schedule (clock to bound the time running of the experiment) or not.

Keywords: Analogue-digital computation; Physical oracle; Non-uniform complexity; Space bounded clocks; Hypercomputation.

Resumo

Neste trabalho estudamos o poder computacional de um modelo de computação abstracto quando limitado em espaço polinomial. O modelo de computação em estudo consiste numa máquina de Turing acoplada a uma experiência física como oráculo (a experiência de dispersão). Este modelo de computação (a máquina de dispersão) já está estudado em relação a restrições em tempo polinomial, de onde adaptamos a maioria das técnicas de prova usadas neste trabalho.

Averiguamos quais seriam as funções para as quais podemos construir um relógio (uma máquina de Turing específica) em espaço polinomial. Existem relógios em espaço polinomial para funções com um crescimento no máximo exponencial. Apresentamos também um novo protocolo de comunicação (o protocolo generalizado) entre os componentes analógico e digital da máquina de dispersão que, neste caso, permite usar todo o poder computacional da experiência de dispersão como oráculo.

Introduzimos uma classe de complexidade uniforme, que usamos como classe de suporte para uma classe de complexidade não uniforme necessária para descrever os resultados computacionais obtidos. Estabelecemos os resultados computacionais da máquina de dispersão limitada em espaço polinomial para ambas, a parede com vértice e para a parede diferenciável, utilizando em cada caso, ambos os protocolos de comunicação padrão e generalizado. Estabelecemos os resultados para os três casos de precisão usuais (precisão infinita, precisão ilimitada e precisão fixa), e em alguns casos da experiência com parede diferenciável obtivemos resultados diferentes dependendo de se usamos o relógio temporizador (relógio para limitar o tempo de execução da experiência) ou não.

Palavras-chave: Computação analógico-digital; Oráculo físico; Complexidade não uniforme; Relógios limitados no espaço; Hipercomputação.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A Turing machine is an abstract device introduced by Alan Turing around 1936 (see [22]). Its aim was to model in a formal way the idea of computation. There are other models of computability, as lambda calculus for example, but the Turing machine is until today the standard model in Theory of Computation. A Turing machine consists of a finite set of infinite tapes (tapes with an infinite amount of cells) together with a finite control which dictates how the machine behaves. We usually have two common types of Turing machines, the decision Turing machines and the computation Turing machines. We say that a decision Turing machine decides a set, if it accepts or rejects the input word when the word belongs or not to the set, respectively. A computation Turing machine computes a function, if it writes in the output tape the word that the function gives for the specified input word (the words can represent numbers). Throughout this dissertation we are most interested in the decision Turing machines.

Each Turing machine is specified to work with a finite set of symbols, its alphabet plus the blank symbol (to denote that a tape cell is empty). The blank symbol is the only symbol allowed to occur infinitely often in any of the tapes, since the input is finite and any other symbol appears only if written by the Turing machine, which will have performed a finite amount of transitions. The Finite control consists of a finite set of states and a transition function, which based on the current state and what is written in the tapes dictates to which state the Turing machine transits and what it writes and where to moves in any of its tapes. The Turing machine must have an initial state, where the computation begins, and an input tape, where the input is written (the input can be empty). The tapes of a Turing machine are usually used to store the input, perform general scratch computations and, when needed, write the output, which leads us to some distinguished tapes, as the input or the output tapes. In the finite control we also have some distinguished states, as the initial state and the accepting and rejecting states (or general halting states).

Based on this model, a definition of what is computable was given. Usually, a set or a function is said to be computable if there is a Turing machine that decides or computes it, respectively. We could also say it to be Turing-computable. There are, in general Mathematics, non-computable objects as sets and functions. With that understanding, strongest versions of computability started to be presented in the twentieth century, giving arise to the hypercomputation. It is an interesting philosophical question if whether the human brain is computable or not, i.e. if it can be simulated by a Turing machine. Besides Philosophy, these subjects also arise interesting questions in both Physics and Mathematics. Here we attempt to study a computation model consisting of a Turing machine coupled with a physical experiment. This model is an analogue-digital machine (or hybrid machine), with its digital part being the Turing machine and its analogue part being the physical experiment.

The hybrid machine in study through the dissertation is a particular kind of a more embracing set of Turing machines, the oracle Turing machines (see [3, 2]). Usually an oracle is a set of words (a

language) and a Turing machine needs one distinguished tape and three distinguished states to interact with it. The query tape and the query, yes and no states. An oracle Turing machine is a Turing machine that can write a word in the query tape and when it performs a transition to the query state is instructed to perform another transition either to the yes or the no state, depending on if the query word belongs or not to the oracle, respectively. On the computation model in study the oracle is a physical experiment, for which we may need some additional states or tapes to interact with it. On an usual oracle when reaching the query state the Turing machine performs exactly one transition to the answer from the oracle. Using a physical experiment as an oracle, a Turing machine needs to wait for conclusions from the experiment which may take more time than one transition.

Throughout this dissertation we are interested in a specific experiment, the scatter experiment (introduced in [14]), which aims to measure the position of a vertex in a wedge. The scatter experiment, as well as other physical experiments, take an intrinsic time until achieve its conclusions. Thus, we must be able to measure time on the digital component of the machine if we want to distinguish results obtained by the hybrid machine allowing different running times for the experiment. A Turing machine to consult the experiment as an oracle needs to set some parameters of the experiment, which could be made with more or less precision. We study, as is usually done in the subject, three different assumptions on the precision with which the parameters could be set, and define then the different kinds of hybrid machines we can get, the scatter machines (introduced in [6]).

Considering different kinds of transition functions we get different kinds of Turing machines. We have deterministic and non-deterministic Turing machines which are distinguished by the nature of their transition functions, and as part of the non-deterministic Turing machines we have the well known probabilistic Turing machines (see [3, 2]). A deterministic Turing machine always follows the same behaviour for any time it is in the same state reading the same in all the tapes while a non-deterministic Turing machine can have different behaviours originated by the same situation. The non-deterministic Turing machines have an external component which randomly chooses between two or more options, in the probabilistic case the external component chooses between two hypothesis with equal probability, $1/2$ each. With different precisions to set the experiment we can also get deterministic and probabilistic scatter machines, where their different nature comes from the experiment which can be deterministic or probabilistic.

Our aim is to study the scatter machine concerning the complexity of its computations. Complexity Theory, in general, studies the resources needed to perform computations, unlike Theory of Computation, which aims to determine what it is computable or not. Here we aim to understand what can be computed with some specific restrictions to the resources of the machine. On Turing machines the bounds on resources are usually concerning time and space for the computations, over the input size. In the physical experiment we try to cover most of its possible restrictions, as time to perform the experiment or precision on which we could set the experiment.

Due to the nature of the scatter machine we need to introduce, besides the basic complexity classes, the non-uniform complexity classes (see [3]). This classes of languages are distinguished from the basic classes because the algorithms to decide them have access to an exterior amount of information which depends on the input size. So, they have at least the particularity of being able to be decided with a different algorithm for each input size, this point of view clears a relation of this classes with Boolean circuits (see [3]). They have actually most relevant issues as the access to non-computable information.

The scatter machines obtain their non-computable information on the position of the vertex, if the vertex is described by a non-computable number. So, in order to have a scatter machine computing beyond the power of a Turing machine we are assuming the existence of real numbers in the considered physical reality. There is a part of the scientific community which believes that there is no hypercomputability, since we would need to make a non-computable parameter so we could use it then to perform

some kind of computation (see [17, 18]). But even if that is true, there is always the interest to understand better the physical systems (if they are in fact on real numbers) which can be living beings or of another kind.

1.1 Our work

The scatter experiment was introduced in [14] and scatter experiments as oracles to Turing machines clocked in polynomial time were studied in [6], [7] and [12], inter alia. Through this dissertation we attempt to study the scatter machine bounded in polynomial space.

We have three basic components constituting a scatter machine, the analogue part (the physical experiment used as an oracle), the digital part assigned to perform the digital computations and the digital part used to rule the physical experiment in matter of running time bounds (see [12]). In the time restriction cases we could also have a digital component to bound the time of the whole computation, or in this context we could consider a digital component to restrict the space allowed to use in the tapes. But in both cases we can, in general, guarantee the accomplishment of the restrictions without an integrand digital component on the analogue-digital machine.

In a analogue-digital machine we must deal with an important issue on its definition, the communication protocol between the analogue and the digital parts. In the context of this dissertation, we noticed that the previously considered protocols could artificially restrict the power of the analogue-digital machine, since here we are considering space restrictions instead of time restrictions to the analogue-digital machine. We then consider a different communication protocol aiming to access the full power of the physical oracle and we study the complexity results on the space restricted analogue-digital machine for both cases, the standard and the generalized protocols. We consider both protocols with any of the three precision assumptions (infinite, arbitrary and fixed precisions). As a corollary of Proposition 2.3.2:

Proposition. *A set $A \in \Sigma^*$ is decidable by a generalized scatter machine which is not space restricted if and only if is decidable by a standard scatter machine.*

We can conclude that a scatter machine clocked in polynomial time decides the same sets with both the standard and the generalized protocols.

We consider two different cases of the scatter experiment, the sharp and the smooth cases, which are distinguished by the nature of their wedge. For which we establish the complexity results of the analogue-digital machine bounded in polynomial space. With the smooth wedge case the experiment takes an intrinsic time to achieve its conclusions and we must use the digital part to rule the time bound for the physical experiment, in order to guarantee that the experiment do not keeps running indefinitely. The digital component aimed to bound the running time of the oracle calls is called time schedule. As we are interested in the study of analogue-digital machines bounded in polynomial space, we must understand which are the time functions that we are able to use for the purpose of bounding the running times of the experiment with this restriction. We conclude that we can use a time function with at most an exponential growing rate (Proposition 2.1.2):

Proposition. *In polynomial space, any clock ticks at most an exponential amount of transitions.*

And that we can actually use exponential time functions within polynomial space (Proposition 2.1.1):

Proposition. *There exists a clock bounded in polynomial space such that, for an input of size n, ticks an exponential amount of transitions on n.*

We determine the lower and upper bounds for the scatter machine bounded in polynomial space, with both the standard and the generalized communication protocols. For that purpose we used some

proof techniques already used to the polynomial time restriction adapted to our case. To state our results we use the already known class $PSPACE/poly$ and another non-uniform complexity class based on the bounded error probabilistic class (as BPP for the polynomial time restriction) for the polynomial space restriction. The class is $BPPSPACE//poly$ and the complexity results with the standard communication protocol are the ones which follow (respectively, Theorem 3.6.5, 3.6.7, 3.6.9, 3.7.5, 3.7.7 and 3.7.9):

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-free ShSM in polynomial space if and only if $A \in PSPACE/poly$.*

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision ShSM in polynomial space if and only if $A \in BPPSPACE//poly$.*

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision ShSM in polynomial space if and only if $A \in BPPSPACE//poly$.*

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-free SmSM in polynomial space, using an exponential time schedule, if and only if $A \in PSPACE/poly$.*

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision SmSM in polynomial space, using an exponential time schedule, if and only if $A \in BPPSPACE//poly$.*

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision SmSM in polynomial space, using the time schedule, if and only if $A \in BPPSPACE//poly$.*

(Note that ShSM stands for sharp scatter machine and SmSM stands for smooth scatter machine, the two different cases of the experiment) With the generalized communication protocol for the smooth scatter machine we obtained different results depending on if we use the time schedule or not. The results for the generalized sharp scatter machine are (respectively, Theorem 4.1.4, 4.1.5 and 4.1.7):

Theorem. *The sets decidable by an error-free ShSM in polynomial space are all the sets $A \in \Sigma^*$.*

Theorem. *The sets decidable by an error-prone arbitrary precision ShSM in polynomial space are all the sets $A \in \Sigma^*$.*

Theorem. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision ShSM in polynomial space if and only if $A \in BPPSPACE//poly$.*

The results for the generalized smooth scatter machine, using the time schedule, are exactly the same as the results for the standard smooth scatter machine (see Theorems 4.2.5, 4.2.7 and 4.2.10). The results for the generalized smooth scatter machine, without using the time schedule, are equal to the results for the generalized sharp scatter machine (see Theorems 4.2.6 and 4.2.8), but we are only able to not using the time schedule with the infinite and arbitrary precisions. For the fixed precision assumption we must use the time schedule to obtain information about the vertex position.

Some of the proof techniques we use are adaptations to coding advice functions in real numbers (Cantor numbers), find advice functions using Chebyshev's inequality, usual comparison between dyadic numbers or using sparse oracles (sparse sets) to code advice functions (see [6, 7]). Some most interesting techniques have to do with the physical experiment and consist on simulate a fair Boolean sequence generator with a probabilistic experiment or simulate a probabilistic answer from the oracle on a general probabilistic Turing machine (see [6, 7]), of course this last one must be restricted to dyadic probability assignments to each answer.

1.2 Mathematical preliminaries

Throughout this dissertation, without loss of generality, we use the binary alphabet $\Sigma = \{0, 1\}$. We call word over Σ (or just word) to a finite sequence of elements of Σ and we denote the set of all words over Σ by Σ^* (including the empty word represented by ε). We define the size of a word as the number of elements of Σ in the sequence and denote the size of a word w by $|w|$; we have that $|\varepsilon| = 0$. We call set to a subset of Σ^*, usually referred to as language (a set of words), and we call class to a set of subsets of Σ^* (a set of languages).

Example. *The words* ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, ... *are the first words of* Σ^* *when ordered lexicographically,* 010011000110 *is a word over* $\{0, 1\}$ *with size* 12 *and* 0^n *denotes the word of size* n *composed by* n *zeros.*

We can consider different kinds of words, as the encoding of two or more words for example. We denote the encoding of the words $w_1, w_2, \ldots, w_n$ by $\langle w_1, w_2, \cdots, w_n \rangle$ (or $w_1 \# w_2 \# \cdots \# w_n$). This encoding can be done in polynomial space on the size of the words to be encoded. We can have $|\langle w_1, w_2, \cdots, w_n \rangle| \in \mathcal{O}(s_1 + s_2 + \cdots + s_n)$, where s_i is the size of the word w_i, for any $i \in \{1, 2, \ldots, n\}$.

We can distinguish classes of languages by the difference on the resources needed to decide the classes and study the structural relations between different classes. Any class whose sets are decided by general Turing machines restricted in any suitable way on its resources would contain the language $\{\varepsilon\}$ (or any regular language) but would not contain a non-computable language as, for example, the well known halting set. Limiting the resources of Turing machines on time and space give us the most important complexity classes, we consider this restrictions on deterministic, non-deterministic and probabilistic Turing machines.

The class of sets which are decidable by a deterministic Turing machine clocked in polynomial time is called P. A Turing machine clocked on a function[1] t is restricted to perform at most $t(n)$ transitions for an input with size n. A very important class for this dissertation is the analogous to P for space restrictions, the class of sets decidable by deterministic Turing machines bounded in polynomial space, the complexity class $PSPACE$. A Turing machine bounded in space by a function[2] s is restricted to use at most $k \cdot s(n)$ cells for an input with size n, where k is the number of tapes of the Turing machine (it can use $s(n)$ cells in each tape). In order to consider probabilistic computations, we assume that, for every Turing machine clocked on a function t, for an input with size n:

- The Turing machine performs exactly $t(n)$ transitions.
- The number of possible computations[3] is exactly $2^{t(n)}$.

In the first point we can have the case that most of the $t(n)$ transitions are not computing anything and are just there so that the machine would perform the supposed amount of transitions. The second point is equivalent as stating that the computation tree of a Turing machine can be seen as a binary tree. In a forthcoming chapter we make use of computation trees for the analogue-digital machines, where ternary trees appear, the ternary nature of the trees comes from the analogue part of the machine and not from the digital part, the Turing machine. The computations of a Turing machine are either identical, as in the deterministic case, where all the $2^{t(n)}$ computations are equal, or may be different, as in the non-deterministic and probabilistic cases. Another very important class for this dissertation is

[1] The function t, so that it can restrict a Turing machine in time, must satisfy some specific conditions to be discussed on Chapter 2.

[2] Again, conditions on s to be discussed on Chapter 2.

[3] A computation is a run of the Turing machine from the initial state until some halting state.

the analogous to the well known class BPP, which is a probabilistic class, for space restrictions. BPP is the class of sets recognizable by probabilistic Turing machines clocked in polynomial time, where the Turing machines besides being probabilistic follow the bounded error probability decision criterion.

Definition 1.2.1. *A bounded error probabilistic Turing machine $\mathcal{M}$ is said to decide a set $A \subseteq \Sigma^*$ if, there is a rational number ϵ with $0 < \epsilon < 1/2$ such that, for every $w \in \Sigma^*$ (a) if $w \in A$, then $\mathcal{M}$ rejects w with probability at most ϵ and (b) if $w \notin A$, then $\mathcal{M}$ accepts w with probability at most ϵ.*

I.e. (a) if $w \in A$, then at least $(1-\epsilon) \cdot 2^{t(|w|)}$ of the computations halt in the accepting state and (b) if $w \notin A$, then at least $(1-\epsilon) \cdot 2^{t(|w|)}$ of the computations halt in the rejecting state, where $t(|w|)$ is the number of transitions of the Turing machine on input w.

The class of sets which are decidable by a bounded error probabilistic Turing machine clocked in polynomial time is called BPP. As through the whole dissertation we are only using probabilistic machines with the bounded error probability decision criterion, we are calling just probabilistic Turing machines (or probabilistic scatter machines) to bounded error probabilistic Turing machines (or bounded error probabilistic scatter machines). The name BPP comes from bounded probability polynomial time, where the beginning BP stands for bounded probability and the final P stands for polynomial time, as in the deterministic class for polynomial time restrictions which is called just P standing for polynomial time.

As we are interested in space restrictions, actually polynomial space restrictions, we need to define a complexity class with the bounded error probability decision criterion but for polynomial space restrictions. We call $BPPSPACE$ to the class of sets which are decidable by a bounded error probabilistic Turing machine bounded in polynomial space. The name, analogously to BPP, comes from bounded probability polynomial space. The beginning BP stands for bounded probability and the ending $PSPACE$ stands for polynomial space, as in the deterministic polynomial space restricted class $PSPACE$.

As we want to study an analogue-digital machine which is an oracle Turing machine we must use a different kind of complexity classes, the non-uniform complexity classes ([3, 4]). Which have a different nature from the uniform ones, since this classes are constituted by sets which are decidable by a Turing machine that consults additional information. There are no restrictions on this information, it can even be of non-computable nature, and it can be different for different input sizes. This information is provided in the form of a function.

An advice function $f : \mathbb{N} \to \Sigma^*$ is a total function which assigns a word for each (input size) $n \in \mathbb{N}$. A prefix advice function is an advice function f such that, for any $n, m \in \mathbb{N}$, if $n < m$, then $f(n)$ is a prefix of $f(m)$. We call class to a set of advice functions, as well as to a set of sets (a set of languages).

Remark 1.2.2. *For each class of advice functions there exist its subclass of prefix advice functions, consisting of the prefix functions in the class.*

Definition 1.2.3. *Let C be a class of sets and F a class of advice functions. We denote by C/F the class of sets A for which there exist a set $B \in C$ and an advice function $f \in F$ such that, for every $w \in \Sigma^*$, $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$.*

Definition 1.2.4. *Let C be a class of sets and F a class of advice functions. We denote by $C/F\star$ the class of sets A for which there exist a set $B \in C$ and a prefix advice function $f \in F$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, $w \in A$ if and only if $\langle w, f(n)\rangle \in B$.*

Each non-uniform class is based on two previously existent classes. We already introduce some classes of sets and we introduce now some classes of advice functions. The non-uniform complexity classes are classes of sets and can be used as support for another non-uniform class.

We have three basic classes of advice functions and the one that has the most important role through the dissertation is the one with polynomially sized functions. The class of advice functions f for which there exist a polynomial p such that, for every $n \in \mathbb{N}$, $|f(n)| \leq p(n)$ is called $poly$. We have also the classes of advice functions with logarithmic and exponential sized functions which are called log and exp, respectively. A short account of non-uniform classes can be found in [3].

Proposition 1.2.1. *Let C and D be classes of sets and F and G classes of advice functions. If $C \subseteq D$ and $F \subseteq G$, then $C/F \subseteq D/G$ and $C/F\star \subseteq D/G\star$.*

Proof. If $A \in C/F$ (or $A \in C/F\star$), then there exist a set $B \in C \subseteq D$ and an advice function $f \in F \subseteq G$ witnessing the fact. Consequently, $B \in D$ and $f \in G$ also witness that $A \in D/G$ (or $A \in D/G\star$). □

Proposition 1.2.2. *If C is a class of sets and F is a class of advice functions (both non-empty), then $C \subseteq C/F$ and $C \subseteq C/F\star$.*

Proof. Let A be a set of C, f any function of F and $\mathcal{M}_A$ a Turing machine which decides A in C. We consider the advice Turing machine $\mathcal{M}$ that on input $\langle w, f(|w|)\rangle$, simulates $\mathcal{M}_A$ on w and accepts its input if and only if $\mathcal{M}_A$ accepts w. Therefore, for every function $f \in F$, $\mathcal{M}$ decides the set $B = \{\langle w, f(|w|)\rangle : w \in A\}$ in C, since $\mathcal{M}$ just simulates $\mathcal{M}_A$ on a part of its input. We conclude then that $A \in C/F$ (and $A \in C/F\star$). □

Proposition 1.2.3. *Let C be a (uniform) class of sets such that $P \subseteq C$. We have that $C/poly = (C/poly)/poly$.*

Proof. $C/poly \subseteq (C/poly)/poly$ follows from Proposition 1.2.2.

If $A \in (C/poly)/poly$, then there is a set $B \in C/poly$ and an advice function $f \in poly$ such that, for every $w \in \Sigma^*$, $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$. If $B \in C/poly$, then there is a set $D \in C$ and an advice function $g \in poly$ such that, for every $y \in \Sigma^*$, $y \in B$ if and only if $\langle y, g(|y|)\rangle \in D$. Thus, for every $w \in \Sigma^*$, $w \in A$ if and only if $\langle \langle w, f(|w|)\rangle, g(|\langle w, f(|w|)\rangle|)\rangle \in D$. Let $\mathcal{M}_D$ be the Turing machine which decides D in C.

Let h be the advice function such that $h(n) = \langle f(n), g(|\langle 0^n, f(n)\rangle|)\rangle$, $h \in poly$ since $f \in poly$ and $g \in poly$ and the composition of polynomials is also a polynomial. Consider the advice Turing machine $\mathcal{M}$ such that, on input $\langle w, h(|w|)\rangle$ simulates the machine $\mathcal{M}_D$ over $\langle \langle w, f(|w|)\rangle, g(|\langle 0^{|w|}, f(|w|)\rangle|)\rangle$. Then $\mathcal{M}$ accepts $\langle w, h(|w|)\rangle$ if and only if $\mathcal{M}_D$ accepted its input, which happens if and only if $w \in A$.

To decode $\langle w, h(|w|)\rangle$ onto $\langle \langle w, f(|w|)\rangle, g(|\langle 0^{|w|}, f(|w|)\rangle|)\rangle$ polynomial time is enough since the word encoding and all the advice functions have polynomial size. So, as $P \subseteq C$ this decoding uses an amount of resources available to decide a set in C. To simulate $\mathcal{M}_D$ over its input we need at most the maximum allowed resources to decide a set in C since $\mathcal{M}_D$ decides a set in C. We can then conclude that $A \in C/poly$, since $\mathcal{M}$ runs on an amount of resources available to decide a set in C. So, $C/poly = (C/poly)/poly$. □

Remark 1.2.5. *The Proposition 1.2.3 also holds when C is a non-uniform class used as support for the non-uniform class in the statement. On the proof we would need $\mathcal{M}_D$ to be the advice Turing machine deciding sets in the support class of sets for C receiving inputs encoding the word of C with the supposed advice.*

Proposition 1.2.4. *Let C be a class of sets and F a class of advice functions. We have that $C/F\star \subseteq C/F$.*

Proof. If $A \in C/F\star$, then there exist a set $B \in C$ and a prefix advice function $f \in F$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, $w \in A$ if and only if $\langle w, f(n)\rangle \in B$. Thus, the same set B and advice function f witness that $A \in C/F$ since the condition holds for $|w| = n$. □

The next proposition has an important role throughout the dissertation and its proof makes use of the following prefix advice function (f').

Let $g : \Sigma^* \to \Sigma^*$ be the function such that $g(w)$ is obtained from w by replacing every 0 by 00 and every 1 by 11. Then, given $f \in poly$, we can define the prefix advice function f' such that $f'(n) = g(f(0))01g(f(1))01 \cdots g(f(n))01$. $f' \in poly$ since as $f \in poly$, there is a polynomial p such that, for every $n \in \mathbb{N}$, $|f(n)| < p(n)$, we can conclude that $|f'(n)| < (2p(n) + 2)n$.

Remark 1.2.6. *The prefix advice function f' can be interpreted as $f'(n) = f(0)\#f(1)\#\cdots\#f(n)\#$. We use prefix advice functions like this in Chapter 3, but without the last separation mark (#).*

Proposition 1.2.5. *Let C be a class of sets with $P \subseteq C$. We have that $C/poly = C/poly\star$.*

Proof. It follows from Proposition 1.2.4 that $C/poly\star \subseteq C/poly$.

If $A \in C/poly$, then there exist a set $B \in C$ and an advice function $f \in poly$ such that $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$. Let $\mathcal{M}_B$ be the Turing machine which decides B in C and $f' \in poly$ be the prefix advice function described above for this f.

Consider the advice Turing machine $\mathcal{M}$ such that on input $\langle w, f'(n)\rangle$, for some $n \geq |w| = r$, reads $f'(n)$ in pairs of digits until the r^{th} occurrence of 01 to find the word $y = g^{-1}(z) \in \Sigma^*$, where $z \in \Sigma^*$ is the word confined between the r^{th} and the $r^{th} + 1$ occurrences of 01 in $f'(n)$. Afterwards $\mathcal{M}$ simulates $\mathcal{M}_B$ on $\langle w, y\rangle$ and accepts if and only if $\langle w, y\rangle \in B$. Thus, $\mathcal{M}$ accepts $\langle w, f'(n)\rangle$ if and only if $w \in A$.

Since $\mathcal{M}_B$ decides a set in C and retrieving y from $f'(n)$ requires a polynomial amount of time, we conclude that $\mathcal{M}$ decides a set also in C (we assumed $P \subseteq C$). Therefore, $C/poly \subseteq C/poly\star$ and the equality follows. □

Remark 1.2.7. *From Proposition 1.2.5 we have $P/poly = P/poly\star$ for $C = P$ and $PSPACE/poly = PSPACE/poly\star$ for $C = PSPACE \supseteq P$.*

The class of sets BPP is a common support for non-uniform classes, as in [1], and we also intend to use $BPPSPACE$ as support for non-uniform classes. However, simply stating that $A \in BPP/F$ (or $A \in BPPSPACE/F$) we are stating that, there must be a set $B \in BPP$ (or $B \in BPPSPACE$) and an advice function $f \in F$ such that, $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$. This means that the advice function is to be chosen after we fix the probabilistic advice Turing machine which witnesses that $B \in BPP$ (or $B \in BPPSPACE$) and so, we fix the bounded error together with the Turing machine and only then the advice function. Inhere, we provide a different definition:

Definition 1.2.8. *Let F be a class of advice functions, we denote by $BPP//F$ the class of sets A for which there exist a probabilistic advice Turing machine $\mathcal{M}$ clocked in polynomial time, a constant ϵ with $0 < \epsilon < 1/2$ and an advice function $f \in F$ such that, for every $w \in \Sigma^*$, (a) if $w \in A$, then $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most ϵ and (b) if $w \notin A$, then $\mathcal{M}$ accepts $\langle w, f(|w|)\rangle$ with probability at most ϵ.*

Definition 1.2.9. *Let F be a class of advice functions, we denote by $BPP//F\star$ the class of sets A for which there exist a probabilistic advice Turing machine $\mathcal{M}$ clocked in polynomial time, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in F$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, (a) if $w \in A$, then $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most ϵ and (b) if $w \notin A$, then $\mathcal{M}$ accepts $\langle w, f(n)\rangle$ with probability at most ϵ.*

Definition 1.2.10. *Let F be a class of advice functions, we denote by $BPPSPACE//F$ the class of sets A for which there exist a probabilistic advice Turing machine $\mathcal{M}$ bounded in polynomial space, a constant ϵ with $0 < \epsilon < 1/2$ and an advice function $f \in F$ such that, for every $w \in \Sigma^*$, (a) if $w \in A$, then $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most ϵ and (b) if $w \notin A$, then $\mathcal{M}$ accepts $\langle w, f(|w|)\rangle$ with probability at most ϵ.*

Definition 1.2.11. *Let F be a class of advice functions, we denote by $BPPSPACE//F\star$ the class of sets A for which there exist a probabilistic advice Turing machine $\mathcal{M}$ bounded in polynomial space, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in F$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, (a) if $w \in A$, then $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most ϵ and (b) if $w \notin A$, then $\mathcal{M}$ accepts $\langle w, f(n)\rangle$ with probability at most ϵ.*

Proposition 1.2.6. *Let F be a class of advice functions. We have that $BPP/F \subseteq BPP//F$, $BPP/F\star \subseteq BPP//F\star$, $BPPSPACE/F \subseteq BPPSPACE//F$ and $BPPSPACE/F\star \subseteq BPPSPACE//F\star$.*

Proof. If $A \in BPP/F$ (or $A \in BPP/F\star$), we have the set $B \in BPP$ and advice function $f \in F$ which witness the fact. The probabilistic advice Turing machine that witnesses $B \in BPP$ has a specified bounded error probability ϵ, if we choose that Turing machine, the same ϵ and the same advice function f, we are deciding the same set A for the definition of $BPP//F$ (or $BPP//F\star$). For the $BPPSPACE$ case is exactly the same where $B \in BPPSPACE$. □

It is known that $BPP/poly = BPP//poly$ but not whether $BPP/log\star = BPP//log\star$ or not.

Proposition 1.2.7. $BPP/poly = BPP//poly$.

Proof. It follows from Proposition 1.2.6 that $BPP/poly \subseteq BPP//poly$.

If $A \in BPP//poly$, there exist a probabilistic advice Turing machine $\mathcal{M}$ clocked in polynomial time, a constant ϵ with $0 < \epsilon < 1/2$ and an advice function $f \in poly$ such that, for every $w \in \Sigma^*$, if $w \in A$, $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most ϵ and if $w \notin A$, $\mathcal{M}$ accepts $\langle w, f(|w|)\rangle$ with probability at most ϵ.

For the advice function $f \in poly$, consider the set $B = \{\langle w, f(|w|)\rangle \in \Sigma^* : w \in A\}$, which is witnessed to be in BPP with bounded error ϵ by $\mathcal{M}$, since $\mathcal{M}$ is clocked in polynomial time. We conclude then that $A \in BPP/poly$ since by definition we have that $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$. Thus, we have the equality $BPP/poly = BPP//poly$. □

Since for any class of advice functions F we have $BPP \subseteq BPP/F \subseteq BPP//F$ follows that $BPP \subseteq BPP//F$, analogously we have $BPP \subseteq BPP//F\star$, $BPPSPACE \subseteq BPPSPACE//F$ and $BPPSPACE \subseteq BPPSPACE//F\star$.

Proposition 1.2.8. *Let F and G be classes of advice functions. If $F \subseteq G$, then $BPP//F \subseteq BPP//G$, $BPP//F\star \subseteq BPP//G\star$, $BPPSPACE//F \subseteq BPPSPACE//G$ and $BPPSPACE//F\star \subseteq BPPSPACE//G\star$.*

Proof. If $A \in BPP//F$ ($A \in BPP//F\star$, $A \in BPPSPACE//F$ or $A \in BPPSPACE//F\star$), then the advice function $f \in F$ which witnesses the fact also belongs to G, since $F \subseteq G$. Thus, for the same advice Turing machine $\mathcal{M}$ and bounded error ϵ which also witness that $A \in BPP//F$, we also have that $A \in BPP//G$ ($A \in BPP//G\star$, $A \in BPPSPACE//G$ or $A \in BPPSPACE//G\star$). □

Proposition 1.2.9. *Let F be a class of advice functions. We have that $BPP//F\star \subseteq BPP//F$ and $BPPSPACE//F\star \subseteq BPPSPACE//F$.*

Proof. If $A \in BPP/F\star$ (or $A \in BPPSPACE/F\star$), then there exist a probabilistic advice Turing machine $\mathcal{M}$, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in F$ which witness the fact. Thus, the same Turing machine $\mathcal{M}$, constant ϵ and advice function f witness that $A \in BPP/F$ (or $A \in BPPSPACE/F$) since the condition holds for $|w| = n$. □

Proposition 1.2.10. $BPPSPACE//poly = BPPSPACE//poly\star$.

Proof. It follows from Proposition 1.2.9 that $BPPSPACE//poly\star \subseteq BPPSPACE//poly$.

If $A \in BPPSPACE//poly$, then there exist a probabilistic advice Turing machine $\mathcal{M}$, a constant ϵ and an advice function $f \in poly$ such that, for every $w \in \Sigma^*$, if $w \in A$, then $\mathcal{M}$ rejects $\langle w, f(|w|)\rangle$ with probability at most ϵ and if $w \notin A$, then $\mathcal{M}$ accepts $\langle w, f(|w|)\rangle$ with probability at most ϵ. Let $f' \in poly$ be the prefix advice function described before Proposition 1.2.5 for this $f \in poly$.

Consider the advice Turing machine $\mathcal{M}'$ such that on input $\langle w, f'(n)\rangle$, for some $n \geq |w| = r$, reads $f'(n)$ in pairs of digits until the r^{th} occurrence of 01 to find the word $y = g^{-1}(z) \in \Sigma^*$, where $z \in \Sigma^*$ is the word confined between the r^{th} and the $r^{th}+1$ occurrences of 01 in $f'(n)$. Afterwards $\mathcal{M}'$ simulates $\mathcal{M}$ on $\langle w, y\rangle$ and accepts $\langle w, f'(n)\rangle$ if and only if $\mathcal{M}$ accepts $\langle w, y\rangle$. Thus, $\mathcal{M}'$ decides A for the same bounded error probability of $\mathcal{M}$ since it rejects $\langle w, f'(n)\rangle$ with probability at most ϵ if $w \in A$ and accepts $\langle w, f'(n)\rangle$ with probability at most ϵ if $w \notin A$.

As retrieving y from $f'(n)$ requires a polynomial amount of time, and so, at most a polynomial amount of space, and simulate $\mathcal{M}$ on $\langle w, y\rangle$ requires also a polynomial amount of space, since $\mathcal{M}$ is bounded in polynomial space and $\langle w, y\rangle$ has polynomial size on $\langle w, f'(n)\rangle$ ($|\langle w, y\rangle| \leq |\langle w, f'(n)\rangle|$). We conclude that $A \in BPPSPACE//poly\star$ and then that $BPPSPACE//poly = BPPSPACE//poly\star$. □

The computation model we intend to study is characterized by non-uniform complexity classes. Sets in non-uniform classes are decided with potential access to non-computable information. The class P/exp, for example, coincides with the class of all languages (the parts of Σ^*, which we denote by 2^{Σ^*}).

The proof technique for the next proposition has an important role throughout the dissertation and it makes use of the following prefix advice function $f \in exp$. Consider all the words in Σ^* ordered lexicographically and note that there are 2^n words in Σ^* with size n. Given a set $A \subseteq \Sigma^*$, consider the advice function f such that $f(n) = f_1^n f_2^n \cdots f_{2^n}^n$, where f_i^n is either 0 or 1 depending on if the i^{th} word with size n of Σ^* ordered lexicographically belongs or not to the set A, respectively. f is obviously in exp since $|f(n)| = 2^n$ for any $n \in \mathbb{N}$.

Proposition 1.2.11. *Any set $A \subseteq \Sigma^*$ belongs to P/exp.*

Proof. Given a set $A \subseteq \Sigma^*$ we can consider the advice function f as described above and the advice Turing machine $\mathcal{M}$ such that on the input $\langle w, f(|w|)\rangle$, computes the position of w among all the $|w|$ sized words in Σ^* ordered lexicographically, afterwards $\mathcal{M}$ checks in $f(|w|)$ if the word w belongs or not to A. Then, $\mathcal{M}$ accepts $\langle w, f(|w|)\rangle$ if and only if $w \in A$.

To compute the position of w a linear time on $|\langle w, f(|w|)\rangle|$ is enough, since a linear time on $|f(|w|)| = 2^{|w|}$ is enough. We conclude then that $A \in P/exp$. □

Remark 1.2.12. *To compute the position of a word w it is not enough a polynomial time on $|w|$. We need indeed the polynomial time over $2^{|w|}$, or exponential time on $|w|$. The non-uniform complexity classes have this property, which goes without seeing through the dissertation, that the restriction on its computational resources is made over the size of the word in the class plus the respective advice.*

Based on this result it would be wise to understand if any non-uniform class equals the whole space of languages or if there is a counter example. There exist indeed a counter example, $P/poly$ do not contain some specific computable sets.

There is a well known characterization for $P/poly$ which states that the languages in $P/poly$ are the languages which have polynomial size Boolean circuits (see [3]). A Boolean circuit is constructed for a specific input size. Stating that a language is decidable by Boolean circuits means that there is a Boolean circuit for each input size deciding the correspondent language when restricted to the words with that size.

It is also known that for a sufficiently large $n \in \mathbb{N}$ exist a n variables Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that no Boolean circuit which computes f has polynomial size (has a polynomial amount of logic gates).

Proposition 1.2.12. *There exist sets in $EXPSPACE$ that are not in $P/poly$ ($EXPSPACE \nsubseteq P/poly$).*

Proof. For any $n \in \mathbb{N}^+$, consider the 2^{2^n} n variables Boolean functions ordered lexicographically by their truth tables. For any n sufficiently large to have a function $f : \{0,1\}^n \to \{0,1\}$ which cannot be computed by polynomially sized Boolean circuits, denote by f_n the first function which is not computed by any polynomially sized Boolean circuit. For all the other n denote by f_n the first n variables Boolean function in the considered order.

Let A be the set $A = \{w \in \Sigma^* : f_{|w|}(w) = 1\}$. The set A is clearly not in $P/poly$ by the characterization described above.

We can decide A in $EXPSPACE$ since enumerating all n variables Boolean functions, computing the circuit complexity (number of logic gates) of each one and find the first which is not of polynomial size can all be done in exponential space. □

It is a known fact that we can consider an enumeration for all the Turing machines, we can even consider an infinite amount of different enumerations. For the next proposition fix an enumeration, say M_1, M_2, M_3, ...

The class $P/poly$, besides not containing every computable sets, contains some non-computable sets, as an incomplete version of the halting set, for example. The set $\{0^n : M_n(n) \text{ halts}\}$, where $M_n(n)$ stands for the Turing machine M_n over the input n (the input can be given in its binary form).

Proposition 1.2.13. *The set $\{0^n : M_n(n)$ halts$\}$, a version of the halting set, is in $P/poly$.*

Proof. Let f be the advice function such that $f(n)$ is either 1 or 0 depending on if the Turing machine M_n halts on input n or not, respectively. f is obviously in $poly$ since $|f(n)| = 1$ for any $n \in \mathbb{N}$.

Consider the advice Turing machine $\mathcal{M}$ which for an input $w = \langle 0^n, f(n) \rangle$ ($|0^n| = n$) consults the oracle and accepts if and only if the advice is 1 (for any other input the Turing machine rejects w). The set $\{0^n : M_n(n) \text{ halts}\}$ is clearly decided by $\mathcal{M}$ in polynomial time, since the decision is directly given by the input. We conclude then that $\{0^n : M_n(n) \text{ halts}\} \in P/poly$. □

Chapter 2

Clocks and physical oracle

2.1 Clocks

With this dissertation we intend to study the computational power of the scatter machine bounded in polynomial space. To that purpose we need to introduce a specific kind of Turing machines – the clocks (see [16]), since we need to use clocks as an integrand digital component of the scatter machines. Through this Section we explicitly specify two clocks (Figures 2.1 and 2.2), we should then explain that we consider the infinite tapes of a Turing machine with a first cell. We assume the first cell to be on the left hand side and the tapes to be infinite to the right hand side.

To introduce the Turing machine as a clock, we need to define a particular set of functions, the time constructible functions.

Definition 2.1.1. *We say that a total function $f : \mathbb{N} \to \mathbb{N}$ is time constructible if there is a deterministic Turing machine $\mathcal{M}$ and a number $n \in \mathbb{N}$ such that, for any input w with $|w| \geq n$, $\mathcal{M}$ halts after performing exactly $f(|w|)$ transitions.*

The Turing machine in the previous definition is called a clock. A clock is a deterministic Turing machine that, for sufficiently large inputs, halts after a specific number of transitions depending on the input size. Given a time constructible function $f : \mathbb{N} \to \mathbb{N}$, a clock for f is a deterministic Turing machine that halts after exactly $f(n)$ transitions, for inputs with sufficiently large size n.

Remark 2.1.2. *A clock for $f(n) = n + 1$ is a deterministic Turing machine that reads the whole input and halts after detecting the end of the input. Any non-constant time constructible function $g : \mathbb{N} \to \mathbb{N}$ is such that $g(n) \geq n + 1$ (except for a finite amount of inputs), since a Turing machine as clock for g must read the input and detect its end in order to halt depending on the input size. There exists a clock for any polynomial p satisfying $p(n) \geq n + 1$ and any polynomial clock operates in polynomial space (see [3], §2.4).*

We have an analogous concept for space, where we want to bound the number of cells used by the Turing machine. We only consider, in the following definition, the cells on the work tapes and not the cells on the input tape.

Definition 2.1.3. *We say that a total function $f : \mathbb{N} \to \mathbb{N}$ is space constructible if there is a deterministic Turing machine $\mathcal{M}$ and a number $n \in \mathbb{N}$ such that, for any input w with $|w| \geq n$, $\mathcal{M}$ halts having exactly $f(|w|)$ non-blank cells (on its work tapes) and no current blank cell was read through the whole computation.*

We can have non-constant space constructible functions $f : \mathbb{N} \to \mathbb{N}$ with $f(n) < n + 1$, for example the total function[1] $f(n) = \lfloor \log n \rfloor + 1$ is indeed space constructible. Actually, $\lfloor \log n \rfloor + 1$ is the number of digits needed to represent n in the usual binary representation of integers and we need to mark that space (amount of cells) in a tape of a clock (see Figure 2.2).

Remark 2.1.4. *If we have a time constructible function f, then f is space constructible. The reciprocal is not true.*

We intend to study an analogue-digital machine consisting of a deterministic Turing machine coupled with a physical experiment. It may be the case that the physical experiment takes an intrinsic time to get its conclusions and we may need to be careful so that the experiment would not take an infinite time. We can then use a clock, embedded in the digital part, to bound the time that the experiment can wait. When we have a clock incorporated in an analogue-digital machine, aiming to bound the time for the experiments, that clock is called time schedule. Those clocks are defined to take the query as its input and so, we have the time for the experiment bounded on the query size.

This particular Turing machines we call clocks are supposed to halt and turn off on some halting state after performing a specific amount of transitions depending on the input size. So, they behave as an alarm clock more than as a clock. This Turing machines can be aware of when they have already had time to accomplish a specific amount of transitions. They are not supposed to run, aware of the amount of transitions they have already performed, for an arbitrary large amount of transitions. We say that a clock ticks a certain amount of transitions when it turns off in some halting state after having performed that amount of transitions, aware that it has already performed them.

In this section we present two exponential clocks which operate in polynomial space. Usually, this is not a common attention while constructing clocks but we need such a device for the purpose of this dissertation. We intend to use analogue-digital machines bounded in polynomial space that may need to have a time schedule to bound the time for the experiments. It is then helpful to know the limitations we have for constructing the clocks (see Proposition 2.1.2) and if we can indeed construct a clock for the maximum growing rate possible (see Proposition 2.1.1). We present in Figure 2.1 the clock used for the proof of Proposition 2.1.1.

Proposition 2.1.1. *There exists a clock bounded in polynomial space such that, for an input of size n, ticks an exponential amount of transitions on n.*

Proof. We depict in Figure 2.1 a clock that halts after $2n \cdot 2^n + 1$ transitions for an input with size n. The clock works in linear space, it uses at most $n + 1$ cells in each tape, so at most $2n + 2$ cells, since it has two tapes. The clock is specified with two tapes where the first tape is the input tape which we use as a general work tape since we only need the size of the input word. But it could be specified with the input tape plus two work tapes, where the work tapes would be used as the two existent tapes on the specified clock, and the input tape would be used at the beginning to detect the end of the input.

We can also compose this clock with a polynomial clock between the first and third states (q_1 and q_3) using some additional tapes, as a third tape for the input and, besides those already considered, other needed tapes for the polynomial clock. Note that considering only the first three states, q_1, q_2 and q_3 with the state q_3 as an halting state, and the transitions between them excluding the loop in q_3 we get a clock for $n + 1$. The composition must start with the polynomial clock, where for any but the last of its transitions must move to the right on the first two tapes writing 0 on the first one (it would write 1 at the first cell of the second tape). Then, instead of having the polynomial clock transiting to its halting state, it must transit to q_3 and follow the instructions in Figure 2.1 ignoring all the tapes but the first two (in the transition to q_3 the clock must move to the left on the first two tapes). We then have, for any polynomial

[1] Throughout the dissertation we always consider the function $\log$ with base 2.

p with $p(n) \geq n + 1$, a clock halting after $2(p(n) - 1) \cdot 2^{p(n)-1} + 1$ transitions for an input of size n using at most a polynomial amount of cells in each tape. We may get, from the composition, a clock which do not works for a greater amount of inputs than the specified one, but it always works for every but a finite amount of inputs. □

We use a general form of depiction for Turing machines (which can be seen as a graph). The circles (vertices of the graph) are the states of the Turing machine where the initial state has a little incoming arrow and the halting state has a double line circumference around it, respectively q_1 and q_7 in Figure 2.1. The arrows between states (edges of the graph) are the transitions of the Turing machine and have a label each. The labels are divided in two parts, by an arrow "$\rightarrow$", that specify before the arrow what each head should read and after the arrow what each head should write and where to move. The specifications for different tapes are divided by semicolons and the commas separate the information of what to write from where to move.

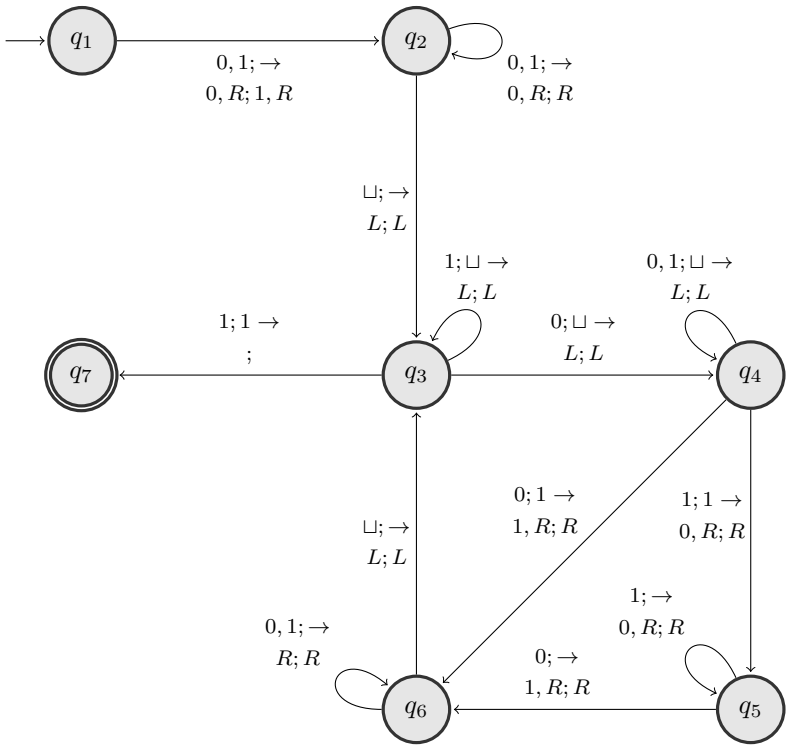

Figure 2.1: Clock for $f(n) = 2n \cdot 2^n + 1$

We divided the labels for the transitions in two lines so that they would not be too much long. On the first line we have what each head should read and on the second one what each head should write and where to move. We constructed the clock with two tapes where the second one is just a mark to know which cell is the first cell, the first tape is the input tape but it is being used as a general work tape. If we would consider an extensive alphabet, we could use other symbols on the first cell and we would not need to have two tapes. For example, we could use the alphabet $\{0,\ 1,\ 0',\ 1'\}$ so we would use the symbols $0'$ and $1'$ instead of 0 and 1 on the first cell, respectively.

We specified this simplified version of the clock which only works for inputs of size greater or equal to 2, but it would be easy to modify it in order to work for all inputs. We also allowed the "$+1$" constant that is easy to avoid.

A transition of a Turing machine is the step of an edge from the graph which depicts the Turing machine. A configuration of a Turing machine is an hypothesis of: state, position of the heads and text written in all the tapes. A transition dictates then a change of configuration, if a transition occurs that do not change the configuration, then either the machine is non-deterministic or it is stuck in an infinite cycle. We must clarify that these clocks do not measure physical time, they measure logic time, time induced by a digital process. We consider that a transition of a Turing machine, that we assume it has constant physical time, is one unit of time (logic time).

The concept of transition and configuration is then deeply related, and we use a common proof technique based on counting the number of configurations to prove the following Proposition.

Proposition 2.1.2. *In polynomial space, any clock ticks at most an exponential amount of transitions.*

Proof. Consider a clock $\mathcal{M}$ bounded in polynomial space p, and lets assume $\mathcal{M}$ has k tapes, r states and uses an alphabet of a symbols. For an input of size n, $\mathcal{M}$ can use at most $k \cdot p(n)$ cells ($p(n)$ cells per tape).

The number of heads coincides with the number of tapes, k, and each head can be positioned at one of the $p(n)$ cells of its tape. There are then $p(n)^k$ possible scenarios for the positions of the heads. There are also $(a+1)^{k \cdot p(n)}$ possible scenarios for what is written in the tapes (considering blank cells), which gives us $p(n)^k \cdot (a+1)^{k \cdot p(n)}$ possible tape configurations. On the other hand, the finite control can happen to be at any of the r states of $\mathcal{M}$ for each different tape configuration, which gives us a total of $r \cdot p(n)^k \cdot (a+1)^{k \cdot p(n)}$ possible configurations for $\mathcal{M}$.

Since a clock is a deterministic Turing Machine, once it assumes a particular configuration (same state, same position of heads and same text written on all the tapes) it follows a deterministic behaviour. Therefore, if $\mathcal{M}$ assumes a configuration that it had already assumed before in the same computation, it must proceed exactly as it proceeded the last time it reached that configuration, and it will again be lead to the same configuration. The reasoning can be applied every time $\mathcal{M}$ is in that same configuration and we conclude then that $\mathcal{M}$ will assume that configuration an infinite amount of times. There is then no possible way for $\mathcal{M}$ to halt: it is stuck in an infinite cycle.

Since a clock must halt, it must do so before repeating a configuration, and $\mathcal{M}$ can tick at most $r \cdot p(n)^k \cdot (a+1)^{k \cdot p(n)}$ transitions without repeating a configuration. Therefore, we conclude that a clock bounded in polynomial space with the considered constants can only be a clock for time constructible functions f, where $f(n) < r \cdot p(n)^k \cdot (a+1)^{k \cdot p(n)}$ and so, it is at most an exponential clock. □

We specified in Figure 2.1 an exponential clock which runs in polynomial space. That clock is for the function $2n \cdot 2^n + 1$ that is not close to the simple time constructible function 2^n, which is usually constructed in time with clocks running in exponential space. We were not able to obtain a precise clock for 2^n in polynomial space, but we managed to have a clock for $f(n) = 2n \cdot 2^{n-(\lfloor \log n \rfloor + 1)} + 1$ in polynomial space, actually in linear space. This time constructible function satisfy $2^n \leq f(n) - 1 < 2^{n+1}$ since $f(n) - 1 = 2^{\log n + 1} \cdot 2^{n - \lfloor \log n \rfloor - 1} = 2^{n + \log n - \lfloor \log n \rfloor}$ and so, it is much closer to 2^n than $2n \cdot 2^n + 1$ which is strictly bigger than 2^{n+1} since $2n \cdot 2^n = 2^{n + \log n + 1}$.

We have the clock for the function $f(n) = 2n \cdot 2^{n-(\lfloor \log n \rfloor + 1)} + 1$ in polynomial space depicted in Figure 2.2. We divided as before the labels in two lines, but this time we omitted the arrow "$\rightarrow$" so that the labels would not be even more inconveniently long. We have three tapes where the first tape is the input tape used as a general work tape, and again, we did not avoid the "$+1$" constant or made the clock working for all inputs, this one only works for inputs greater or equal to 5. On this clock we used the alphabet $\{0, 1, 0', 1'\}$, so that we would not need an extra tape marking the first cell. The symbols $0'$ and $1'$ should be read as 0 and 1, respectively, and only occur at the first cell of the first tape.

An analogous composition as the one suggested in proof of Proposition 2.1.1 (composition with a polynomial clock) can also be used in this clock. In this case the polynomial clock must also be po-

sitioned between the first and third states (q_1 and q_3), but we must prepare also the second and third tapes besides the first one during its transitions. We must separate the input tape from the first tape, using a fourth tape as input tape plus other needed tapes for the polynomial clock. We then get, for any polynomial p with $p(n) \geq n+1$, a clock for $2(p(n)-1) \cdot 2^{p(n)-1-(\lfloor \log(p(n)-1) \rfloor + 1)} + 1$ using at most a polynomial amount of cells in each tape.

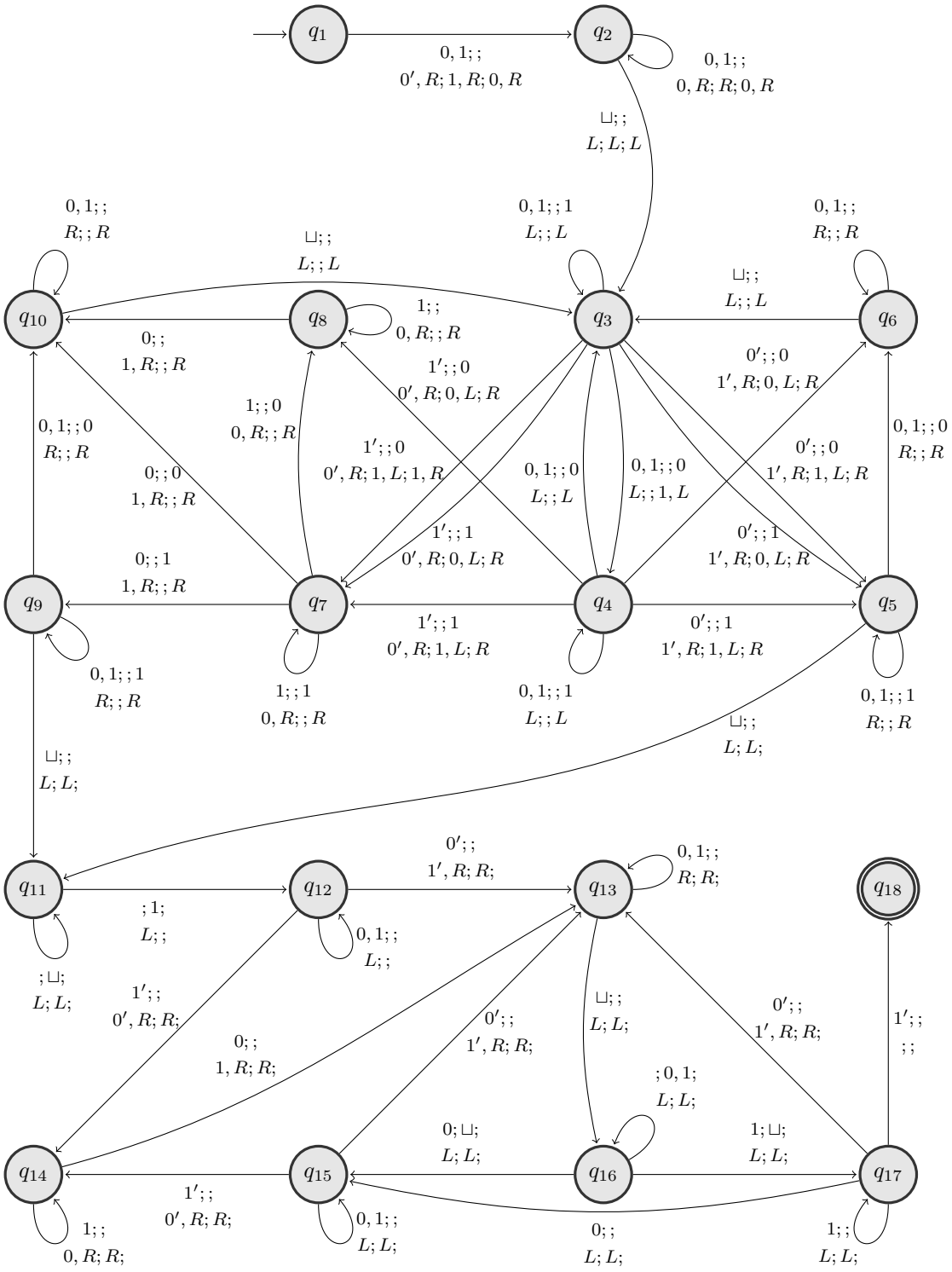

Figure 2.2: Clock for $f(n) = 2n \cdot 2^{n-(\lfloor \log n \rfloor + 1)} + 1$

2.2 Scatter experiment

Coupling oracles to Turing machines is a way to boost the computational power of Turing machines, or to speed up or reduce the space needed for the computations. As already explained, an oracle is a set for which the Turing machine has three distinguished states and a distinguished tape to communicate with: the 'query', 'yes' and 'no' states and the query tape. We call oracle Turing machine to a Turing machine coupled with an usual oracle.

We intend to use a particular non-standard type of oracle which consists of a physical experiment. In fact we are interested in a specific experiment, the scatter experiment. This experiment is supposed to measure the position of a vertex of a wedge by shooting particles from a cannon and checking if they hit the wedge to the right or to the left of the vertex. To this purpose there are two collecting boxes, one at each side of the wedge, which detect if a particle has entered it. We consider two different versions of this experiment, the sharp and the smooth scatter experiments, which consist of an experiment with a sharp or a smooth wedge, respectively.

The physical experiment is described as part of a physical system, which is governed by some physical laws. This subject is already studied in [14, 12], but basically, we assume that all the components from the experiment behave suitably. The particles obey Newton's laws of motion, the wedges are rigid and we have perfectly elastic collisions as well as preservation of kinetic energy and we have well behaved cannons, detectors and clocks in terms of direction and velocity of shots, detection precision and consistency in time measure, respectively.

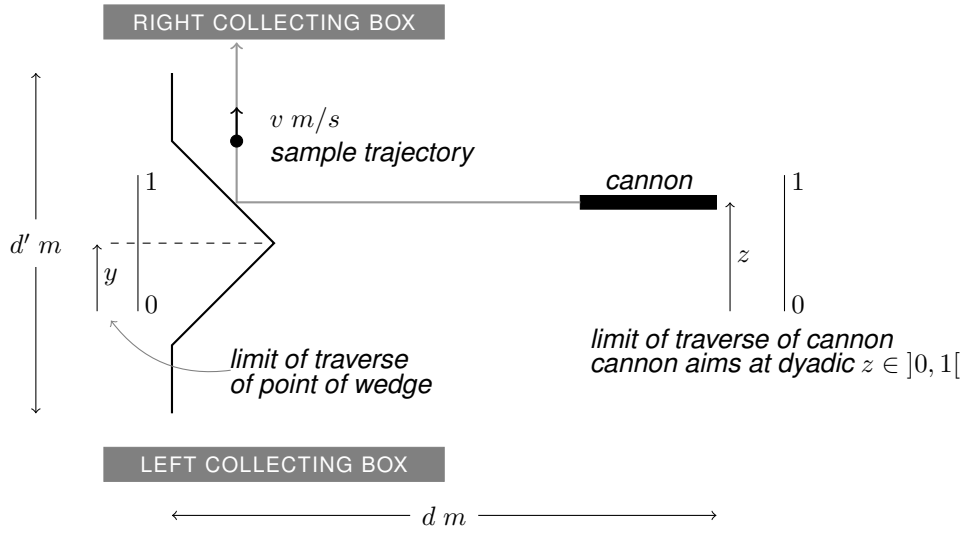

Figure 2.3: Sharp scatter experiment

In the sharp version of the experiment depicted in Figure 2.3 (introduced in [14]) the wedge is sharp and has two linear sides, which make a right angle between them in the vertex position and a $45°$ angle with the direction defined by the cannon shot. Let y be the position of the vertex and say that the particle is fired with a fixed velocity v from the position of the cannon z. We assume, without loss of generality, that the positions are described by real numbers in the interval $]0,1[$. Approximations to the vertex position are to be obtained by a Turing machine that can only specify dyadic numbers in the query tape.

When there is a shot, the particle hits the wedge and follows the behaviour:

- If $z > y$, the particle enters the right collecting box;

- If $z < y$, the particle enters the left collecting box;
- If $z = y$, we make no assumptions on the behaviour of the particle.

Remark 2.2.1. *So that the third case ($z = y$) would happen, we would need a dyadic number y, since z is always dyadic as it is given by a Turing machine. But even then, choosing random positions for the cannon, z, the probability of having $z = y$ is zero, since there are an infinite amount of dyadic numbers within $]0, 1[$ (or any of its subintervals).*

We know that unless $z = y$ the particle is detected in a collecting box within a constant time, determined by the velocity of the particle v and the distances d and d' from the cannon to the wedge and between the two collecting boxes, respectively. The sides of the wedge make $45°$ angles with the line of the cannon and then the particle are reflected in a right angle towards a collecting box. Thus, we can either have the Turing machine waiting the proper amount of time for the result of the experiment, depending on v, d and d', or set v, d and d' in such a way that the particle is detected by a collecting box in the amount of time a Turing machine needs to perform a transition, since that time does not depend on the distance between z and y.

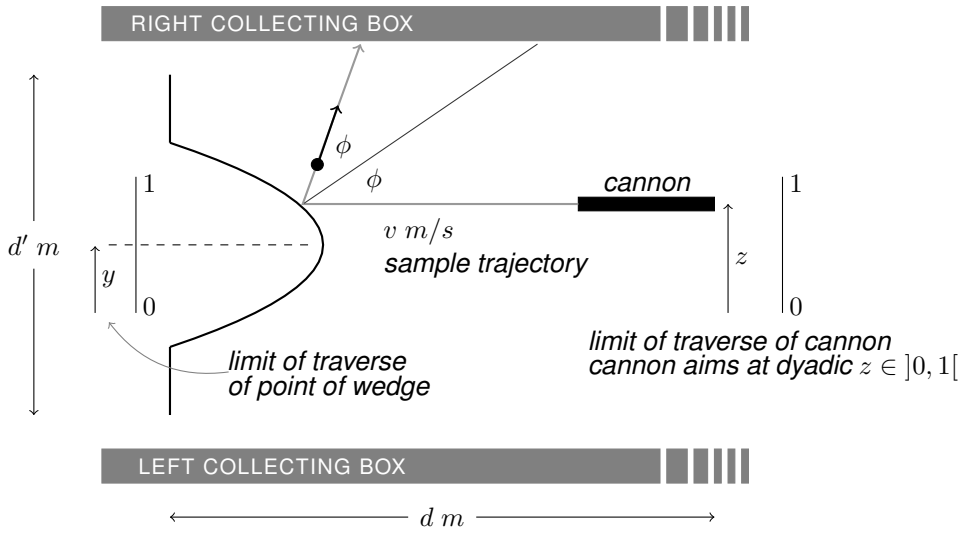

Figure 2.4: Smooth scatter experiment

In the smooth version of the experiment (introduced in [12]), as depicted in Figure 2.4, we have a smooth wedge. In this case, we do not have a constant time bound as before for the time of the experiment, the time the experiment takes to run depends on the difference between y and z. When the particle hits the wedge is reflected as if it hits the tangent line to the wedge at the point of impact, the angle the particle against the wedge makes with the normal to the tangent is the same as the particle against the collecting box makes. So, as close to the vertex the particle hits the wedge is as much time it takes to enter some collecting box. This time is called physical time. In this case, we assume that if $z = y$, the particle is reflected backwards and it is not detected in any of the collecting boxes. Note that, here, the collecting boxes are necessarily infinite.

As studied in [12] we know that the physical time taken by the smooth scatter experiment is exponential on $|y-z|^{-1}$. Lets consider the function $g :]0, 1[\to \mathbb{R}$ such that $g(x)$ describes the shape of the wedge of a smooth scatter experiment (considering the x axis from left to right). If $g(x)$ is n times continuously differentiable near the vertex position y, with non-zero n^{th} derivative and all the other derivatives until

the $(n-1)^{th}$ vanishing for $x = y$. Then, with a shot performed at the position z, the time the experiment takes is $t(z)$, where

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

for some real numbers $A,\ B > 0$, when $|y - z|$ is sufficiently small. (see [12] for details)

We can then conclude that the physical time of the smooth scatter experiment goes to infinity when z gets closer to y, being infinite if $z = y$. Without loss of generality, we assume, for computation purposes, that the smooth scatter experiments we use have a shape, $g(x)$, satisfying the conditions above for $n = 2$ and that the physical time of the experiment is indeed $t(z) = |y - z|^{-1}$. We are just setting the constants to 1.

2.3 Protocol

We intend to use these experiments as oracles to Turing machines. So, we have to explain how is the information processed between a Turing machine and an experiment — the communication protocol.

We have three different assumptions on the precision an experiment can set the parameters specified by a Turing machine (see [19]).

- The infinite precision, which is when the experiment can set the parameters exactly as the Turing machine specifies.
- The arbitrary precision, which is when the experiment can commit errors setting the parameters, and the errors can be arbitrarily small and have their limits specified by the Turing machine.
- And the fixed precision, which is when the experiment can commit errors setting the parameters, and the errors have fixed limits, intrinsic to the experiment, that cannot be reduced.

They could also be thought as three different precision assumptions an human technician could set the parameters for an experiment. We define a different protocol for each different precision assumption. The Turing machine, which aims to use the experiment as an oracle, is modelling a technician, having to set the parameters for each run of the experiment. Since considering a fixed experiment, which has a specific vertex position, the only parameter left to set is the cannon position. We must describe how the Turing machine specifies the cannon position.

To interact with the experiment the Turing machine have a distinguished tape, as well as to interact with an usual oracle, which is the query tape, and three (or four) additional states, the ‘query’ state (as for an usual oracle) and the ‘right’ and the ‘left’ states, which can be thought as corresponding to the ‘yes’ and ‘no’ states. In some cases of the experiment, mainly in the smooth ones, we may need another state, called ‘timeout’ state, with the purpose of being another possible answer of the experiment as the ‘right’ or ‘left’ states, aimed for when the experiment takes too much time and we want to stop it, so we can keep going with the digital computation on the Turing machine. We may also have to use it to specify that the particle has hit the vertex in the sharp case of the experiment.

To perform a call to the oracle, when the Turing machine is on its digital computations, writes a word in the query tape and if it transits to the ‘query’ state, the word in the query tape is used, according to the protocol (to be defined below), to set the cannon position of the experiment. Afterwards, in the analogue part (the scatter experiment) a shot is performed and an answer is possibly given by the experiment detecting the particle in one of its collecting boxes. If the particle is detected in the right collecting box, in the digital part there is a transition of the Turing machine to the ‘right’ state, and if the particle is detected

in the left collecting box, there is a transition to the 'left' state. We sometimes couple the analogue-digital machine with the time schedule (mostly in the smooth case), in its digital part, and then if the time schedule halts before the particle being detected in a collecting box, a transition to the 'timeout' state occurs. We may need to use a time schedule in the sharp case to decide if the particle has hit the vertex.

As noticed before, we have to specify how the cannon position is specified by the word in the query tape. The cannon position must be a dyadic number in $]0,1[$ since it is given in a finite word to be set in a finite amount of time. Given a word q in the query tape, say $q = q_1q_2\cdots q_n$ (q_i is th i^{th} digit of q) where $n = |q|$, we consider the real number $z = 0.q$, i.e. $z = 0.q_1q_2\cdots q_n$, to describe the cannon position. As we work over the binary alphabet (with words over Σ^*) q_i is either 0 or 1, and the number z is considered in its binary representation, we call binary form of z to $q_1q_2\cdots q_n$. Working for both the sharp and the smooth experiments, we start to specify the communication protocols already studied concerning time restrictions ([6, 7, 1]).

Protocol 1.1. Error-free protocol
Given a word q in the query tape, the experiment sets the cannon position to the real number z (using an infinite precision).

Protocol 1.2. Error-prone arbitrary precision protocol
Given a word q in the query tape, the experiment sets the cannon position, within a uniform distribution, to a real number in the interval $]z - 2^{|q|}, z + 2^{|q|}[$ (using an arbitrary precision).

Protocol 1.3. Error-prone finite precision protocol
Given a word q in the query tape, the experiment sets the cannon position, within a uniform distribution, to a real number in the interval $]z - \xi, z + \xi[$, where ξ is a fixed positive real number (using a fixed precision).

With "within a uniform distribution" we mean that a real number in the interval is chosen randomly, having all the numbers the same probability of being chose. A deeper study on distributions to the precision of physical experiments can be found in [5]. We call accuracy interval, in the error-prone protocols, to the interval where the experiment chooses the cannon position.

Concerning the error-prone finite precision protocol, without loss of generality, we assume that ξ is small enough to have the interval $]z - \xi, z + \xi[$ embedded in $]0,1[$ at least for $z = 1/2$. So, we are assuming $\xi \leq 1/2$. But sometimes, it might happen that the scatter experiment with fixed precision is used with another cannon positions, when that is the case, we have a more restricted error ξ and the Turing machine only write in the query tape words q which keep $]z - \xi, z + \xi[$ embedded in $]0,1[$, where $z = 0.q$. We also assume, without loss of generality, for computation purposes, that the fixed error ξ is of the form $\xi = 2^{-N}$, for some $N \in \mathbb{N}$.

Remark 2.3.1. *With the error-prone arbitrary precision protocol the length of the accuracy interval is defined by the size of q, the query word. If we want to shrink the interval keeping its centre, we just need to pad the word q with zeros, and each zero shrinks the interval to the half of its length.*

In this case, for some word q with at least one 1, we always have $]z - 2^{|q|}, z + 2^{|q|}[$ embedded in $]0,1[$. If we want to perform a shot with 0 as lower limit to the accuracy interval, we just have to choose a non-empty query word of the form $q = 0\cdots 01$ and the accuracy interval is then $]0, 2^{-(|q|-1)}[$.

These were the communication protocols already used for the computational bounds of the analogue-digital machine concerning time restrictions. We study space restricted scatter machines with these protocols in Chapter 3 and in Chapter 4 we study them for the protocols to be introduced below. With the communication protocols between the digital and analogue parts we can introduce properly the

machines to be studied throughout the dissertation. For the case of a sharp scatter experiment we call analogue-digital sharp scatter machine (ShSM for short) to a Turing machine coupled with a sharp scatter experiment as its oracle. We call error-free ShSM to a ShSM with the error-free protocol, error-prone arbitrary precision ShSM to a ShSM with the error-prone arbitrary precision protocol and error-prone finite precision ShSM to a ShSM with the error-prone finite precision protocol. Analogously to the sharp case, for a smooth scatter experiment we call analogue-digital smooth scatter machine (SmSM for short) to a Turing machine coupled with a smooth scatter experiment. We call error-free SmSM, error-prone arbitrary precision SmSM and error-prone finite precision SmSM to a SmSM with the error-free protocol, error-prone arbitrary precision protocol and error-prone finite precision protocol, respectively.

Remark 2.3.2. *Throughout this dissertation we use decision scatter machines, which have the 'accept' and 'reject' states but not an output tape. Computation scatter machines could also de used but they are not interesting for our purposes.*

To the time restriction cases these protocols are thought as sufficiently powerful, we see that these protocols are equipotent to the new ones which follow (see Proposition 2.3.2), for time restriction cases. But to space restrictions on a scatter machine they may not be sufficient since we are restricting the cannon precision with the space restriction of the query tape. To get a solution on the subject to the polynomial case, we first thought of consider a communication protocol where the analogue part would have stored the cannon position from the previous shot. Like start to shoot at $z = 1/2$, and depending on the result, set the following shot (which is for a future oracle call) at either $z = 1/4$ or $z = 3/4$, and so on. With this method we would not need the query tape at all, or we could use it to count the calls to the oracle or some other purpose. Then, we concluded that this should be equipotent to simply do not restrict the space of the query tape, which would be a lot simpler to deal with. Since in both methods we could get access to exponentially long advices in polynomial space, which is sufficient to decide all languages (see Proposition 1.2.11).

We have then the ultimate protocol we introduce for the purpose of this dissertation, that is mostly like the previous one with the exception that, the query tape is even more distinguished. On this new communication protocol the query tape is not affected by the space restrictions to the scatter machine, so, we can write as much as we need on the query tape as soon as we use it only to query the oracle (the physical experiment). The idea of having a tape not affected by the space restrictions to the machine is already used, for example, to the space bounded reducibility (see [3], §3.8). To the space bounded reducibility the machine reducing a set to another has a bound on the space for the work tapes, but the output tape is not space restricted. In our case the tape which is not restricted is the query tape. With this new protocol, the scatter machine has the same distinguished states to interact with the experiment and distinguished tape, the query tape. The query tape is then restricted in the sense that it must not be used, by the digital part, to perform scratch working. The Turing machine, in its finite control, cannot have the transition function overusing the query tape. The query tape can only be read to decide either what to write in itself or when to change to the 'query' state. The transition function can then be partitioned in two parts, a part which can read any tape but can only write in the query tape and transit to the 'query' state and another part which cannot read the query tape.

All the scatter machines are used satisfying the restrictions to the query tape we just described, and their transition functions could be partitioned. They did not need to satisfy that conditions but they would not have any kind of advantage for using the query tape to other than query purposes. But in this last case is different, since as the query tape is not space restricted, use it for scratch computations could boost the computational power of the digital component of the analogue-digital machine.

We then introduce the three communication protocols (for the three precision assumptions) between the analogue and the digital parts. The protocols are exactly the same once the query word is given, the

difference is in the length properties of the query word.

Protocol 2.1. Error-free protocol
Given a word q in the query tape, the experiment sets the cannon position to the real number z.

Protocol 2.2. Error-prone arbitrary precision protocol
Given a word q in the query tape, the experiment sets the cannon position, within a uniform distribution, to a real number in the interval $]z - 2^{|q|}, z + 2^{|q|}[$.

Protocol 2.3. Error-prone finite precision protocol
Given a word q in the query tape, the experiment sets the cannon position, within a uniform distribution, to a real number in the interval $]z - \xi, z + \xi[$, where ξ is a fixed positive real number.

We call standard communication protocol to the first protocol we specified, already used in [6] for example, and we call space-generalized communication protocol (or generalized communication protocol) to this last one we just specified. We have the same designations for the scatter machines with this new protocol as we have with the other one. We call analogue-digital sharp scatter machine (ShSM for short) and analogue-digital smooth scatter machine (SmSM for short) to a Turing machine coupled with a sharp or a smooth scatter experiment, respectively. We call error-free ShSM, error-prone arbitrary precision ShSM and error-prone finite precision ShSM to a ShSM with the error-free protocol, error-prone arbitrary precision protocol and error-prone finite precision protocol, respectively. The same is to the case of a SmSM, we have the error-free SmSM, the error-prone arbitrary precision SmSM and the error-prone finite precision SmSM. We call, when not already clear, standard analogue-digital scatter machine or generalized analogue-digital scatter machine to distinguish between them.

Concerning the differences between the two protocols we can always obtain the results from a standard scatter machine on a generalized scatter machine. We only need to simulate the standard scatter machine $\mathcal{M}$ on a generalized scatter machine with an additional tape, where $\mathcal{M}'$ (the generalized scatter machine) writes on that tape exactly what it writes in the query tape. Then, when $\mathcal{M}$ reads the query tape $\mathcal{M}'$ reads the additional tape and obtain exactly the same information. Leading us to the following Proposition:

Proposition 2.3.1. *If a standard scatter machine can decide a set $A \in \Sigma^*$, then there is a generalized scatter machine that decides the set A.*

The result holds for either the sharp or the smooth scatter machines, with any precision assumption and for any constraints to the scatter machine (time or space bounds, for example). The reciprocal is not true as we conclude through the dissertation (see Propositions 3.6.4 and 4.1.1).

Proposition 2.3.2. *Concerning scatter machines without space restrictions, a set $A \in \Sigma^*$ is decidable by a generalized scatter machine if and only if is decidable by a standard scatter machine.*

Proof. By Proposition 2.3.1 we have that a set decidable by a standard scatter machine is decidable by a generalized scatter machine.

If a set $A \in \Sigma^*$ is decidable by a generalized scatter machine $\mathcal{M}$ which is not space restricted, then a standard scatter machine $\mathcal{M}'$, behaving like $\mathcal{M}$, can use exactly the same queries $\mathcal{M}$ uses in any of its computations. Since the query tape of $\mathcal{M}'$ is not space restricted as well as any other tape in either $\mathcal{M}'$ or $\mathcal{M}$. Therefore, a standard scatter machine can decide all the sets that a generalized scatter machine can, and the desired equivalence follows. □

As a corollary, of course, if a scatter machine is clocked in polynomial time (it is not space restricted), both the standard and the generalized protocols lead to the same results.

2.4 Scatter machine

The analogue-digital scatter machine, as described earlier, consists of coupling a Turing machine with a scatter experiment. We specified, for the three different precision assumptions, two communication protocols between the two components of the scatter machine, the analogue and the digital parts. We then got, for each one between the standard and the generalized protocols, an error-free protocol and two error-prone ones. With the error-free protocol (infinite precision) we have a deterministic experiment, in the sense that two runs with the same parameter z, the specified cannon position, always have the same result. But with the error-prone protocols (arbitrary and fixed precisions), if we have the vertex position inside the accuracy interval, we have a non-deterministic experiment, we may have different results in two calls with the same parameter z. We have then two different kinds of scatter machine, for which we should define decidability of languages, the deterministic and the probabilistic kind.

We always consider a deterministic Turing machine as the digital component for a scatter machine, its possible probabilistic behaviour is always provided by the experiment (see Proposition 2.4.1). We can use an error-prone protocol to get a deterministic scatter machine, we just have to be careful to guarantee that no call of probabilistic answer is made, as it actually happens on the lower bounds when we want to find the advice with the arbitrary precision (see Propositions 3.6.2 and 3.7.2).

For a deterministic scatter machine, as we have for a deterministic Turing machine, we say that a word $w \in \Sigma^*$ is accepted if the scatter machine halts the computation for the input w in the 'accept' state.

Definition 2.4.1. *We say that a deterministic scatter machine $\mathcal{M}$ decides a set $A \subseteq \Sigma^*$, if for all $w \in \Sigma^*$, $\mathcal{M}$ accepts w if and only if $w \in A$.*

For a probabilistic scatter machine, we use the bounded error probability decision criterion (see Definition 1.2.1).

Definition 2.4.2. *We say that a probabilistic scatter machine $\mathcal{M}$ decides a set $A \subseteq \Sigma^*$ if, there is a rational number ϵ with $0 < \epsilon < 1/2$ such that, for every $w \in \Sigma^*$ (a) if $w \in A$, then $\mathcal{M}$ rejects w with probability at most ϵ and (b) if $w \notin A$, then $\mathcal{M}$ accepts w with probability at most ϵ.*

It is the same decision criterion as to define BPP or $BPPSPACE$. We say that a probabilistic scatter machine accepts a word $w \in \Sigma^*$ if it halts the computation for the input w in the 'accept' state. But if we run the same probabilistic scatter machine on w twice, it is possible to have w accepted exactly once. Given these definitions, we have a bounded error probabilistic scatter machine, but we call it probabilistic scatter machine, as we do for the bounded error probabilistic Turing machines through the dissertation.

The probabilistic uniform class BPP has the property that given a set $A \in BPP$, witnessed by a probabilistic Turing machine $\mathcal{M}$ with constant ϵ, we have a probabilistic Turing machine (based on $\mathcal{M}$) that decides A with error probability at most $2^{-p(n)}$ for each input w with $|w| = n$. Analogously as to the class BPP, we assume the same result for general probabilistic scatter machines, since as the proof technique consists in running the Turing machine enough times we can do the same with the scatter machines. We use the assumption, without loss of generality, that a probabilistic scatter machine has its bounded error less or equal to $1/4$.

Remark 2.4.3. *If a probabilistic scatter machine has a deterministic behaviour and decides a set $A \subseteq \Sigma^*$ with the deterministic criterion, then it also decides A with the bounded error probability decision criterion. Since if a word $w \in A$, the machine rejects w with probability $0 < \epsilon$, and if $w \notin A$, the machine accepts w with probability $0 < \epsilon$.*

With respect to the lower bounds we may need to simulate a probabilistic advice Turing machine on a scatter machine, this subject was already discussed in [6, 8, 9]. It would be easy to just consider a

probabilistic Turing machine in the digital part of the scatter machine to the effect, but we can do it in a different way, more elegant and appropriate to the power of the scatter experiment.

First, we have to notice that a scatter experiment with an error-prone protocol can be used as a coin. If we consider a cannon position close enough to the vertex position so that the vertex position would be inside the accuracy interval, we have a probabilistic experiment with two or three possible answer cases with probability greater than zero (depending on if we have a sharp or a smooth experiment). We can always consider the union of some answer cases in order to have two hypothesis with complementary probability values. With the arbitrary precision assumption we can always use the query $q = 1$ which produces the accuracy interval $]0, 1[$ to have a probabilistic coin toss. With the fixed precision assumption we can choose a query word that gets the vertex position inside the accuracy interval based on the vertex position and the fixed error of the protocol, resulting in a probabilistic coin toss. In both cases we can obtain an interval $]\delta, 1-\delta[$, for some δ with $0 < \delta < 1/2$, that has at least one of the complementary probability values inside it.

It may happen that we obtain a fair coin toss from the scatter experiment, but generally, we obtain a biased coin toss. We have a result, already stated in [1] for example, that states that a biased coin can be used to generate a sequence of fair coin tosses of length n.

Proposition 2.4.1. *Given a biased coin with probability of heads $p \in]\delta, 1-\delta[$, for some δ with $0 < \delta < 1/2$, and a real number $\lambda \in]0, 1[$, we can generate a sequence of independent fair coin tosses of length n with a linear amount of biased coin tosses on n, up to probability λ.*

We can then simulate a probabilistic Turing machine with a probabilistic scatter machine. If we have a scatter machine bounded in polynomial space we may need to have a sequence of 2^n independent fair coin tosses. We cannot generate them all at once since we cannot write a sequence of length 2^n as we are bounded in polynomial space. If we generate one fair coin toss at a time, we need each probability of success greater than $1 - 2^{-n-4}$ in order to have the probability of success for all the 2^n coin tosses greater than $1 - 2^{-4}$, for example. We need then an exponential amount of unfair coin tosses (calls to the oracle) to obtain each fair coin toss (see [6], §7, Lemma 7.7). As we can use the smooth scatter experiment with a constant time schedule and the sharp one has already a constant physical time, we do not need more than an exponential time to generate all the 2^n fair coin tosses.

With respect to the upper bounds, we need to simulate the probabilistic answers from the oracle on a probabilistic Turing machine, the subject is already discussed in [7]. For each call to the oracle that we may need to simulate, we must have the probability of the answers from the oracle as dyadic rationals, otherwise it would not be possible to simulate them on a general probabilistic Turing machine.

We do not have a probabilistic scatter experiment with the infinite precision assumption, we only have it when using the error-prone protocols. Based on the behaviour we assume for the error-prone protocols we always have an accuracy interval centred on z (where $z = 0.q$ and q is the query word) with dyadic boundaries $z - 2^{-N}$ and $z + 2^{-N}$, for some $N \in \mathbb{N}$.

Given a dyadic number with polynomially sized representation (a dyadic vertex position for example) we can check in polynomial time whether it is inside the accuracy interval or not. When we already know that the answer from the oracle is probabilistic, i.e. (in the sharp case) the dyadic vertex position y_d is inside the accuracy interval or (in the smooth case) at least one of the boundary numbers is (see Section 3.5), we can (in the sharp case) compute both the probability for the answer being left and right, which correspond to $y_d - (z - 2^{-N})$ and $z + 2^{-N} - y_d$, respectively. Thus, we can simulate digitally an event with probability $(y_d - (z - 2^{-N}))/2^{-N+1}$ to simulate the oracle call (see [7]). Analogously, in the smooth case we can simulate, if necessary, an event with three possible outcomes, each one with dyadic probability. We can use a fair coin toss to produce any event with dyadic probability, which takes

n units of time where n comes from the dyadic probability $k/2^n$, for some k with $0 \leq k \leq 2^n$ (n is linear on the dyadic number representation). We can also use fair coin tosses to produce a probabilistic event with three possible outcomes, as for example using two coin tosses. Of course we assume that on a general probabilistic Turing machine, we have a fair coin toss which takes exactly one transition.

The cost for the computations needed above is polynomial on the size of the dyadic number representations. So, if we have polynomially sized representations for y_d, z and 2^{-N}, which we do if we have polynomially sized words q and describing y_d, all the simulation takes a polynomial time. The same is, in the smooth case, with polynomially sized representations for the dyadic boundary numbers.

Chapter 3

Standard scatter machine

Throughout this chapter we intend to study the computational power of the scatter machine bounded in polynomial space with the standard communication protocol. We stablish the computational power for both the sharp and the smooth scatter experiments with any of the three precision assumptions.

At first we present auxiliary results needed in the proofs and only then the proofs itself. This communication protocol between the analogue and the digital part is the protocol used for the study of the scatter machine bounded in polynomial time. Here we adapt the results obtained in [6, 7, 13] to the case of bounded space.

We can start with noticing that the scatter machine bounded in polynomial space can decide all the sets that the scatter machine clocked in polynomial time decide. Since a scatter machine clocked in polynomial time cannot use more than a polynomial space, the scatter machine bounded in polynomial space can simulate any scatter machine clocked in polynomial time, because a scatter machine bounded in polynomial space can use an exponential amount of time (see Proposition 3.4.1).

We conclude then that the complexity classes that a scatter machine clocked in polynomial compute (see [10]) are at least embedded in the lower bounds for the scatter machine bounded in polynomial space.

We can state then the following lower bounds for the analogue-digital sharp scatter machine bounded in polynomial space:

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$P/poly$	$P/poly$	$BPP//\log\star$

Table 3.1: Standard communication protocol ShSM

And we can state the following lower bounds for the analogue-digital smooth scatter machine bounded in polynomial space:

	Infinite	Arbitrary	Fixed
Lower Bound	$P/\log\star$	$BPP//\log\star$	$BPP//\log\star$

Table 3.2: Standard communication protocol SmSM

The proof techniques presented below are adaptions from the techniques used to proof the results above, as well as other results concerning computational power of analogue-digital machines.

3.1 Vertex position

In this section we present a useful coding to proof the lower bounds of the computational power of the scatter machine. With the aim of simulate an advice Turing machine on a scatter machine, we should be able to code the information of an advice function in the vertex position.

To code a prefix advice function as a vertex position

- We consider the function $c : \Sigma^* \to \Sigma^*$, where the encoding $c(w)$ is obtained from the word $w \in \Sigma^*$ by replacing every 0 by 100 and every 1 by 010.
- Given a prefix advice function f, we consider the function $\overline{x(f)} : \mathbb{N} \to \Sigma^*$ such that $\overline{x(f)}(n) = c(f(n))$.
- We call $x(f)$ to the limit sequence[1] of $\overline{x(f)}$.
- And finally, we define the real number $y \in]0, 1[$ such that $y = 0.x(f)$.

We can then place the vertex of a scatter experiment at y.

Example.
$c(\varepsilon) = \varepsilon$.
If $f(0) = 0\,1\,0\,0\,1\,1$ *then* $\overline{x(f)}(0) = c(f(0)) = 1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0$,
$x(f) = 1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0 \cdots$ *and* $y = 0.1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0 \cdots$.

We use the encoding c so that we would not have to distinguish between the two possible binary expansions of dyadic[2] rationals. No dyadic rational can occur as an $y = 0.x(f)$, since $x(f)$ was constructed not to have the occurrence of 0000 or 1111, and a dyadic number would have its binary expansion ending in either all zeros or all ones. As for example, $1/2$ has the two possible binary expansions $0.10000\cdots$ and $0.01111\cdots$. This ambiguity (over Σ^*) is limited to the dyadic rationals, any other real number has a unique binary expansion.

The numbers which can occur as an $y = 0.x(f)$ are all elements of the so-called Cantor Set. This kind of codification is often used to distinguish between close values. The Cantor Set has been considered to boost computation in the neural net framework as in [20, 21] and in other computational models working with real valued parameters such has in [15, 23].

To find $f(n)$ from $x(f)$, when f is a prefix advice function, we read the digits of $x(f)$ in triples. Whenever the triple 100 is read we have a 0 in the advice and whenever the triple 010 is read we

[1] The limit sequence of $\overline{x(f)}$ is the infinite word over Σ^* for which any $\overline{x(f)}(n)$ is a finite prefix of it.
[2] A dyadic number is a rational of the form $n/2^m$ for some $n \in \mathbb{Z}$, $m \in \mathbb{N}$.

have a 1. When dealing with a prefix advice function $f \in poly$, as we have a polynomial p such that $|f(n)| < p(n)$, it is enough to consider the first $3 \cdot (p(n) - 1)$ digits of $x(f)$. As we can consider $f \in poly$ as an advice function of known size, we can easily get exactly $f(n)$ from $x(f)$ and no more than that.

To find $x(f)$ by means of the scatter experiment, the Turing machine applies the linear search method. This is a linear search method with additional information we have about the vertex position. We know that either $y = 0.100\cdots$ or $y = 0.010\cdots$, so, if we set the cannon to shoot in the middle, at $z = 0.011$, then if the answer is right it is because $y < z$ and if the answer is left it is because $y > z$. The first digit of the advice is then known, i.e. the first triple of $x(f)$. To get the i^{th} digit of the advice, having read the first $i-1$ digits, we set the cannon at $z = 0.x(f)_1x(f)_2\cdots x(f)_{i-1}011$, where $x(f)_i$ is the i^{th} triple in $x(f)$ and follow the same reasoning.

This search method leads us to the following algorithm:

Algorithm 1: Linear search method

Data: Number of triples we want, m
Set $q' := \varepsilon$;
while $|q'| < 3 \cdot n$ **do**
  Make an oracle call with the query word $q = q'011$;
  if *The answer from the oracle is right* **then**
    Set $q' := q'010$;
  end
  if *The answer from the oracle is left* **then**
    Set $q' := q'100$;
  end
end
return q' (q' is a prefix of $x(f)$)

Remark 3.1.1. *When a scatter machine is using this linear search method the query coincide with $x(f)$ but at the last three digits.*

The method described needs a space equal to three times the size of the advice, which is polynomial on the size of the input for a polynomial advice. Also, in the sharp case of the experiment the same amount of time is enough, since we need to perform a call to the oracle for each digit of the advice and the oracle calls take a constant time. In the smooth case we need an exponential amount of time on the query size for each call to the oracle. To obtain m triples of $x(f)$ we need less than 2^{3m+3} units of time, since as at the i^{th} call $|y - z| > 2^{-3i-2}$, we conclude that the i^{th} call needs at most 2^{3i+2} units of time to be answered.

If we consider an exponential advice, as with the generalized communication protocol, we need an exponential space to write the query, and as we need to perform an exponential amount of queries to obtain the advice we need an exponential amount of time even for the sharp scatter machine. In the smooth case we need a time with magnitude order of 2^{2^n} (see Propositions 4.2.2 and 4.2.3).

Proposition 3.1.1. *Let q be a query to the linear search method of a scatter machine with vertex position $y = 0.x(f)$, for some prefix advice function f. The error-prone arbitrary precision scatter experiment, with the precision set to at least $2^{-|q|-3}$, has the same result as the error-free scatter experiment. In the smooth case of the experiment, the time is at most doubled.*

Proof. Since with the linear search method, the scatter experiment is set to fire the cannon at a distance of at least $2^{-|q|-2}$ from the vertex position, if we set the error to be at most $2^{-|q|-3}$, the shot is performed

at at least $2^{-|q|-3}$ from the vertex position, on the same side. So, we have the same answer from the oracle.

In the smooth case, with $|y-z| > 2^{-|q|-2}$ we had the time to get an answer bounded by $2^{|q|+2}$. Now, with $|y-z| > 2^{-|q|-3}$ we have the time bounded by $2^{|q|+3}$, which is twice as much. □

Remark 3.1.2. *The time in the smooth case can also be reduced by a third. In the error-prone case (arbitrary precision), we have to pad the query word q with three 0's in order to get the desired accuracy interval, and to use the linear search method is not necessary to increase the number of oracle calls.*

For the proof of the upper bounds, as explained below (see Section 3.3), the scatter machine may have a vertex position not in the Cantor set. The scatter machine then needs a measurement algorithm to a more general case, this subject is already studied in [11]. An analogous algorithm to the specified one for determining a digit of the binary form of the vertex position at a time would suffice, and would also work only with non-dyadic vertex positions.

An important result on the subject is that an analogue-digital machine, with a general linear search method, can measure any measurable real valued parameter that could be measured by any analogue-digital algorithm (see [11]).

3.2 Find the vertex position with fixed precision

With the fixed precision assumption we must use a different technique, since with a fixed precision we are only able to approximate the vertex position until a finite amount of digits. We make then use of the Chebyshev's inequality to obtain an accurate enough approximation of the vertex position.

Proposition 3.2.1. *Let $z \in]0,1[$ be a real number and $y = 0.x(f)$ a number in the Cantor Set, for some prefix advice function f. If $|y-z| < 2^{-n-5}$ then the first n digits of the binary form of z and y coincide.*

Proof. If at least one of the first n digits of the binary expansion of z and y differ, lets assume that the i^{th} digit is different and the previous $i-1$ are equal, for $1 \leq i \leq n$. Then, in the worst case scenario, either $z_i = 1$ followed by a sequence of 0's and $y_i = 0$ followed by a sequence of 1's, or $z_i = 0$ followed by a sequence of 1's and $y_i = 1$ followed by a sequence of 0's. Where z_i and y_i are the i^{th} digit in the binary form of z and y, respectively. In any of the two cases, as $y = 0.x(f)$ for some prefix advice function f, the sequence of 0's or 1's in y has at most 3 digits and thus, we conclude that $|y-z| \geq 2^{-i-5}$. So, if $|y-z| < 2^{-n-5}$, then the first n digits of the binary form of z and y are equal. □

A scatter machine must then find a real number close enough to the vertex position so that the first wanted amount of digits from both numbers are equal.

Proposition 3.2.2. *Given a prefix advice function f, an error-prone finite precision ShSM $\mathcal{M}$ can obtain m triples of $x(f)$ with error probability smaller than 2^{-c} with 2^{6m+8+c} oracle calls.*

Proof. Note that if $y' = 1/2 + \xi - 2\xi y$ is the vertex position of $\mathcal{M}$, where $y = 0.x(f)$ and ξ is the fixed error of the protocol, and the cannon position is set at $z = 1/2$, then the particle has probability y of being detected in the right collecting box.

Let R be the number of particles detected in the right collecting box after s shots, we can then estimate y as R/s. So that we could apply Proposition 3.2.1 we want s such that $|y - R/s| < 2^{-3m-5}$. Since R is a random variable with expected value $\mu = sy$ and variance[3] $\sigma^2 = sy(1-y)$, using Chebyshev's inequality, we conclude that $\Pr(|\mu - R| \geq k\sigma) \leq k^{-2}$, for any $k \in \mathbb{R}^+$. Considering $k = 2^{-3m-5}s\sigma^{-1}$, we obtain $\Pr(|y - R/s| < 2^{-3m-5}) > 1 - 2^{6m+10}s^{-1}y(1-y)$. As $y(1-y) \leq 2^{-2}$, since $y \in]0,1[$, if we consider $s = 2^{6m+8+c}$, we get $\Pr(|y - R/s| < 2^{-3m-5}) > 1 - 2^{-c}$. □

[3] Of course, σ is the positive value such that $\sigma^2 = sy(1-y)$.

To obtain the first $|f(n)|$ triples of $x(f)$ with error probability smaller than δ we should choose c such that $2^{-c} \leq \delta$ and perform $2^{6|f(n)|+8+c}$ oracle calls.

If $f \in poly$, we need an exponential amount of oracle calls on n. So, to write and read R and s and to compute R/s until $3 \cdot |f(n)|$ digits, we need a polynomial space on n. An exponential time on n is enough, since there is a division algorithm that runs in polynomial time.

Proposition 3.2.3. *Given a prefix advice function f, an error-prone finite precision SmSM $\mathcal{M}$ can obtain m triples of $x(f)$ with error probability smaller than 2^{-c} with 2^{6m+8+c} oracle calls.*

Proof. Note that, if $y' = 1/2 + \xi - 2\xi y$ is the vertex position of $\mathcal{M}$, where $y = 0.x(f)$ and ξ is the fixed error of the protocol, and the cannon position is set at $z = 1/2$, then the particle has probability y of being detected in the right collecting box. As we are using the smooth case of the experiment, if we use the time schedule, we have three possible outcomes with probability greater than zero. For any fixed time schedule T, there is a number $\eta \in]0, 1]$ such that, if the shot $z = 1/2$ is performed within $]y' - \eta, y' + \eta[$, the time schedule halts before the oracle gets to an answer. We want $\eta < 2\xi(1 - y)$ and $\eta < 2\xi y$ so that the interval $]y' - \eta, y' + \eta[$ would be embedded in $]1/2 - \xi, 1/2 + \xi[$. Since $1/4 < y < 5/8$, as y is of the form $0.x(f)$ for some prefix advice function f, it is enough to consider a time schedule which makes $\eta < \xi/2$, a constant time schedule is enough. So, the probability of the particle being detected in the right collecting box before the time schedule halts is $(2\xi y - \eta)/2\xi$, and the probability of the time schedule halts before the particle being detected in some collecting box is η/ξ.

Considering the time schedule T in use. Let R be the number of particles detected in the right collecting box before the time schedule halts after s shots and U the number of particles that were not detected in any of the collecting boxes until the time schedule halts, after the same s shots. We can then estimate y as $(R + U/2)/s = (2R + U)/2s$.

So that we could apply Proposition 3.2.1, we want s such that $|y - (2R + U)/2s| < 2^{-3m-5}$. As $2R + U$ is a random variable with expected value $\mu = 2sy$ and variance $\sigma^2 = s(4y(1 - y) - \eta/\xi)$, using Chebyshev's inequality, we conclude $\Pr(|\mu - (2R + U)| \geq k\sigma) \leq k^{-2}$, for any $k \in \mathbb{R}^+$. Considering $k = 2^{-3m-4}s\sigma^{-1}$, we obtain $\Pr(|y - (2R + U)/2s| < 2^{-3m-5}) > 1 - 2^{6m+8}s^{-1}(4y(1 - y) - \eta/\xi)$. As $3/4 < 4y(1 - y) \leq 1$ and $0 < \eta/\xi < 1/2$ since $1/4 < y < 5/8$ and $0 < \eta < \xi/2$, respectively. If we consider $s = 2^{6m+8+c}$, we get $\Pr(|y - (2R + U)/2s| < 2^{-3m-5}) > 1 - 2^{-c}$. □

To obtain the first $|f(n)|$ triples of $x(f)$ with error probability smaller than δ we should choose c such that $2^{-c} \leq \delta$ and perform $2^{6|f(n)|+8+c}$ oracle calls.

If $f \in poly$, we need an exponential amount of oracle calls on n. So, to write and read R and s and compute R/s until $3 \cdot |f(n)|$ digits, we need a polynomial space on n. An exponential time on n is enough, since there is a division algorithm that runs in polynomial time.

From the results stated in the last two Propositions we can conclude that for the fixed precision protocol, both the sharp and the smooth scatter machines can obtain the same information with the same error probability for a sufficiently centred vertex position.

3.3 Sparse oracles

As described above to the lower bounds, we construct the vertex positions so that they would not be dyadic numbers. To the upper bounds, as in general, the vertex of a scatter experiment can happen to be placed at dyadic positions. We use here proof techniques, not as simple as the to the lower bounds, for which we need to introduce some other concepts. We aim to simulate scatter machines in general advice Turing machines, where we use the advice to answer the oracle calls. With the standard communication protocol we only need to simulate polynomially sized oracle calls, since we are studying scatter machines bounded in polynomial space.

Concerning the sharp scatter experiment, in the error-prone protocols we assume that if the particle hits the vertex, it deterministically enter the same collecting box (the right or the left one). The proofs are analogous to other assumptions, we could just have to use ternary trees instead of binary ones. We use the same proof technique for any of the precision assumptions with te smooth scatter experiment, but in this case we always need ternary trees. With the error-free protocol (sharp scatter experiment) we use a different proof method analogous to the one in [6]. We do not need to assume what happens when the particle hits the vertex, but we do have to assume that the behaviour is also deterministic i.e. if the particle hits the vertex, the same result always happens.

To code a vertex position as a sparse oracle

Given a real number $y \in]0,1[$, considered with its binary expansion in the terminating form (i.e. 0.01 instead of $0.001111\cdots$), and an increasing total function $f : \mathbb{N} \to \mathbb{N}$, we construct a sparse[4] set. For example, lets consider $y = 0.0110001010110100\cdots$ and the polynomial function $f(n) = (n+1)^2$.

To construct the set O_y^f, first we consider the infinite word over Σ^* that represents the binary form of y, possibly ending in all zeros, and then, after each $f(n)$ digits in the infinite word, we append either the symbol '$=$' or the symbol '$>$', depending on if y is equal to its binary expansion truncated at that point or not, respectively.

In our example, we would get, $0{>}110{>}00101{>}0110100{=}\cdots$ or $0{>}110{>}00101{>}0110100{>}\cdots$, depending on if the binary form keeps on with all zeros or not, respectively. Since $f(0) = 1$, $f(1) = 4$, $f(2) = 9$ and $f(3) = 16$, for the function chose as example ($f(n) = (n+1)^2$). Note that the digit $f(n) + n + 1$, for any $n \in \mathbb{N}$, in any word from $\mathcal{O}_y^f$ is always a '$>$' or '$=$' sign.

If an '$=$' sign appears, we know that from there on, no 1 ever occurs again and only the symbol '$=$' occurs besides the zeros. We consider the set O_y^f as the set with all (finite) prefixes of the resulting word. O_y^f is clearly a sparse set (it can even be seen as a tally[5] set).

Proposition 3.3.1. $PSPACE/poly = \bigcup_{O\ sparse} PSPACE(O)$.

Proof. First, lets assume $A \in PSPACE/poly$. If $A \in PSPACE/poly$, there exists $B \in PSPACE$ and an advice function $f \in poly$ such that, for every $w \in \Sigma^*$, $w \in A$ if and only if $\langle w, f(|w|)\rangle \in B$. Let $\mathcal{M}_B$ be an advice Turing machine which decides B in polynomial space.

Let O be the set $\{\langle 0^n, x\rangle \in \Sigma^* : x \text{ is a prefix of } f(n)\}$. O is a sparse set since, for any $n \in \mathbb{N}$, O has at most n words of size n. Since for any $n \in \mathbb{N}$ and $k \in \{0, 1, \ldots, n-1\}$, there is at most one prefix x of $f(k)$ that makes $\langle 0^k, x\rangle$ with size n.

Consider the oracle Turing machine $\mathcal{M}$, which for an input w, uses the usual oracle O to obtain $\langle 0^{|w|}, f(|w|)\rangle$ (see [3], Chapter 5, Theorem 5.26). Afterwards, $\mathcal{M}$ simulates $\mathcal{M}_B$ over $\langle w, f(|w|)\rangle$ and accepts w if and only if $\mathcal{M}_B$ accepts $\langle w, f(|w|)\rangle$. Thus, $\mathcal{M}$ decides A. As to get $\langle 0^{|w|}, f(|w|)\rangle$ and write $\langle w, f(|w|)\rangle$ a polynomial space on $|w|$ is enough, and also, to simulate $\mathcal{M}_B$ on $\langle w, f(|w|)\rangle$ a polynomial

[4] A set is sparse if it has at most $p(n)$ words of length n, $\forall n \in \mathbb{N}$, for some polynomial p.
[5] A set is tally if it has at most one word of length n, $\forall n \in \mathbb{N}$.

space on $|\langle w, f(|w|)\rangle|$, which is polynomial on $|w|$, is enough. We can conclude that $\mathcal{M}$ operates in polynomial space on $|w|$ and so, that $A \in PSPACE(O)$.

The computation time of $\mathcal{M}$ on w is the time that $\mathcal{M}_B$ needs for the input $\langle w, f(|w|)\rangle$ plus the time needed to get $\left\langle 0^{|w|}, f(|w|)\right\rangle$ and write $\langle w, f(|w|)\rangle$. Except for the simulation, this time is polynomial on $|w|$.

Now, conversely, lets assume $A \in PSPACE(O)$, for some sparse set O. So, there is an oracle Turing machine $\mathcal{M}$ which decides A in polynomial space p with the usual oracle O.

Let $m_n \in \mathbb{N}$ be the number of words of length n in O, so, as O is sparse, there is a polynomial q such that $m_n \leq q(n)$, for any $n \in \mathbb{N}$. Let also the words $w_1^n, w_2^n, \ldots, w_{m_n}^n$ be all the words of length n in O. And let finally f be the advice function such that

$$f(n) = w_1^1 \# w_2^1 \# \cdots \# w_{m_1}^1 \# w_1^2 \# w_2^2 \# \cdots \# w_{m_2}^2 \# \cdots \# w_1^{p(n)} \# w_2^{p(n)} \# \cdots \# w_{m_{p(n)}}^{p(n)}$$

$f \in poly$ since, as $m_i \leq q(i)$, for any $i \in \mathbb{N}$, we have $f(n) \leq \sum_{i=1}^{p(n)} (i+1)q(i) - 1$ which is a polynomial on n.

Consider the advice Turing machine $\mathcal{M}'$, which for an input $\langle w, f(|w|)\rangle$, behaves exactly like $\mathcal{M}$ for the input w except when the oracle calls. Where for a query q uses the advice $f(|w|)$ to give the same answer as the oracle (if q is in $f(|w|)$ answers yes, otherwise answers no). For an input w, $\mathcal{M}$ can only write in the query tape a word q with $|q| \leq p(|w|)$. So, $\mathcal{M}'$ can answer any query it may need for a computation of $\mathcal{M}$. We conclude then that $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ if and only if $\mathcal{M}$ accepts w. To simulate $\mathcal{M}$ on w, $\mathcal{M}'$ needs a polynomial space on $|\langle w, f(|w|)\rangle|$, since $\mathcal{M}$ is space bounded by a polynomial, and to have the answers from the oracle, $\mathcal{M}'$ would need no further space. Therefore, we conclude that $A \in PSPACE/poly$.

The computation time $\mathcal{M}'$ takes on $\langle w, f(|w|)\rangle$ is the same time $\mathcal{M}$ takes on w plus the time needed to get the answers from the oracle. Except for the simulation, this time is polynomial on $|w|$. □

With this last result we use a sparse oracle like O_y^f to prove the upper bound for the error-free ShSM (Proposition 3.6.4). Given an oracle Turing machine, with oracle O_y^f for some increasing total function f and real number $y \in]0, 1[$, to find the word of size n in O_y^f (O_y^f is a tally set), it must find al the smaller words in O_y^f since they are a prefix of the wanted word. Lets call d_i to the i^{th} digit of the word. To find the i^{th} digit of the word, when already known the first $i-1$ digits, the oracle Turing machine must query the oracle with the word $d_1 d_2 \cdots d_{i-1} 0$ and conclude the i^{th} digit is 0 if and only if the answer is yes; if $i \neq f(j) + j + 1$ for any $j \in \mathbb{N}$. If $i = f(j) + j + 1$ for some $j \in \mathbb{N}$ (and the first $i-1$ digits are known); the oracle Turing machine must query the oracle with the word $d_1 d_2 \cdots d_{i-1} >$ and conclude the i^{th} digit is $>$ if and only if the answer is yes.

We need an increasing function f so that the condition $i = f(j) + j + 1$ for some $j \in \mathbb{N}$ can be decided by a Turing machine.

3.4 Probabilistic trees

As already discussed in Chapter 1, we can see the computation tree of a Turing machine as a binary tree. In the case of a scatter machine, if we consider an experiment with two possible answers, we can still see its computation tree as a binary tree, but if we consider an experiment with three possible answers we have to consider a ternary tree for its computation tree. For the sharp cases, we are assuming that we have just two possible answers. Even if we would consider three possible answers, one of them would always be assigned with probability zero.

Concerning the depth of the computation trees, which is a relevant subject to the upper bound proof techniques, we can use an analogous reasoning to proof of Proposition 2.1.2 to state that the depth of a computation tree of an analogue-digital machine bounded in polynomial space is at most exponential. Note that the bound we give there to the possible number of configurations is not possibly to accomplish, i.e. we do not have as much configurations to reach in the same computation of the Turing machine. Here, to avoid unnecessary constants, we say that the limit to the depth of an analogue-digital machine bounded in polynomial space p is $2^{p(n)}$ for an input of size n. For a different limit, also exponential, all the results would be analogous, possibly with different constants but keeping the same magnitude order.

Proposition 3.4.1. *An analogue-digital machine bounded in polynomial space p, for an input of size n, has the depth of its computation tree limited by $2^{p(n)}$.*

A probabilistic tree is a weighted tree where the sum of the weights of the edges from the same parent is one. Given a tree $\mathcal{T}$, an assignment of probabilities for $\mathcal{T}$, α, is a map $\alpha : E_{\mathcal{T}} \to [0,1]$, where $E_{\mathcal{T}}$ is the set of edges of $\mathcal{T}$, that assigns a probability to each edge of $\mathcal{T}$, making $\mathcal{T}$ into a probabilistic tree. We call $\mathcal{T}^{\alpha}$ to the this probabilistic tree and $\rho(\mathcal{T})$ to the set of all possible assignments of probabilities for $\mathcal{T}$.

Given a rooted tree, we call total path to a path from the root to a leaf of the tree, and given a probabilistic rooted tree, we call probability of the total path to the product of the weights along the path.

Note that given a rooted tree $\mathcal{T}$, we can consider a collection of total paths from $\mathcal{T}$, having then a rooted subtree of $\mathcal{T}$ ($\mathcal{T}_s$).

Definition 3.4.1. *Given a probabilistic rooted tree $\mathcal{T}^{\alpha}$, consider any weighted rooted tree consisting on a collection of total paths from $\mathcal{T}^{\alpha}$ and call it $\mathcal{T}_s^{\alpha}$. We say that the probability of $\mathcal{T}_s^{\alpha}$, denoted by $P(\mathcal{T}_s, \alpha)$, is the sum of the probabilities of the total paths of $\mathcal{T}_s^{\alpha}$, for all its total paths.*

Remark 3.4.2. *Given a probabilistic rooted tree $\mathcal{T}$, if we consider the subtree $\mathcal{T}_s$ consisting of all its total paths, then $P(\mathcal{T}_s, \alpha) = 1$. $P(\mathcal{T}, \alpha)$ is always 1.*

We denote by $\mathcal{T}_{n,m}$ the set of all n-ary rooted trees of height less or equal to m.

Given a tree $\mathcal{T}$, we define a distance among the assignments in $\rho(\mathcal{T})$, which we denote by $d_{\mathcal{T}}$ (or usually just d). This distance is given by the maximum of the absolute values of the differences of the probabilities over each edge. Given $\alpha, \alpha' \in \rho(\mathcal{T})$

$$d_{\mathcal{T}}(\alpha, \alpha') = \max\{|\alpha(e) - \alpha'(e)| : e \in E_{\mathcal{T}}\}.$$

We can then define a set of useful functions.

Definition 3.4.3. *Given $n \in \mathbb{N}_+$, we define the function $f_n : \mathbb{N} \times [0,1] \to [0,1]$ as*

$$f_n(m,k) = \max_{\mathcal{T} \in \mathcal{T}_{n,m}} \sup\{|P(\mathcal{T}, \alpha) - P(\mathcal{T}, \alpha')| : \forall \alpha, \alpha' \in \rho(\mathcal{T}) : d_{\mathcal{T}}(\alpha, \alpha') \leq k\}.$$

Remark 3.4.4. *For a n-ary rooted tree $\mathcal{T}$ of depth less or equal to m, if $d(\alpha, \alpha') \leq k$, for all $\alpha, \alpha' \in \rho(\mathcal{T})$, then $|P(\mathcal{T}, \alpha) - P(\mathcal{T}, \alpha')| \leq f_n(m,k)$.*

Proposition 3.4.2. $f_2(m,k) \leq 2mk.$

Proof. See [7], §2, Proposition 2.1.

Or this is an immediate corollary of Proposition 3.4.4, since $2-1<2$. □

Proposition 3.4.3. *Let $\mathcal{M}$ be an error-prone (arbitrary or finite precision) ShSM that decides the set $A \subseteq \Sigma^*$ in polynomial space p, with error probability $\epsilon \leq 1/4$. If $\mathcal{M}'$ is an error-prone ShSM that, for an input w with $|w| = n$, behaves exactly like $\mathcal{M}$ except when the oracle calls. Where for any call, the probability of the answer differs at most $2^{-p(n)-4}$. Then, $\mathcal{M}'$ decides A with error probability $\epsilon' \leq 3/8$.*

Proof. Given a word $w \in \Sigma^*$, consider the computation tree $\mathcal{T}_w$ of $\mathcal{M}$ for the input w,[6] and let $\alpha, \alpha' \in \rho(\mathcal{T}_w)$ be the assignments that give to the edges of $\mathcal{T}_w$ the probabilities given by the experiments of $\mathcal{M}$ and $\mathcal{M}'$, respectively. Consider the subtree of $\mathcal{T}_w$ consisting on all the total paths from $\mathcal{T}_w$ that correspond to a wrong computation[7] of $\mathcal{M}$ when deciding A, and call it $\mathcal{T}_w^{\times}$.

As $\mathcal{M}$ is space bounded by the polynomial p, for an input w, the depth of $\mathcal{T}_w$ is limited by $2^{p(|w|)}$ (see Proposition 3.4.1), and as we considered $\mathcal{T}_w$ as a binary tree. Using Proposition 3.4.2, since then $|P(\mathcal{T}_w^{\times}, \alpha) - P(\mathcal{T}_w^{\times}, \alpha')| \leq f_2(m,k) \leq 2mk$, where m and k are the depth and the biggest difference for any probability of $\mathcal{T}_w^{\times}$, respectively, we have

$$\begin{aligned} P(\mathcal{T}_w^{\times}, \alpha') &\leq P(\mathcal{T}_w^{\times}, \alpha) + |P(\mathcal{T}_w^{\times}, \alpha) - P(\mathcal{T}_w^{\times}, \alpha')| \\ &\leq \epsilon + 2 \cdot 2^{p(n)} \cdot k \\ &\leq 2^{-2} + 2^{p(n)+1} k. \end{aligned}$$

So, if we have k such that $2^{p(n)+1}k \leq 2^{-3}$, we get $P(\mathcal{T}_w^{\times}, \alpha') \leq 3/8$. Therefore, for any word $w \in \Sigma^*$, $\mathcal{M}'$ make the same decision as $\mathcal{M}$ with error probability $\epsilon' \leq 3/8$ instead of $1/4$. □

Concerning Proposition 3.4.2 we can also get a more general result. Needed in the dissertation to the smooth scatter experiment cases, since we need ternary trees instead of binary ones. This result is already stated in [1, 5].

Proposition 3.4.4. $f_n(m,k) \leq (n-1)mk.$

Proof. The proof follows by induction on m. The result is valid for $m = 0$, since the only tree in any of the sets $\mathcal{T}_{n,0}$ has zero edges.

Lets then assume that $f_n(m,k) \leq (n-1)mk$ and proof that $f_n(m+1,k) \leq (n-1)(m+1)k$. Consider a tree $\mathcal{T} \in \mathcal{T}_{n,m+1}$, and lets see it as a root, n edges from the root, $e_1, e_2, \ldots, e_n$, and n trees in $\mathcal{T}_{n,m}$, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, with their own root on the other side of $e_1, e_2, \ldots, e_n$, respectively. Let $\alpha, \alpha' \in \rho(\mathcal{T})$ with $d(\alpha, \alpha') \leq k$ and let $\alpha_1, \alpha_2, \ldots, \alpha_n$ and $\alpha'_1, \alpha'_2, \ldots, \alpha'_n$ be the restrictions of α and α' to $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, respectively.

As we have that
$P(\mathcal{T}, \alpha) = \alpha(e_1)P(\mathcal{T}_1, \alpha_1) + \alpha(e_2)P(\mathcal{T}_2, \alpha_2) + \cdots + \alpha(e_n)P(\mathcal{T}_n, \alpha_n)$,
$P(\mathcal{T}, \alpha') = \alpha'(e_1)P(\mathcal{T}_1, \alpha'_1) + \alpha'(e_2)P(\mathcal{T}_2, \alpha'_2) + \cdots + \alpha'(e_n)P(\mathcal{T}_n, \alpha'_n)$,
$\alpha(e_1) + \alpha(e_2) + \cdots + \alpha(e_n) = 1$ and
$\alpha'(e_1) + \alpha'(e_2) + \cdots + \alpha'(e_n) = 1$.

We conclude that $|P(\mathcal{T}, \alpha) - P(\mathcal{T}, \alpha')| = |(\alpha(e_1) - \alpha'(e_1))(P(\mathcal{T}_1, \alpha_1) - P(\mathcal{T}_n, \alpha_n)) + (\alpha(e_2) - \alpha'(e_2))(P(\mathcal{T}_2, \alpha_2) - P(\mathcal{T}_n, \alpha_n)) + \cdots + (\alpha(e_{n-1}) - \alpha'(e_{n-1}))(P(\mathcal{T}_{n-1}, \alpha_{n-1}) - P(\mathcal{T}_n, \alpha_n)) + \alpha'(e_1)(P(\mathcal{T}_1, \alpha_1) - P(\mathcal{T}_1, \alpha'_1)) + \alpha'(e_2)(P(\mathcal{T}_2, \alpha_2) - P(\mathcal{T}_2, \alpha'_2)) + \cdots + \alpha'(e_n)(P(\mathcal{T}_n, \alpha_n) - P(\mathcal{T}_n, \alpha'_n))|$.

[6] This tree does not depend on the probabilities of the answers from the oracle.
[7] The wrong computations of $\mathcal{M}$ are exactly the same as the ones of $\mathcal{M}'$.

And now, as the difference of two real numbers in $[0,1]$ is less or equal to 1 and we are assuming that $f_n(m,k) \leq (n-1)mk$ is true for any tree in $\mathcal{T}_{n,m}$, we obtain $|P(\mathcal{T},\alpha) - P(\mathcal{T},\alpha')| \leq |\alpha(e_1) - \alpha'(e_1)| + |\alpha(e_2) - \alpha'(e_2)| + \cdots + |\alpha(e_{n-1}) - \alpha'(e_{n-1})| + \alpha'(e_1)f_n(m,k) + \alpha'(e_2)f_n(m,k) + \cdots + \alpha'(e_n)f_n(m,k) \leq (n-1)k + f_n(m,k) \leq (n-1)(m+1)k$.

We conclude then that $f_n(m,k) \leq (n-1)mk$ holds for any $m \in \mathbb{N}$. □

Remark 3.4.5. *With this result, Proposition 3.4.3 would just need the difference of the probabilities bounded by $2^{-p(n)-3}$ instead of $2^{-p(n)-4}$, which would reduce the size of the advices in the upper bounds of the ShSM by one digit (Propositions 3.6.6 and 3.6.8).*

The following Proposition is a strongest version of Proposition 3.4.3, in the sense that works for both sharp and smooth scatter machines. They are considered with oracles with three possible answers and so, as machines with ternary trees for they computation tree.

Proposition 3.4.5. *Let $\mathcal{M}$ be an error-prone scatter machine that decides the set $A \subseteq \Sigma^*$ in polynomial space p, with error probability $\epsilon \leq 1/4$. If $\mathcal{M}'$ is a scatter machine that, for an input w with $|w| = n$, behaves exactly like $\mathcal{M}$ except when the oracle calls. Where for any call the probability of the answer differs at most $2^{-p(n)-4}$. Then, $\mathcal{M}'$ decides A with error probability $\epsilon' \leq 3/8$.*

Proof. Given a word $w \in \Sigma^*$, consider the computation tree $\mathcal{T}_w$ of $\mathcal{M}$ for the input w, and let $\alpha, \alpha' \in \rho(\mathcal{T}_w)$ be the assignments that give to the edges of $\mathcal{T}_w$ the probabilities given by the experiments of $\mathcal{M}$ and $\mathcal{M}'$, respectively. Consider the subtree of $\mathcal{T}_w$ consisting on all the total paths from $\mathcal{T}_w$ that correspond to a wrong computation of $\mathcal{M}$ when deciding A, and call it $\mathcal{T}_w^\times$.

As $\mathcal{M}$ is space bounded by the polynomial p, for an input w, the depth of $\mathcal{T}_w$ is limited by $2^{p(|w|)}$ (see Proposition 3.4.1), and as we considered $\mathcal{T}_w$ as a ternary tree. Using Proposition 3.4.4, since then $|P(\mathcal{T}_w^\times,\alpha) - P(\mathcal{T}_w^\times,\alpha')| \leq f_3(m,k) \leq (3-1)mk$, where m and k are the depth and biggest difference for any probability of $\mathcal{T}_w^\times$, respectively, we have

$$\begin{aligned} P(\mathcal{T}_w^\times,\alpha') &\leq P(\mathcal{T}_w^\times,\alpha) + |P(\mathcal{T}_w^\times,\alpha) - P(\mathcal{T}_w^\times,\alpha')| \\ &\leq \epsilon + (3-1)\cdot 2^{p(n)}\cdot k \\ &\leq 2^{-2} + 2^{p(n)+1}k. \end{aligned}$$

So, if we have k such that $2^{p(n)+1}k \leq 2^{-3}$, we get $P(\mathcal{T}_w^\times,\alpha') \leq 3/8$. Therefore, for any word $w \in \Sigma^*$, $\mathcal{M}'$ make the same decision as $\mathcal{M}$ with error probability $\epsilon' \leq 3/8$ instead of $1/4$. □

3.5 Boundary numbers

We introduce now an useful concept for the upper bounds when using the smooth scatter experiment, the boundary numbers, already used in [8, 1]. The boundary numbers are the numbers $y' - \eta$ and $y' + \eta$ (where y' is the vertex position) from Proposition 3.2.3, for (the query size) $1 \in \mathbb{N}$, which is the size of the query for specify the cannon position $z = 1/2$.

Definition 3.5.1. *Given a SmSM $\mathcal{M}$ with vertex position y and time schedule T, for any $n \in \mathbb{N}$, we say that a real number $x \in]0,1[$ is an n-timeout of $\mathcal{M}$, if when we set the cannon position at x, the time schedule $T(n)$ halts before the experiment gets to an answer.*

Remark 3.5.2. *If x with $x < x'$ (respectively $x' < x$) is an n-timeout of $\mathcal{M}$ and x' is not, for some $n \in \mathbb{N}$, then any x'' with $x' < x''$ (respectively $x'' < x'$) is not an n-timeout of $\mathcal{M}$. Of course, for any $n \in \mathbb{N}$, y is an n-timeout of $\mathcal{M}$, since y is the vertex position.*

Definition 3.5.3. *Given a SmSM $\mathcal{M}$ with vertex position y and time schedule T, for any $n \in \mathbb{N}$, we define the boundary numbers of $\mathcal{M}$, l_n and r_n, as the two real numbers in $[0,1]$ such that l_n and r_n are the infimum and supremum of the n-timeouts of $\mathcal{M}$, respectively.*

Remark 3.5.4. *We know by construction that $l_n < y < r_n$ and that if $l_n > 0$ and $r_n < 1$, then the physical time for a shot performed at a boundary number, l_n or r_n, is exactly $T(n)$ and that $|y - l_n| = |y - r_n|$, since the wedge is symmetrical for the vertex position.*

Given a dyadic rational $q \in]0,1[$ and a real number $b \in]0,1[$ (in their binary form) a Turing machine can decide if either $q \leq b$ or $q \geq b$ but it may not be able to decide between $q \leq b$ or $q < b$ (and $q \geq b$ or $q > b$).

As q is a dyadic number its binary form has a finite length, say n. To compare q with b we must compare them digit by digit (starting from the most significant digit) among the first n digits of their binary forms. If the first difference is a 1 in q (where b has a 0) we conclude $q \geq b$, otherwise we conclude $q \leq b$ (meaning that either there are no differences or the first difference is a 0 in q).

The comparison we just described between q and b can be made with at most n units of time and no space is needed.

With this comparison we are not able to know if $q = b$ because we do not have enough information. If we know if[8] $b = b\lfloor_n$, $b = b\lfloor_n + 2^{-n}$ or neither of them, we can decide between $q < b$, $q > b$ or $q = b$.

Remark 3.5.5. *If $b = b\lfloor_n$ we have that all the digits after the n^{th} digit in the binary form of b are 0. If $b = b\lfloor_n + 2^{-n}$ we have that all the digits after the n^{th} digit in the binary form of b are 1. With their respective negations we have that there is at least one 1 or at least one 0 after the n^{th} digit in the binary form of b.*

The comparison can then be made digit by digit as well (starting from the most significant digit and among the first n digits). If the first difference is a 0 in q we conclude $q < b$. If the first difference is a 1 in q, then if after that difference there is a 1 in q, a 0 in b or $b \neq b\lfloor_n + 2^{-n}$ we conclude $q > b$, otherwise we conclude $q = b$. If $q = b\lfloor_n$ (there are no differences among the first n digits), then if $b \neq b\lfloor_n$ we conclude $q < b$, otherwise we conclude $q = b$.

This comparison between q and b can be made with at most $n + 1$ units of time and no space is needed.

[8] We denote by $x\lfloor_n$, when $x \in]0,1[$, the first n digits of the binary form of x, the first n digits after the dot.

3.6 Sharp scatter machine

This section aims to obtain the computational power of the analogue-digital sharp scatter machine with the standard communication protocol.

Lower bounds

Proposition 3.6.1. *If a set $A \in PSPACE/poly$, then A is decidable by an error-free ShSM in polynomial space.*

Proof. Since $PSPACE/poly = PSPACE/poly\star$ (see Proposition 1.2.5), we conclude that $A \in PSPACE/poly\star$. If $A \in PSPACE/poly\star$, there exists $B \in PSPACE$ and a prefix advice function $f \in poly$ such that, for every $n \in \mathbb{N}$ and every $w \in \Sigma^*$ with $|w| \leq n$, $w \in A$ if and only if $\langle w, f(n)\rangle \in B$. Let $\mathcal{M}_B$ be an advice Turing machine which decides B in polynomial space.

Consider the error-free ShSM $\mathcal{M}$ with vertex position $y = 0.x(f)$ (see Section 3.1), which for an input w with $|w| = n$, consults the oracle to obtain $f(n)$, i.e. the first $|f(n)|$ triples of $x(f)$, as described in Section 3.1. Afterwards, $\mathcal{M}$ simulates $\mathcal{M}_B$ over $\langle w, f(n)\rangle$. If $\mathcal{M}_B$ accepts $\langle w, f(n)\rangle$, then $\mathcal{M}$ accepts w, otherwise $\mathcal{M}$ rejects w. Since $\mathcal{M}$ accepts w if and only if $\mathcal{M}_B$ accepts $\langle w, f(n)\rangle$, we conclude that the scatter machine $\mathcal{M}$ decides A.

Since $f \in poly$, $n = |w|$ and $|\langle w, f(n)\rangle| \in \mathcal{O}(|w| + |f(n)|)$ we have $|x(f)|$ and $|\langle w, f(n)\rangle$ polynomial on n. Thus, to obtain $f(n)$ with the oracle or to write $x(f)$ or $\langle w, f(n)\rangle$ in a tape $\mathcal{M}$ needs polynomial space on n. To simulate $\mathcal{M}_B$ over $\langle w, f(n)\rangle$ $\mathcal{M}$ needs polynomial space on $|\langle w, f(n)\rangle|$, since $\mathcal{M}_B$ runs in polynomial space. So, we conclude that $\mathcal{M}$ operates in polynomial space. □

The computation time of $\mathcal{M}$ on w ($|w| = n$) is the time that $\mathcal{M}_B$ needs for the input $\langle w, f(n)\rangle$ plus the time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$. Except for the simulation of $\mathcal{M}_B$, the time is polynomial on n. The time for $\mathcal{M}_B$ can be at most exponential on n.

With the error-prone protocols, as we have a probabilistic analogue component (the experiment), we can simulate a probabilistic Turing machine on the scatter machine. With that purpose we need to use the probabilistic nature of the experiment to simulate the probabilistic nature of the Turing machine (see Section 2.4). We are then going to use the experiment with probabilistic calls, which implies to have the vertex position inside the accuracy interval, and the particle may hits the vertex (event with probability zero). We must then use the time schedule from the scatter machine, a constant time schedule, set so the constant physical time the experiment takes, since if a particle hits the vertex we do not know what may happens and we need to stop the experiment and keep going with the digital computations.

The same happens in the fixed precision case, below, when we are using the experiment to obtain the vertex position. With the arbitrary precision we only need the time schedule to simulate the probabilistic Turing machine, but with the fixed precision we need the time schedule to both simulate the probabilistic Turing machine and obtain the vertex position as advice.

Proposition 3.6.2. *If a set $A \in BPPSPACE//poly$, then A is decidable by an error-prone arbitrary precision ShSM in polynomial space.*

Proof. Since $BPPSPACE//poly = BPPSPACE//poly\star$ (see Proposition 1.2.10) we conclude that $A \in BPPSPACE//poly\star$. If $A \in BPPSPACE//poly\star$, there exists a probabilistic advice Turing machine $\mathcal{M}$ bounded in polynomial space, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in poly$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, if $w \in A$, $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most ϵ and if $w \notin A$, $\mathcal{M}$ accepts $\langle w, f(n)\rangle$ with probability at most ϵ.

Consider the probabilistic error-prone arbitrary precision ShSM $\mathcal{M}'$ with vertex position $y = 0.x(f)$, which for an input w with $|w| = n$, consults the oracle to obtain $f(n)$ as described in Section 3.1. Afterwards, $\mathcal{M}'$ simulates $\mathcal{M}$ over $\langle w, f(n)\rangle$ and accepts w if and only if $\mathcal{M}$ accepts $\langle w, f(n)\rangle$. As $\mathcal{M}'$ can simulate $\mathcal{M}$ using its probabilistic oracle, it do not need to have a probabilistic digital component (see Section 2.4). We conclude then that $\mathcal{M}'$ is making a mistake to decide A with probability at most ϵ, so it is deciding A.

Since $f \in poly$, $n = |w|$ and $|\langle w, f(n)\rangle| \in \mathcal{O}(|w|+|f(n)|)$ we have $|x(f)|$ and $|\langle w, f(n)\rangle$ polynomial on n. Thus, to obtain $f(n)$ with the oracle or to write $x(f)$ or $\langle w, f(n)\rangle$ in a tape $\mathcal{M}'$ needs polynomial space on n. To simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ $\mathcal{M}'$ needs polynomial space on $|\langle w, f(n)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the simulation of the probabilistic component can be done in polynomial space. So, we conclude that $\mathcal{M}'$ operates in polynomial space. □

The computation time of $\mathcal{M}'$ on w ($|w| = n$) is the time that $\mathcal{M}$ needs for the input $\langle w, f(n)\rangle$ simulating the probabilistic component with the oracle plus the time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$. The time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$ is polynomial on n and the time needed to simulate the probabilistic component with the oracle and simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ can be at most exponential on n. The time needed to simulate the probabilistic component with the oracle is exponential on n if and only if we need an exponential amount of fair coin tosses to simulate $\mathcal{M}$.

Proposition 3.6.3. *If a set $A \in BPPSPACE//poly$, then A is decidable by an error-prone finite precision ShSM in polynomial space.*

Proof. Since $BPPSPACE//poly = BPPSPACE//poly\star$ (see Proposition 1.2.10) we conclude that $A \in BPPSPACE//poly\star$. If $A \in BPPSPACE//poly\star$, there exists a probabilistic advice Turing machine $\mathcal{M}$ bounded in polynomial space, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in poly$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, if $w \in A$, $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most ϵ and if $w \notin A$, $\mathcal{M}$ accepts $\langle w, f(n)\rangle$ with probability at most ϵ.

Consider the probabilistic error-prone finite precision ShSM $\mathcal{M}'$ with fixed error ξ and vertex position $y' = 1/2 + \xi - 2\xi y$, where $y = 0.x(f)$, as described below and a constant δ with $0 < \delta < 1/2 - \epsilon$. We want the vertex position at y' so that a call to the oracle with the cannon position set at $z = 1/2$ answers right with probability y and we want $\delta < 1/2 - \epsilon$ so that $\epsilon + \delta < 1/2$.

By Proposition 3.2.2 $\mathcal{M}'$, on an input w with $|w| = n$, can obtain $f(n)$ with error probability smaller than δ. We define $\mathcal{M}'$ to, after getting $f(n)$, simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ and accepts w if and only if $\mathcal{M}$ accepts $\langle w, f(n)\rangle$. As $\mathcal{M}'$ can simulate $\mathcal{M}$ using its probabilistic oracle, it do not need to have a probabilistic digital component (see Section 2.4). We conclude then that $\mathcal{M}'$ fails to obtain $f(n)$ or simulate a wrong computation of $\mathcal{M}$ with probability at most $\delta + \epsilon < 1/2$. Therefore, $\mathcal{M}'$ decides A.

Since $f \in poly$, $n = |w|$ and $|\langle w, f(n)\rangle| \in \mathcal{O}(|w|+|f(n)|)$ we have $|x(f)|$ and $|\langle w, f(n)\rangle$ polynomial on n. Thus, to obtain $f(n)$ with the oracle or to write $x(f)$ or $\langle w, f(n)\rangle$ in a tape $\mathcal{M}'$ needs polynomial space on n. To simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ $\mathcal{M}'$ needs polynomial space on $|\langle w, f(n)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the simulation of the probabilistic component can be done in polynomial space. So, we can conclude that $\mathcal{M}'$ operates in polynomial space. □

The computation time of $\mathcal{M}'$ on w ($|w| = n$) is the time that $\mathcal{M}$ needs for the input $\langle w, f(n)\rangle$ simulating the probabilistic component with the oracle plus the time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$. The time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$ is polynomial on n and the time needed to simulate the probabilistic component with the oracle and simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ can be at most exponential on n. The time needed to simulate the probabilistic

component with the oracle is exponential on n if and only if we need an exponential amount of fair coin tosses to simulate $\mathcal{M}$.

Upper bounds

This proof could be made as the other upper bound proofs through the dissertation, simulating directly the scatter machine in the oracle Turing machine. We should then use a comparison method as the one described in Section 3.5 as in Proposition 3.7.4, and we should be careful with the comparison to be able to distinguish between strict and non-strict inequalities. With the technique we use in this proof, already used in [6], we use the signs '$>$' and '$=$' to make this distinction.

Proposition 3.6.4. *If a set $A \subseteq \Sigma^*$ is decidable by an error-free ShSM in polynomial space, then $A \in PSPACE/poly$.*

Proof. Let $\mathcal{M}$ be an error-free ShSM which decides A with vertex position y bounded in polynomial space p. Let O_y^p be the sparse set as described in Section 3.3, we must have an increasing function p (the polynomial) to use the oracle O_y^p, but we can consider the function p' such that $p'(0) = f(0)$ and $p'(n) = \max\{f(n), p'(n-1)\}$ for any $n \in \mathbb{N}$ with $n \geq 1$. Since p' is also a polynomial function that bound the space used by $\mathcal{M}$.

Consider the oracle Turing machine $\mathcal{M}'$ that, for an input w with $|w| = n$, consults the usual oracle O_y^p to obtain its word of length $p(n) + n + 1$ as described in Section 3.3, let the word be called o (O_y^p is a tally set). Afterwards, $\mathcal{M}'$ simulates $\mathcal{M}$ on w using o to answer the queries. As for any input w $\mathcal{M}$ can only write in the query tape a query q with $|q| \leq p(|w|)$, any possible query of $\mathcal{M}$ can be answered comparing it to o, since o ends with either an '$=$' or a '$>$' sign and no query q can have more numeric digits than o. We conclude then that $\mathcal{M}'$ decides A.

To simulate $\mathcal{M}$ over w answering the oracle calls with the word o $\mathcal{M}'$ needs polynomial space on $|w|$ since $\mathcal{M}$ is bounded in polynomial space and no space is needed to the comparison of a query q with o. To get o out of O_y^p $\mathcal{M}'$ needs polynomial space on $|w|$ since $|o| = p(|w|) + |w| + 1$. So, we can conclude that $\mathcal{M}'$ operates in polynomial space and using the Proposition 3.3.1 we get the desired result $A \in PSPACE/poly$. □

The computation time of $\mathcal{M}'$ on w is the time $\mathcal{M}$ takes on the same input plus the time needed to get the word o and make the comparisons between the queries and o. The time needed to get the word o is polynomial on $|w|$ and the time needed to simulate $\mathcal{M}$ and make the comparisons can be at most exponential on $|w|$. The time needed to make the comparisons is exponential on $|w|$ if and only if $\mathcal{M}$ makes an exponential amount of oracle calls.

Since we have both the lower and upper bounds for the same analogue-digital machine, we can take the bounds together to state the following theorem.

Theorem 3.6.5. *A set $A \subseteq \Sigma^*$ is decidable by an error-free ShSM in polynomial space if and only if $A \in PSPACE/poly$.*

Proof. It follows from Propositions 3.6.1 and 3.6.4. □

Proposition 3.6.6. *If a set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision ShSM in polynomial space, then $A \in BPPSPACE//poly$.*

Proof. Let $\mathcal{M}$ be an error-prone arbitrary precision ShSM which decides A with vertex position y bounded in polynomial space p. We assume, without loss of generality, that $\mathcal{M}$ has error probability $\epsilon \leq 1/4$.

Let f be the advice function such that $f(n) = y\downarrow_{2p(n)+3}$, which is clearly in $poly$. We have that $|y - 0.y\downarrow_{2p(n)+3}| \leq 2^{-2p(n)-3}$ and that $0.y\downarrow_{2p(n)+3}$ is a dyadic rational.

Consider the probabilistic advice Turing machine $\mathcal{M}'$ which, for an input $\langle w, f(|w|)\rangle$ with $|w| = n$, behaves exactly like $\mathcal{M}$ for the input w except when the oracle calls. Where for a query q uses the advice $f(n)$, since it is a finite word, to simulate digitally the probabilistic answer from the oracle (see Section 2.4). Thus, $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ if its simulation of $\mathcal{M}$ accepts w.

As $\mathcal{M}$ is bounded in polynomial space by p, for an input w, it can only write in the query tape a word with $|q| \leq p(|w|)$. So, according to the protocol, the uncertainty to the cannon position is greater or equal to $2 \cdot 2^{-p(|w|)}$. As also $|y - 0.y\downarrow_{2p(n)+3}| \leq 2^{-2p(n)-3}$ we conclude that, for any call to the oracle, the difference of the probabilities for the answer between $\mathcal{M}$ and $\mathcal{M}'$ is at most $2^{-p(n)-4}$. Thus, $\mathcal{M}'$ is simulating a ShSM as $\mathcal{M}$, but with mandatory dyadic vertex position and close enough probability values. Therefore, calling $\mathcal{M}''$ to that ShSM, by Proposition 3.4.3, we conclude that $\mathcal{M}''$ decides A with error probability $\epsilon'' \leq 3/8$ and so, $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ with error probability $\epsilon' \leq 3/8$. We conclude then that $\mathcal{M}'$ decides A.

To simulate $\mathcal{M}$ over w with the oracle calls simulated digitally $\mathcal{M}'$ needs polynomial space on $|\langle w, f(|w|)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the oracle is simulated in polynomial space on $|w|$. We conclude then that $\mathcal{M}'$ operates in polynomial space and thus $A \in BPPSPACE//poly$. □

The computation time $\mathcal{M}'$ takes on $\langle w, f(|w|)\rangle$ is the time $\mathcal{M}$ takes on w plus the time needed to simulate the experiments. The time needed to simulate $\mathcal{M}$ over w and simulate the experiments can be at most exponential on $|w|$. The time needed to simulate the experiments is exponential on $|w|$ if and only if $\mathcal{M}$ makes an exponential amount of oracle calls.

Theorem 3.6.7. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision ShSM in polynomial space if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Propositions 3.6.2 and 3.6.6. □

Proposition 3.6.8. *If a set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision ShSM in polynomial space, then $A \in BPPSPACE//poly$.*

Proof. Let $\mathcal{M}$ be an error-prone finite precision ShSM which decides A with vertex position y bounded in polynomial space p. We assume, without loss of generality, that $\mathcal{M}$ has error probability $\epsilon \leq 1/4$ and fixed error $\xi = 2^{-N}$ for some $N \in \mathbb{N}$.

Let f be the advice function such that $f(n) = y\downarrow_{p(n)+3+N}$, which is clearly in $poly$. We have that $|y - 0.y\downarrow_{p(n)+3+N}| \leq 2^{-p(n)-3-N}$ and that $0.y\downarrow_{p(n)+3+N}$ is a dyadic rational.

Consider the probabilistic advice Turing machine $\mathcal{M}'$ which, for an input $\langle w, f(|w|)\rangle$ with $|w| = n$, behaves exactly like $\mathcal{M}$ for the input w except when the oracle calls. Where for a query q uses the advice $f(n)$ to simulate digitally the probabilistic answers from the oracle (see Section 2.4). Thus, $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ if its simulation of $\mathcal{M}$ accepts w.

According to the protocol, the uncertainty to the cannon position is $2{\cdot}2^{-N}$. As also $|y{-}0.y\downarrow_{p(n)+3+N}| \leq 2^{-p(n)-3-N}$ we conclude that, for any call to the oracle, the difference of the probabilities for the answer between $\mathcal{M}$ and $\mathcal{M}'$ is at most $2^{-p(n)-4}$. Thus, $\mathcal{M}'$ is simulating a ShSM as $\mathcal{M}$, but with mandatory dyadic vertex position and close enough probability values. Therefore, calling $\mathcal{M}''$ to that ShSM, by Proposition 3.4.3, we conclude that $\mathcal{M}''$ decides A with error probability $\epsilon'' \leq 3/8$ and so, $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ with error probability $\epsilon' \leq 3/8$. We conclude then that $\mathcal{M}'$ decides A.

To simulate $\mathcal{M}$ over w with the oracle calls simulated digitally $\mathcal{M}'$ needs polynomial space on $|\langle w, f(|w|)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the oracle is simulated in polynomial space on $|w|$. We conclude then that $\mathcal{M}'$ operates in polynomial space and thus $A \in BPPSPACE//poly$. □

The computation time $\mathcal{M}'$ takes on $\langle w, f(|w|)\rangle$ is the time $\mathcal{M}$ takes on w plus the time needed to simulate the experiments. The time needed to simulate $\mathcal{M}$ over w and simulate the experiments can be at most exponential on $|w|$. The time needed to simulate the experiments is exponential on $|w|$ if and only if $\mathcal{M}$ makes an exponential amount of oracle calls.

Theorem 3.6.9. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision ShSM in polynomial space if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Propositions 3.6.3 and 3.6.8. □

We can then state all the results concerning the standard analogue-digital sharp scatter machine bounded in polynomial space.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$
Upper Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$

Table 3.3: Standard communication protocol ShSM

3.7 Smooth scatter machine

Lower bounds

To the lower bounds of the smooth scatter machine we can either choose to use the time schedule or not, but either way we have to guarantee that the experiment do not keeps running indefinitely. Accordingly to the search method described in Section 3.1, a SmSM do not shoot at the vertex position. Even better, it shoots at a far enough distance from the vertex position so that the physical time of each shot is at most exponential on the query size. Therefore, on the first two precision assumptions the lower bound proofs do not need to use the time schedule to bound the physical time of the experiment when using the linear search method, since the time is already guaranteed as finite (actually exponential).

Proposition 3.7.1. *If a set $A \in PSPACE/poly$, then A is decidable by an error-free SmSM in polynomial space.*

Proof. Since $PSPACE/poly = PSPACE/poly\star$ (see Proposition 1.2.5), we conclude that $A \in PSPACE/poly\star$. If $A \in PSPACE/poly\star$, there exists $B \in PSPACE$ and a prefix advice function $f \in poly$ such that, for every $n \in \mathbb{N}$ and for every $w \in \Sigma^*$ with $|w| \leq n$, $w \in A$ if and only if $\langle w, f(n)\rangle \in B$. Let $\mathcal{M}_B$ be an advice Turing machine which decides B in polynomial space.

Consider the error-free SmSM $\mathcal{M}$ with vertex position $y = 0.x(f)$, which for an input w with size $|w| = n$, consults the oracle to obtain $f(n)$ as described in Section 3.1. With the search method described any call to the oracle has a physical time at most exponential on the query size. Afterwards, $\mathcal{M}$ simulates $\mathcal{M}_B$ over $\langle w, f(n)\rangle$. If $\mathcal{M}_B$ accepts $\langle w, f(n)\rangle$, then $\mathcal{M}$ accepts w, otherwise $\mathcal{M}$ rejects w. Since $\mathcal{M}$ accepts w if and only if $\mathcal{M}_B$ accepts $\langle w, f(n)\rangle$, we conclude that the scatter machine $\mathcal{M}$ decides A.

Since $f \in poly$, $n = |w|$ and $|\langle w, f(n)\rangle| \in \mathcal{O}(|w| + |f(n)|)$ we have $|x(f)|$ and $|\langle w, f(n)\rangle$ polynomial on n. Thus, to obtain $f(n)$ with the oracle or to write $x(f)$ or $\langle w, f(n)\rangle$ in a tape $\mathcal{M}$ needs polynomial

space on n. To simulate $\mathcal{M}_B$ over $\langle w, f(n)\rangle$ $\mathcal{M}$ needs polynomial space on $|\langle w, f(n)\rangle|$, since $\mathcal{M}_B$ runs in polynomial space. So, we conclude that $\mathcal{M}$ operates in polynomial space. □

The computation time of $\mathcal{M}$ on w is the time that $\mathcal{M}_B$ needs for the input $\langle w, f(n)\rangle$ plus the time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$. The time needed to perform the experiments and simulate $\mathcal{M}_B$ over $\langle w, f(n)\rangle$ can be at most exponential on n.

If we want to use the time schedule from $\mathcal{M}$ we should use a time schedule T such that $T(n) \geq 2^{n+2}$ in order to be able to apply the linear search method. We can have a clock for the function $2(n+2) \cdot 2^{n+2-(\lfloor \log n+2 \rfloor+1)} \geq 2^{n+2}$ in polynomial space (see Section 2.1).

Proposition 3.7.2. *If a set $A \in BPPSPACE//poly$, then A is decidable by an error-prone arbitrary precision SmSM in polynomial space.*

Proof. Since $BPPSPACE//poly = BPPSPACE//poly\star$ (see Proposition 1.2.10) we conclude that $A \in BPPSPACE//poly\star$. If $A \in BPPSPACE//poly\star$, there exists a probabilistic advice Turing machine $\mathcal{M}$ bounded in polynomial space, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in poly$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, if $w \in A$, $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most ϵ and if $w \notin A$, $\mathcal{M}$ accepts $\langle w, f(n)\rangle$ with probability at most ϵ.

Consider the probabilistic error-prone arbitrary precision SmSM $\mathcal{M}'$ with vertex position $y = 0.x(f)$, which for an input w with size $|w| = n$, consults the oracle to obtain $f(n)$ as described in Section 3.1. With the search method described any call to the oracle have a physical time at most exponential on the query size. Afterwards, $\mathcal{M}'$ simulates $\mathcal{M}$ over $\langle w, f(n)\rangle$ and accepts w if and only if $\mathcal{M}$ accepts $\langle w, f(n)\rangle$. As $\mathcal{M}'$ can simulate $\mathcal{M}$ using its probabilistic oracle, it do not need to have a probabilistic digital component (see Section 2.4). We conclude then that $\mathcal{M}'$ is making a mistake to decide A with probability at most ϵ, so it is deciding A.

Since $f \in poly$, $n = |w|$ and $|\langle w, f(n)\rangle| \in \mathcal{O}(|w|+|f(n)|)$ we have $|x(f)|$ and $|\langle w, f(n)\rangle$ polynomial on n. Thus, to obtain $f(n)$ with the oracle or to write $x(f)$ or $\langle w, f(n)\rangle$ in a tape $\mathcal{M}'$ needs polynomial space on n. To simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ $\mathcal{M}'$ needs polynomial space on $|\langle w, f(n)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the simulation of the probabilistic component can be done in polynomial space. So, we conclude that $\mathcal{M}'$ operates in polynomial space. □

The computation time of $\mathcal{M}'$ on w ($|w| = n$) is the time that $\mathcal{M}$ needs for the input $\langle w, f(n)\rangle$ simulating the probabilistic component with the oracle plus the time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$. The time needed to perform the experiments, simulate the probabilistic component with the oracle and simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ can be at most exponential on n.

If we want to use the time schedule from $\mathcal{M}$ we should use a time schedule T such that $T(n) \geq 2^{n+3}$ in order to be able to apply the linear search method. We can have a clock for the function $2(n+3) \cdot 2^{n+3-(\lfloor \log n+3 \rfloor+1)} \geq 2^{n+3}$ in polynomial space (see Section 2.1). To simulate the probabilistic nature of the advice Turing machine we can use any time schedule as long as it provides a probabilistic experiment (see Section 2.4).

With the smooth scatter experiment we have to be careful so that the experiment do not keeps running indefinitely. With the fixed precision assumption we must use the time schedule, or we are not able go get any relevant information from the scatter experiment. As explained in Section 3.2 (see Proposition 3.2.3) we can use any time schedule as long as it satisfies the conditions for the query we use, in order to obtain the vertex position we use always the same query so we can use a constant time schedule. To simulate the probabilistic nature of the advice Turing machine we can use any time schedule as long as it provides a probabilistic experiment (see Section 2.4).

Proposition 3.7.3. *If a set $A \in BPPSPACE//poly$, then A is decidable by an error-prone finite precision SmSM in polynomial space, which makes use of the time schedule.*

Proof. Since $BPPSPACE//poly = BPPSPACE//poly\star$ (see Proposition 1.2.10) we conclude that $A \in BPPSPACE//poly\star$. If $A \in BPPSPACE//poly\star$, there exists a probabilistic advice Turing machine $\mathcal{M}$ bounded in polynomial space, a constant ϵ with $0 < \epsilon < 1/2$ and a prefix advice function $f \in poly$ such that, for every $n \in \mathbb{N}$ and $w \in \Sigma^*$ with $|w| \leq n$, if $w \in A$, $\mathcal{M}$ rejects $\langle w, f(n)\rangle$ with probability at most ϵ and if $w \notin A$, $\mathcal{M}$ accepts $\langle w, f(n)\rangle$ with probability at most ϵ.

Consider the probabilistic error-prone finite precision SmSM $\mathcal{M}'$ with fixed error ξ and vertex position $y' = 1/2 + \xi - 2\xi y$, where $y = 0.x(f)$, as described below and a constant δ with $0 < \delta < 1/2 - \epsilon$. We want the vertex position at y' so that a call to the oracle with the cannon position set at $z = 1/2$ answers right with probability y and we want $\delta < 1/2 - \epsilon$ so that $\epsilon + \delta < 1/2$.

By Proposition 3.2.3 $\mathcal{M}'$, on an input w with $|w| = n$, can obtain $f(n)$ with error probability smaller than δ. We need to use the time schedule to obtain the advice, a constant time schedule is enough to the purpose but any one that satisfies the condition on section 3.2 (see Proposition 3.2.3) in polynomial space would suffice. We define $\mathcal{M}'$ to, after getting $f(n)$, simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ and accepts w if and only if $\mathcal{M}$ accepts $\langle w, f(n)\rangle$. As $\mathcal{M}'$ can simulate $\mathcal{M}$ using its probabilistic oracle, it do not need to have a probabilistic digital component (see Section 2.4). We conclude then that $\mathcal{M}'$ fails to obtain the correct $f(n)$ or simulate a wrong computation of $\mathcal{M}$ with probability at most $\delta + \epsilon < 1/2$. Therefore, $\mathcal{M}'$ decides A.

Since $f \in poly$, $n = |w|$ and $|\langle w, f(n)\rangle| \in \mathcal{O}(|w|+|f(n)|)$ we have $|x(f)|$ and $|\langle w, f(n)\rangle$ polynomial on n. Thus, to obtain $f(n)$ with the oracle or to write $x(f)$ or $\langle w, f(n)\rangle$ in a tape $\mathcal{M}'$ needs polynomial space on n. To simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ $\mathcal{M}'$ needs polynomial space on $|\langle w, f(n)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the simulation of the probabilistic component can be done in polynomial space. So, we can conclude that $\mathcal{M}'$ operates in polynomial space. □

The computation time of $\mathcal{M}'$ on w ($|w| = n$) is the time that $\mathcal{M}$ needs for the input $\langle w, f(n)\rangle$ simulating the probabilistic component with the oracle plus the time needed to perform the experiments and write $x(f)$ and $\langle w, f(n)\rangle$. The time needed to perform the experiments, simulate the probabilistic component with the oracle and simulate $\mathcal{M}$ over $\langle w, f(n)\rangle$ can be at most exponential on n.

Upper bounds

To the upper bounds of the smooth scatter machine we have to simulate general scatter machines that can have any vertex position and time schedule. We know that the maximum growing rate function we can use as the time schedule for an analogue-digital machine bounded in polynomial space is the exponential one (see Proposition 2.1.2). We must then be able to simulate a scatter machine with any time schedule of at most an exponential growing rate. As we can only access dyadic probability values with a general probabilistic Turing machine, to simulate the experiment we generate a dyadic probability event with probability sufficiently close to the probability of the real analogue oracle (see Section 2.4). The probability values of the scatter experiment depend on the time schedule, but as we code the boundary numbers on the advice function we use, we have the dyadic probability values dependent on the time schedule already since the boundary numbers depend on the time schedule.

Proposition 3.7.4. *If a set $A \subseteq \Sigma^*$ is decidable by an error-free SmSM in polynomial space, using the time schedule, then $A \in PSPACE/poly$.*

Proof. Let $\mathcal{M}$ be an error-free SmSM which decides A with vertex position y and time schedule T bounded in polynomial space p.

Consider, for any (input size) $n \in \mathbb{N}$, the boundary numbers of $\mathcal{M}$, l_n and r_n, and the digits f_n^l and f_n^r. We set $f_n^l = 0$ if $l_n = l_n\downarrow_n$, $f_n^l = 1$ if $l_n = l_n\downarrow_n + 2^{-n}$ and $f_n^l = \varepsilon$ otherwise. Analogously, we set $f_n^r = 0$ if $r_n = r_n\downarrow_n$, $f_n^r = 1$ if $r_n = r_n\downarrow_n + 2^{-n}$ and $f_n^r = \varepsilon$ otherwise (see Section 3.5). Let f be the prefix advice function such that

$$f(n) = l_1\downarrow_1 \# f_1^l \# r_1\downarrow_1 \# f_1^r \# l_2\downarrow_2 \# f_2^l \# r_2\downarrow_2 \# f_2^r \# \cdots \# l_{p(n)}\downarrow_{p(n)} \# f_{p(n)}^l \# r_{p(n)}\downarrow_{p(n)} \# f_{p(n)}^r$$

which is in $poly$, since[9][10] $6 \cdot p(n) - 1 + 2 \cdot p(n) \cdot (p(n)+1)/2$ is a polynomial on n.

Consider the advice Turing machine $\mathcal{M}'$ which, for an input $\langle w, f(|w|)\rangle$, behaves exactly like $\mathcal{M}$ for the input w except when the oracle calls. Where for a query q uses the approximations of the boundary numbers $l_{|q|}\downarrow_{|q|}$ and $r_{|q|}\downarrow_{|q|}$ and the digits $f_{|q|}^l$ and $f_{|q|}^r$ in the advice $f(|w|)$ to compare them with q and give the same answer as the oracle. For an input w, $\mathcal{M}$ can only write in the query tape a word q with $|q| \leq p(|w|)$ and so, $\mathcal{M}'$ can compare any query it may need for a computation of $\mathcal{M}$ (see Section 3.5). We conclude then that $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ if and only if $\mathcal{M}$ accepts w, thus $\mathcal{M}'$ decides A.

To simulate $\mathcal{M}$ over w with the oracle calls answered digitally $\mathcal{M}'$ needs polynomial space on $|\langle w, f(|w|)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the oracle simulation needs no further space. We conclude then that $\mathcal{M}'$ operates in polynomial space and thus $A \in BPPSPACE//poly$. □

The computation time $\mathcal{M}'$ takes on $\langle w, f(|w|)\rangle$ is the time $\mathcal{M}$ takes on w plus the time needed to simulate the experiments. The time needed to simulate $\mathcal{M}$ over w and simulate the experiments can be at most exponential on $|w|$. The time needed to simulate the experiments is exponential on $|w|$ if and only if $\mathcal{M}$ makes an exponential amount of oracle calls.

As we do not need the time schedule in the lower bounds, but we can use them, and we must to consider them to the upper bound proofs. We state the final results with exponential time schedules, but we are also able to decide with polynomial advice and polynomial space the sets decided by SmSM's with different time schedules.

Theorem 3.7.5. *A set $A \subseteq \Sigma^*$ is decidable by an error-free SmSM in polynomial space, using an exponential time schedule, if and only if $A \in PSPACE/poly$.*

Proof. It follows from Propositions 3.7.1 and 3.7.4. □

Proposition 3.7.6. *If a set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision SmSM in polynomial space, using the time schedule, then $A \in BPPSPACE//poly$.*

Proof. Let $\mathcal{M}$ be an error-prone arbitrary precision SmSM which decides A with vertex position y and time schedule T bounded in polynomial space p. We assume, without loss of generality, that $\mathcal{M}$ has error probability $\epsilon \leq 1/4$.

Consider, for any $n \in \mathbb{N}$, the boundary numbers of $\mathcal{M}$, l_n and r_n, and let f be the prefix advice function such that

$$f(n) = l_1\downarrow_{p(n)+3+1} \# r_1\downarrow_{p(n)+3+1} \# l_2\downarrow_{p(n)+3+2} \# r_2\downarrow_{p(n)+3+2} \# \cdots \# l_{p(n)}\downarrow_{p(n)+3+p(n)} \# r_{p(n)}\downarrow_{p(n)+3+p(n)}$$

which is in $poly$, since $2 \cdot p(n) - 1 + 2 \cdot p(n) \cdot (p(n)+3) + 2 \cdot p(n) \cdot (p(n)+1)/2$ is a poly on n.

Consider the probabilistic advice Turing machine $\mathcal{M}'$ which, for an input $\langle w, f(|w|)\rangle$, behaves exactly like $\mathcal{M}$ for the input w except when the oracle calls. Where for a query q uses the approximations of

[9] $1 + 2 + 3 + \cdots + n = n \cdot (n+1)/2$. Note that, $n \cdot (n+1)/2$ is an integer, since either n or $n+1$ is even.

[10] The size we estimate for the advice is counting each digit (0, 1 and $\#$) as with length 1, if we want to code the advice in Σ^* we should count each digit as with length 2 (as well as in Proposition 3.6.4 with the symbols '$>$' and '$=$').

the boundary numbers $l_{|q|}\downarrow_{p(|w|)+3+|q|}$ and $r_{|q|}\downarrow_{p(|w|)+3+|q|}$ in the advice $f(|w|)$ to simulate digitally the probabilistic answer from the oracle. For an input w, $\mathcal{M}$ can only write in the query tape a word q with $|q| \leq p(|w|)$ and so, $\mathcal{M}'$ can simulate the answer for any query it may need for a computation of $\mathcal{M}$ (see Section 2.4). We conclude then that $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ if its simulation of $\mathcal{M}$ accepts w.

According to the protocol, for a query q, the uncertainty to the cannon position is $2 \cdot 2^{-|q|}$. As $|l_{|q|} - 0.l_{|q|}\downarrow_{p(n)+3+|q|}| \leq 2^{-p(n)-3-|q|}$ and $|r_{|q|} - 0.r_{|q|}\downarrow_{p(n)+3+|q|}| \leq 2^{-p(n)-3-|q|}$, and also $0.r_{|q|}\downarrow_{p(n)+3+|q|} \leq r_{|q|}$. We conclude that, for any call to the oracle, the difference of the probabilities for the answer between $\mathcal{M}$ and $\mathcal{M}'$ is at most $2^{-p(n)-4}$. Thus, $\mathcal{M}'$ is simulating a SmSM as $\mathcal{M}$, but with mandatory dyadic boundary numbers and close enough probability values. Therefore, calling $\mathcal{M}''$ to that SmSM, by Proposition 3.4.5, we conclude that $\mathcal{M}''$ decides A with error probability $\epsilon'' \leq 3/8$ and so, $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ with error probability $\epsilon' \leq 3/8$. We conclude then that $\mathcal{M}'$ decides A.

To simulate $\mathcal{M}$ over w with the oracle calls simulated digitally $\mathcal{M}'$ needs polynomial space on $|\langle w, f(|w|)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the oracle is simulated in polynomial space on $|w|$. We conclude then that $\mathcal{M}'$ operates in polynomial space and thus $A \in BPPSPACE//poly$. □

The computation time $\mathcal{M}'$ takes on $\langle w, f(|w|)\rangle$ is the time $\mathcal{M}$ takes on w plus the time needed to simulate the experiments. The time needed to simulate $\mathcal{M}$ over w and simulate the experiments can be at most exponential on $|w|$. The time needed to simulate the experiments is exponential on $|w|$ if and only if $\mathcal{M}$ makes an exponential amount of oracle calls.

Theorem 3.7.7. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision SmSM in polynomial space, using an exponential time schedule, if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Propositions 3.7.2 and 3.7.6. □

Proposition 3.7.8. *If a set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision SmSM in polynomial space, using the time schedule, then $A \in BPPSPACE//poly$.*

Proof. Let $\mathcal{M}$ be an error-prone finite precision SmSM which decides A with vertex position y and time schedule T bounded in polynomial space p. We assume, without loss of generality, that $\mathcal{M}$ has error probability $\epsilon \leq 1/4$ and fixed error $\xi = 2^{-N}$ for some $N \in \mathbb{N}$.

Consider, for any $n \in \mathbb{N}$, the boundary numbers of $\mathcal{M}$, l_n and r_n, and let f be the prefix advice function such that

$$f(n) = l_1\downarrow_{p(n)+3+N}\#r_1\downarrow_{p(n)+3+N}\#l_2\downarrow_{p(n)+3+N}\#r_2\downarrow_{p(n)+3+N}\#\cdots\#l_{p(n)}\downarrow_{p(n)+3+N}\#r_{p(n)}\downarrow_{p(n)+3+N}$$

which is in $poly$, since $2 \cdot p(n) - 1 + 2 \cdot p(n) \cdot (p(n) + 3 + N)$ is a poly on n.

Consider the probabilistic advice Turing machine $\mathcal{M}'$ which, for an input $\langle w, f(|w|)\rangle$, behaves exactly like $\mathcal{M}$ for the input w except when the oracle calls. Where for a query q uses the approximations of the boundary numbers $l_{|q|}\downarrow_{p(|w|)+3+N}$ and $r_{|q|}\downarrow_{p(|w|)+3+N}$ in the advice $f(|w|)$ to simulate digitally the probabilistic answer from the oracle. For an input w, $\mathcal{M}$ can only write in the query tape a word q with $|q| \leq p(|w|)$ and so, $\mathcal{M}'$ can simulate the answer of any query it may need for a computation of $\mathcal{M}$ (see Section 2.4). We conclude then that $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ if its simulation of $\mathcal{M}$ accepts w.

According to the protocol, the uncertainty to the cannon position is $2{\cdot}2^{-N}$. As $|l_{|q|} - 0.l_{|q|}\downarrow_{p(|w|)+3+N}| \leq 2^{-p(|w|)-3-N}$ and $|r_{|q|} - 0.r_{|q|}\downarrow_{p(|w|)+3+N}| \leq 2^{-p(|w|)-3-N}$, and also $0.r_{|q|}\downarrow_{p(|w|)+3+N} \leq r_{|q|}$. We conclude that, for any call to the oracle, the difference of the probabilities for the answer between $\mathcal{M}$ and $\mathcal{M}'$ is at most $2^{-p(n)-4}$. Thus, $\mathcal{M}'$ is simulating a SmSM as $\mathcal{M}$, but with mandatory dyadic boundary numbers and close enough probability values. Therefore, calling $\mathcal{M}''$ to that SmSM, by Proposition 3.4.5, we conclude that $\mathcal{M}''$ decides A with error probability $\epsilon'' \leq 3/8$ and so, $\mathcal{M}'$ accepts $\langle w, f(|w|)\rangle$ with error probability $\epsilon' \leq 3/8$. We conclude then that $\mathcal{M}'$ decides A.

To simulate $\mathcal{M}$ over w with the oracle calls simulated digitally $\mathcal{M}'$ needs polynomial space on $|\langle w, f(|w|)\rangle|$, since $\mathcal{M}$ is bounded in polynomial space and the oracle is simulated in polynomial space on $|w|$. We conclude then that $\mathcal{M}'$ operates in polynomial space and thus $A \in BPPSPACE//poly$. □

The computation time $\mathcal{M}'$ takes on $\langle w, f(|w|)\rangle$ is the time $\mathcal{M}$ takes on w plus the time needed to simulate the experiments. The time needed to simulate $\mathcal{M}$ over w and simulate the experiments can be at most exponential on $|w|$. The time needed to simulate the experiments is exponential on $|w|$ if and only if $\mathcal{M}$ makes an exponential amount of oracle calls.

Theorem 3.7.9. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision SmSM in polynomial space, using the time schedule, if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Propositions 3.7.3 and 3.7.8. □

We can then state all the results concerning the standard analogue-digital smooth scatter machine bounded in polynomial space.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$
Upper Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$

Table 3.4: Standard communication protocol SmSM

Chapter 4

Generalized scatter machine

In the previous chapter we studied the computational power of the standard scatter machine restricted in polynomial space. Through this chapter we intend to study the computational power of the scatter machine restricted in polynomial space with the generalized communication protocol, for both the sharp and the smooth scatter experiment with any of the three precision assumptions.

We use here a proof technique introduced in Chapter 1, and we have some proofs that are exactly as the ones in Chapter 3 since the protocol is not enough to change the computational power of the machine.

Analogously to the beginning of Chapter 3 we can start by importing some lower bounds for the generalized scatter machine bounded in polynomial space. Since the generalized scatter machine can decide all the sets a standard scatter machine decides (see Proposition 2.3.1). We conclude that the complexity classes that the standard scatter machine bounded in polynomial space compute are at least embedded in the lower bounds for the generalized scatter machine bounded in polynomial space.

We can state then the following lower bounds for the analogue-digital sharp scatter machine bounded in polynomial space:

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$

Table 4.1: Generalized communication protocol ShSM

And we can state the following lower bounds for the analogue-digital smooth scatter machine bounded in polynomial space:

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$

Table 4.2: Generalized communication protocol SmSM

4.1 Sharp scatter machine

Lower bounds

For the next two propositions we need an advice function $f : \mathbb{N} \to \Sigma^*$ introduced just above Proposition 1.2.11 (stating that P/exp is the class of all languages, 2^{Σ^*}). The advice function f is constructed over a set $A \subseteq \Sigma^*$ in such a way that $f(n) = f_1^n f_2^n \cdots f_{2^n}^n$, where f_i^n is either 0 or 1 depending on if the i^{th} word with size n of Σ^*, ordered lexicographically, belongs or not to the set A, respectively. For any $n \in \mathbb{N}$ $|f(n)| = 2^n$.

Proposition 4.1.1. *Any set $A \subseteq \Sigma^*$ is decidable by an error-free ShSM in polynomial space.*

Proof. Consider a set $A \subseteq \Sigma^*$ lexicographically ordered. Let the advice function $f : \mathbb{N} \to \Sigma^*$ be as described above for the the set A, the prefix advice function $g : \mathbb{N} \to \Sigma^*$ be such that $g(n) = f(0)\#f(1)\#\cdots\#f(n)$ and $y \in]0,1[$ be the real number $y = 0.x(g)$ (where $x(g)$ is the encoding x described in Section 3.1 to the function g). The advice word $f(0)\#f(1)\#\cdots\#f(n)$ is given with two symbols from Σ^* for each of the symbols 0, 1 or $\#$, so that we have the advice words over Σ^* and we are able to apply the encoding x to g. We then have that[1] $|g(n)| = 2 \cdot (2^{n+1} - 1) + 2n$ for any $n \in \mathbb{N}$, since $|f(n)| = 2^n$.

Consider the error-free ShSM $\mathcal{M}$ with vertex position $y = 0.x(g)$ which for an input w with $|w| = n$ computes the position of w within all the n sized words in Σ^* ordered lexicographically, lets say that w is the i^{th} word. Afterwards, $\mathcal{M}$ performs $2^{n+1} - 1 + 2n + 2i$ oracle calls with the search method described in Section 3.1 (which makes the query word into $q = x(f(0)\#f(1)\#\cdots\#f(n-1)\#f_1^n f_2^n \cdots f_i^n)$) and accepts w if and only if the answer from the last call is right, meaning that $f_i^n = 1$ and so $w \in A$. We conclude then that $\mathcal{M}$ decides A.

To compute the position of w ($|w| = n$) within the n sized words in Σ^* ordered lexicographically, count the $2^{n+1} - 1 + 2n + 2i$ oracle calls and perform them $\mathcal{M}$ needs polynomial space on n. So, we can conclude that $\mathcal{M}$ operates in polynomial space (actually in linear space). □

The computation time of $\mathcal{M}$ on w ($|w| = n$) is the time needed to compute the position of w and count the oracle calls plus the time needed to perform the experiments. The three times specified are exponential on n. To compute a position within 2^n words ordered lexicographically and to count until $2^{n+1} - 1 + 2n + 2i$ ($1 \leq i \leq 2^n$) $\mathcal{M}$ uses clearly an exponential time on n, but no more than that. To perform the $2^{n+1} - 1 + 2n + 2i$ oracle calls $\mathcal{M}$ needs an exponential time on n since the oracle calls take a constant physical time.

Proposition 4.1.2. *Any set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision ShSM in polynomial space.*

Proof. The proof of this Proposition is, almost in every detail, similar to the proof of Proposition 4.1.1. The remaining difference is that, when using the oracle with the search method of Section 3.1 and the error-prone arbitrary precision protocol, we need to pad the query word with three 0's in order to get the desired error (accuracy interval) and obtain the same results as with the error-free scatter machine. Follows a full proof for the reader seeking the details in this place.

Consider a set $A \subseteq \Sigma^*$ lexicographically ordered. Let the advice function $f : \mathbb{N} \to \Sigma^*$ be as described above for the the set A, the prefix advice function $g : \mathbb{N} \to \Sigma^*$ be such that $g(n) = f(0)\#f(1)\#\cdots\#f(n)$ and $y \in]0,1[$ be the real number $y = 0.x(g)$ (where $x(g)$ is the encoding x described in Section 3.1 to the function g). The advice word $f(0)\#f(1)\#\cdots\#f(n)$ is given with two symbols from Σ^* for each of

[1] $2^0 + 2^1 + \cdots + 2^n = 2^{n+1} - 1$.

the symbols 0, 1 or $\#$, so that we have the advice words over Σ^* and we are able to apply the encoding x to g. We then have that $|g(n)| = 2 \cdot (2^{n+1} - 1) + 2n$ for any $n \in \mathbb{N}$, since $|f(n)| = 2^n$.

Consider the error-prone arbitrary precision ShSM $\mathcal{M}$ with vertex position $y = 0.x(g)$ which for an input w with $|w| = n$ computes the position of w within all the n sized words in Σ^* ordered lexicographically, lets say that w is the i^{th} word. Afterwards, $\mathcal{M}$ performs $2^{n+1} - 1 + 2n + 2i$ oracle calls with the search method described in Section 3.1 (which makes the query word into $q = x(f(0)\#f(1)\#\cdots\#f(n-1)\#f_1^n f_2^n \cdots f_i^n)$) and accepts w if and only if the answer from the last call is right, meaning that $f_i^n = 1$ and so $w \in A$. We conclude then that $\mathcal{M}$ decides A.

To compute the position of w ($|w| = n$) within the n sized words in Σ^* ordered lexicographically, count the $2^{n+1} - 1 + 2n + 2i$ oracle calls and perform them $\mathcal{M}$ needs polynomial space on n. So, we can conclude that $\mathcal{M}$ operates in polynomial space (actually in linear space). □

The computation time of $\mathcal{M}$ on w ($|w| = n$) is the time needed to compute the position of w and count the oracle calls plus the time needed to perform the experiments. The three times specified are exponential on n. To compute a position within 2^n words ordered lexicographically and to count until $2^{n+1} - 1 + 2n + 2i$ ($1 \leq i \leq 2^n$) $\mathcal{M}$ uses clearly an exponential time on n, but no more than that. To perform the $2^{n+1} - 1 + 2n + 2i$ oracle calls $\mathcal{M}$ needs an exponential time on n since the oracle calls take a constant physical time.

Using the generalized protocol for the fixed precision assumption we are not able to access an exponentially long advice in polynomial space. Due to the necessity of counting the oracle calls used in the Chebyshev's inequality (see Section 3.2), we need an exponential space to be able to count the calls needed to get an exponential amount of triples from the binary form of the vertex position. For an input of size n, we need an exponential space on n (we need $6 \cdot 2^n + 8 + c + 1$ cells of a tape) to be able to count $2^{6 \cdot 2^n + 8 + c}$ oracle calls so that we would get 2^n triples from the binary expansion of the vertex position with error probability smaller than 2^{-c} (see Proposition 3.2.2). Note that we should be able to get $2^{n+1} - 1 + 2n + 2i$ triples ($1 \leq i \leq 2^n$) to make a proof like the previous two. Therefore, an error-prone finite precision ShSM in polynomial space decides the same with the generalized protocol as it decides with the standard one (the same happens with the smooth scatter experiment). We here state the results, for detailed proofs see Section 3.6.

Proposition 4.1.3. *If a set $A \in BPPSPACE//poly$, then A is decidable by an error-prone finite precision ShSM in polynomial space.*

Proof. The proof of this Proposition is exactly the same as the proof of Proposition 3.6.3. The relevant difference between the generalized and the standard protocols is that we can use an unbounded amount of cells from the query tape with the generalized protocol, even with a bounded space scatter machine. For the lower bound with the error-prone finite precision protocol we do not use this difference since all the oracle calls with the purpose of obtaining the vertex position are performed with the same query.

For a detailed proof see Proposition 3.6.3 □

Upper bounds

Theorem 4.1.4. *The sets decidable by an error-free ShSM in polynomial space are all the sets $A \in \Sigma^*$.*

Proof. It follows from Proposition 4.1.1 and the fact that all sets (languages) belong to Σ^*. □

Theorem 4.1.5. *The sets decidable by an error-prone arbitrary precision ShSM in polynomial space are all the sets $A \in \Sigma^*$.*

Proof. It follows from Propositions 4.1.2 and the fact that all sets (languages) belong to Σ^*. □

Proposition 4.1.6. *If a set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision ShSM in polynomial space, then $A \in BPPSPACE//poly$.*

Proof. The proof of this Proposition is exactly the same as the proof of Proposition 3.6.8. The relevant difference between the generalized and the standard protocols is that we can use an unbounded amount of cells from the query tape with the generalized protocol, even with a bounded space scatter machine. For the upper bound with the error-prone finite precision protocol we aim to apply Proposition 3.4.3, so we need to simulate the scatter machine with a difference from the real probability values for the oracle answers of at most $2^{-p(n)-4}$. Thus, we only need a polynomial amount of digits from the query in order to probabilistically choose a shot position to compare with the vertex position, keeping the approximate probabilities close enough to the real ones.

For a detailed proof see Proposition 3.6.8 □

Theorem 4.1.7. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision ShSM in polynomial space if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Propositions 4.1.3 and 4.1.6. □

We can then state all the results concerning the generalized analogue-digital sharp scatter machine bounded in polynomial space.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	2^{Σ^*}	2^{Σ^*}	$BPPSPACE//poly$
Upper Bound	2^{Σ^*}	2^{Σ^*}	$BPPSPACE//poly$

Table 4.3: Generalized communication protocol ShSM

4.2 Smooth scatter machine

The smooth scatter machine is intrinsically different from the sharp one, in this case we need to guarantee that the experiment do not keeps running indefinitely. To this purpose we may need to use the time schedule, but we are limited to exponential time schedules since we use scatter machines bounded in polynomial space (see Proposition 2.1.2). With the fixed precision assumption we can only access a polynomially long advice, as well as with the sharp scatter experiment, in polynomial space, since we need further space to be able to count the oracle calls needed to obtain a longer advice with the Chebyshev's inequality. To the other precision assumptions, if we want to use the time schedule, we are limited to use polynomially long advices as well, since for an exponential time schedule we can only use polynomially sized queries to have the time schedule running in polynomial space on the input size.

Proposition 4.2.1. *A set $A \in \Sigma^*$ is decidable by a generalized smooth scatter machine bounded in polynomial space, using the time schedule, if and only if is decidable by a standard smooth scatter machine bounded in polynomial space, using the time schedule.*

Proof. By Proposition 2.3.1 we have that a set decidable by a standard scatter machine is decidable by a generalized scatter machine.

If $\mathcal{M}$ is a generalized SmSM bounded in polynomial space, using the time schedule, which may use queries as long as the input, then $\mathcal{M}$ can use at most an exponential time schedule (see Proposition 2.1.2). The exponential clocks depicted in Section 2.1 use polynomial space on the size of their inputs and thus, used as time schedules the clocks use polynomial space on the size of the queries. We conclude then that in order to have $\mathcal{M}$ running in polynomial space we need to consider queries of at most polynomial size. If $\mathcal{M}$ only use polynomially sized queries, then a standard SmSM can use exactly the same queries $\mathcal{M}$ uses in any of its computations. Therefore, a standard SmSM can decide all the sets that $\mathcal{M}$ decides, and the desired equivalence follows. □

Lower bounds

If we want to use the time schedule, we obtain the same results as with the standard protocol as explained above. We can decide $PSPACE/poly$ or $BPPSPACE//poly$ with a SmSM with the error-free protocol or the error-prone arbitrary precision protocol, respectively (see Propositions 3.7.1 and 3.7.2 for detailed proofs). If we do not use the time schedule, we can get the same results as in the sharp case, but the scatter machines take, in general, more than exponential time. However, the time the scatter machines take is always guaranteed as finite.

For the lower bounds without time schedule we need the advice function $f : \mathbb{N} \to \Sigma^*$ detailed for the sharp case lower bound proofs (see Section 4.1).

Proposition 4.2.2. *Any set $A \subseteq \Sigma^*$ is decidable by an error-free SmSM in polynomial space.*

Proof. The proof of this Proposition is, almost in every detail, similar to the proof of Proposition 4.1.1. The remaining difference is that, when using the oracle with the search method of Section 3.1 and the smooth scatter experiment, the scatter machine takes more than a constant physical time but it can get the same answers from the oracle and the same computational results. Follows a full proof for the reader seeking the details in this place.

Consider a set $A \subseteq \Sigma^*$ lexicographically ordered. Let the advice function $f : \mathbb{N} \to \Sigma^*$ be as described in Section 4.1 for the the set A, the prefix advice function $g : \mathbb{N} \to \Sigma^*$ be such that $g(n) = f(0)\#f(1)\#\cdots\#f(n)$ and $y \in]0,1[$ be the real number $y = 0.x(g)$ (where $x(g)$ is the encoding x described in Section 3.1 to the function g). The advice word $f(0)\#f(1)\#\cdots\#f(n)$ is given with two

symbols from Σ^* for each of the symbols 0, 1 or $\#$, so that we have the advice words over Σ^* and we are able to apply the encoding x to g. We then have that $|g(n)| = 2 \cdot (2^{n+1} - 1) + 2n$ for any $n \in \mathbb{N}$, since $|f(n)| = 2^n$.

Consider the error-free SmSM $\mathcal{M}$ with vertex position $y = 0.x(g)$ which for an input w with $|w| = n$ computes the position of w within all the n sized words in Σ^* ordered lexicographically, lets say that w is the i^{th} word. Afterwards, $\mathcal{M}$ performs $2^{n+1} - 1 + 2n + 2i$ oracle calls with the search method described in Section 3.1 (which makes the query word into $q = x(f(0)\#f(1)\#\cdots\#f(n-1)\#f_1^n f_2^n \cdots f_i^n)$) and accepts w if and only if the answer from the last call is right, meaning that $f_i^n = 1$ and so $w \in A$. We conclude then that $\mathcal{M}$ decides A.

To compute the position of w ($|w| = n$) within the n sized words in Σ^* ordered lexicographically, count the $2^{n+1} - 1 + 2n + 2i$ oracle calls and perform them $\mathcal{M}$ needs polynomial space on n. So, we can conclude that $\mathcal{M}$ operates in polynomial space (actually in linear space). □

The computation time of $\mathcal{M}$ on w ($|w| = n$) is the time needed to compute the position of w and count the oracle calls plus the time needed to perform the experiments. The time needed to compute the position of w and count the oracle calls is exponential on n. The time needed to perform the $2^{n+1} - 1 + 2n + 2i$ oracle calls is more than $2^{3 \cdot 2^{n+1} + 6n + 6i - 4}$ units of time, since the j^{th} call needs more than 2^{3j-1} units of time to be answered. On the other hand, the time needed to perform the $2^{n+1} - 1 + 2n + 2i$ oracle calls is less than $2^{3 \cdot 2^{n+1} + 6n + 6i}$ units of time, since the j^{th} call needs less than 2^{3j+2} units of time to be answered.

Proposition 4.2.3. *Any set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision SmSM in polynomial space.*

Proof. The proof of this Proposition is, almost in every detail, similar to the proof of Proposition 4.2.2, as well as in the sharp case. The remaining difference is that, when using the oracle with the search method of Section 3.1 and the error-prone arbitrary precision protocol, we need to pad the query word with three 0's in order to get the desired error (accuracy interval) and obtain the same results as with the error-free scatter machine. Follows a full proof for the reader seeking the details in this place.

Consider a set $A \subseteq \Sigma^*$ lexicographically ordered. Let the advice function $f : \mathbb{N} \to \Sigma^*$ be as described in Section 4.1 for the the set A, the prefix advice function $g : \mathbb{N} \to \Sigma^*$ be such that $g(n) = f(0)\#f(1)\#\cdots\#f(n)$ and $y \in]0, 1[$ be the real number $y = 0.x(g)$ (where $x(g)$ is the encoding x described in Section 3.1 to the function g). The advice word $f(0)\#f(1)\#\cdots\#f(n)$ is given with two symbols from Σ^* for each of the symbols 0, 1 or $\#$, so that we have the advice words over Σ^* and we are able to apply the encoding x to g. We then have that $|g(n)| = 2 \cdot (2^{n+1} - 1) + 2n$ for any $n \in \mathbb{N}$, since $|f(n)| = 2^n$.

Consider the error-prone arbitrary precision SmSM $\mathcal{M}$ with vertex position $y = 0.x(g)$ which for an input w with $|w| = n$ computes the position of w within all the n sized words in Σ^* ordered lexicographically, lets say that w is the i^{th} word. Afterwards, $\mathcal{M}$ performs $2^{n+1} - 1 + 2n + 2i$ oracle calls with the search method described in Section 3.1 (which makes the query word into $q = x(f(0)\#f(1)\#\cdots\#f(n-1)\#f_1^n f_2^n \cdots f_i^n)$) and accepts w if and only if the answer from the last call is right, meaning that $f_i^n = 1$ and so $w \in A$. We conclude then that $\mathcal{M}$ decides A.

To compute the position of w ($|w| = n$) within the n sized words in Σ^* ordered lexicographically, count the $2^{n+1} - 1 + 2n + 2i$ oracle calls and perform them $\mathcal{M}$ needs polynomial space on n. So, we can conclude that $\mathcal{M}$ operates in polynomial space (actually in linear space). □

The computation time of $\mathcal{M}$ on w ($|w| = n$) is the time needed to compute the position of w and count the oracle calls plus the time needed to perform the experiments. The time needed to compute the position of w and count the oracle calls is exponential on n. The time needed to perform the $2^{n+1} -$

$1 + 2n + 2i$ oracle calls is more than $2^{3 \cdot 2^{n+1}+6n+6i-4}$ units of time, since the j^{th} call needs more than 2^{3j-1} units of time to be answered. On the other hand, the time needed to perform the $2^{n+1} - 1 + 2n + 2i$ oracle calls is less than $2^{3 \cdot 2^{n+1}+6n+6i+1}$ units of time, since the j^{th} call needs less than 2^{3j+3} units of time to be answered.

This scatter machine can either take more time than the error-free one from Proposition 4.2.2 or less time. The time of each shot can be increased for the double or reduced by a third, the lower bound time we describe in last paragraph is the same for both Propositions since we took a lower bound which works for both cases.

For the fixed precision assumption we are not able to access an exponentially long advice in polynomial space by the same reason as in the sharp scatter experiment case. We here state the results, for detailed proofs see Section 3.7.

Proposition 4.2.4. *If a set $A \in BPPSPACE//poly$, then A is decidable by an error-prone finite precision SmSM in polynomial space, which makes use of the time schedule.*

Proof. The proof of this Proposition is exactly the same as the proof of Proposition 3.7.3. The relevant difference between the generalized and the standard protocols is that we can use an unbounded amount of cells from the query tape with the generalized protocol, even with a bounded space scatter machine. For the lower bound with the error-prone finite precision protocol we do not use this difference since all the oracle calls with the purpose of obtaining the vertex position are performed with the same query.

For a detailed proof see Proposition 3.7.3 □

Upper bounds

With the first two precision assumptions (infinite and arbitrary precisions) we get different results according with if we want the scatter machine to use the time schedule or not. As well as with the standard communication protocol, in the lower bound proofs, if we want to use the time schedule this must to be of an exponential growing rate.

Theorem 4.2.5. *A set $A \subseteq \Sigma^*$ is decidable by an error-free SmSM in polynomial space, using an exponential time schedule, if and only if $A \in PSPACE/poly$.*

Proof. It follows from Proposition 4.2.1 and Theorem 3.7.5. □

Theorem 4.2.6. *The sets decidable by an error-free SmSM in polynomial space, without using the time schedule, are all the sets $A \in \Sigma^*$.*

Proof. It follows from Proposition 4.2.2 and the fact that all sets (languages) belong to Σ^*. □

Theorem 4.2.7. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone arbitrary precision SmSM in polynomial space, using an exponential time schedule, if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Proposition 4.2.1 and Theorem 3.7.7. □

Theorem 4.2.8. *The sets decidable by an error-prone arbitrary precision SmSM in polynomial space, without using the time schedule, are all the sets $A \in \Sigma^*$.*

Proof. It follows from Proposition 4.2.3 and the fact that all sets (languages) belong to Σ^*. □

Proposition 4.2.9. *If a set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision SmSM in polynomial space, using the time schedule, then $A \in BPPSPACE//poly$.*

Proof. The proof of this Proposition is exactly the same as the proof of Proposition 3.7.8. The relevant difference between the generalized and the standard protocols is that we can use an unbounded amount of cells from the query tape with the generalized protocol, even with a bounded space scatter machine. For the upper bound with the error-prone finite precision protocol we aim to apply Proposition 3.4.5, so we need to simulate the scatter machine with a deviation from the real probability values for the oracle answers of at most $2^{-p(n)-4}$. Thus, we only need a polynomial amount of digits from the query in order to probabilistically choose a shot position to compare with the vertex position, keeping the approximate probabilities close enough to the real ones.

For a detailed proof see Proposition 3.7.8 □

Theorem 4.2.10. *A set $A \subseteq \Sigma^*$ is decidable by an error-prone finite precision SmSM in polynomial space, using the time schedule, if and only if $A \in BPPSPACE//poly$.*

Proof. It follows from Propositions 4.2.4 and 4.2.9. □

We can then state all the results concerning the generalized analogue-digital smooth scatter machine bounded in polynomial space.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound with time schedule	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$
Lower Bound without time schedule	2^{Σ^*}	2^{Σ^*}	—
Upper Bound	2^{Σ^*}	2^{Σ^*}	$BPPSPACE//poly$

Table 4.4: Generalized communication protocol SmSM

Note that using the time schedule, the upper bounds are also equal to the upper bounds of the standard case (see Section 3.7). To compute 2^{Σ^*} with a smooth scatter machine we must be able to use exponential advices, and with that purpose we must use exponentially sized queries. Exponentially sized queries take at least $2^{3 \cdot 2^n - 1}$ units of time to be answered (see Proposition 4.2.2) and a time schedule needs $3 \cdot 2^n - 1$ units of space to tick that amount of transitions (see Section 2.1).

Chapter 5

Conclusion

With this work we intended to study the computational power of a particular analogue-digital machine bounded in polynomial space, the scatter machine, which consists of a Turing machine coupled with a physical experiment, the scatter experiment. This computation model was already studied concerning polynomial time restrictions and we established the results for the polynomial space restrictions.

With an analogue-digital machine of this nature, there are some issues we must discuss concerning its technical behaviour. The scatter experiment has two different versions, the sharp wedge and the smooth wedge cases. With the smooth wedge we have a physical experiment with intrinsic time costs, and we should deal with it in order to guarantee that no run of infinite time is performed by the experiment in a computation of the analogue-digital machine. Another technical issue is the communication protocol between the digital and the analogue components, which must be adapted to the precision that the analogue component can be set.

Concerning the time issue of the smooth scatter experiment we use a clock to bound the time of each run of the experiment. This clock is called time schedule, and as we aim to study scatter machines bounded in polynomial space we must understand which clocks we can use with that restriction. This is a former technique on the subject already used for the time restriction cases, where the clocks to be used must correspond to the general time restrictions. We concluded that in polynomial space we can use clocks with an exponential growing rate or lower.

Concerning the communication protocols, we studied the three usual different precision assumptions, the infinite precision, the arbitrary precision and the fixed precision. The results to the time restrictions are established to each one of these precisions with a communication protocol which makes use of a distinguished tape and three or four distinguished states. The Turing machine writes in the query tape the parameters to the experiment, and the conclusions of the experiment lead to the correspondent answer state. The former communication protocol used for the time restrictions is not sufficient in our case since the query tape is restricted by the polynomial space restriction to the machine. We presented a different communication protocol which leads us to different results, and we established the computational results for both protocols.

We concluded that a probabilistic scatter machine with a deterministic digital component is able to simulate a probabilistic Turing machine with an exponential amount of transitions, using also an exponential amount of time. All the usual probabilistic simulations can then be used in this case as they were used for the time restriction cases.

We use non-uniform complexity classes to describe our results since, as we are using a specific kind of oracle Turing machine, they seem to be appropriate to the purpose. We introduced a probabilistic uniform complexity class, the class of sets decidable by bounded error probabilistic Turing machines bounded in polynomial space ($BPPSPACE$), used then as support for non-uniform classes.

We established the computational power for both scatter machines, the sharp and the smooth, and in each case for the two communication protocols, the standard and the generalized. We used adaptations of the usual proof techniques, used for the time restriction cases, and another usual technique for when the power of the scatter machine allows us to compute all languages. The results for the sharp scatter machine can then be summarized in the following tables, respectively for the standard and the generalized protocols.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$
Upper Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$

Table 5.1: Standard communication protocol ShSM

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	2^{Σ^*}	2^{Σ^*}	$BPPSPACE//poly$
Upper Bound	2^{Σ^*}	2^{Σ^*}	$BPPSPACE//poly$

Table 5.2: Generalized communication protocol ShSM

The smooth scatter machine is not that simple to describe as we obtained different results depending on if we want to use or not the time schedule with the generalized communication protocol. The results for the standard smooth scatter machine can be summarized in the following table.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$
Upper Bound	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$

Table 5.3: Standard communication protocol SmSM

With the generalized protocol, we obtain the same results as the ones with the standard protocol if we use the time schedule, and we obtain the same results as the ones with the sharp scatter machine if we do not use the time schedule. All the discussion about the time schedule do not applies to the fixed precision, since we always need the time schedule in that case, even with the sharp scatter experiment.

The results for the generalized smooth scatter machine can be summarized in the following table. Note that the upper bounds with the generalized protocol are equal to the lower bounds if we want to use the time schedule, the specified upper bounds are without time schedule.

	Infinite	**Arbitrary**	**Fixed**
Lower Bound with time schedule	$PSPACE/poly$	$BPPSPACE//poly$	$BPPSPACE//poly$
Lower Bound without time schedule	2^{Σ^*}	2^{Σ^*}	—
Upper Bound	2^{Σ^*}	2^{Σ^*}	$BPPSPACE//poly$

Table 5.4: Generalized communication protocol SmSM

With this results we can observe that for the fixed precision assumption the results are not different from each other when we change either the communication protocol or the wedge version of the experiment. In the protocols case this equality comes from the fact that we always use the same cannon position to obtain the vertex position, the protocols then make no difference. Concerning the different versions of the experiment we have the same results as for the time restriction cases, where the computational power is the same with both the sharp and the smooth wedge. This equality comes from the fact that as we use the same cannon position to obtain the vertex position, we use the smooth wedge version of the experiment with a constant physical time as well as the sharp wedge version, which always has a constant physical time.

For future work we can suggest, for example, the study of the scatter machine bounded in polynomial space and clocked in quasi-exponential time ($2^{\log(n)^k}$, for some $k \in \mathbb{N}$). Here, as we do not bound the time resource of the general analogue-digital machine, we obtain scatter machines running in exponential time, except when using the generalized smooth scatter machine without the time schedule where the time is of the magnitude order of 2^{2^n}.

For this study it would be useful to use a notation analogous to the one in the conclusion of [7] ($\S$4) with a new parameter for the space restriction to the machine. This notation could also be useful to describe the results of our last table as we are, basically, considering different time restrictions with the same space restriction. The notation describes a complexity class based on the sets computed by the analogue-digital machines. In our case, with the scatter machine, we could have something like

$$SCATTER(t(n),\, s(n),\, f(n))$$

where $t(n)$ is the time restriction to the scatter machine, $s(n)$ is the space restriction and $f(n)$ is the bound for the time schedule. We clearly should have $f \in \mathcal{O}(t(n))$, otherwise the time restriction would not be followed.

References

[1] Tânia Ambaram, Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. Advances in unconventional computing, volume 1 (theory). In Andrew Adamatzky, editor, *An analogue-digital model of computation: Turing machines with physical oracles*, volume 22 of *Emergence, Complexity and Computation*, pages 73–115. Springer, 2016.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[3] José Luis Balcázar, Josep Días, and Joaquim Gabarró. *Structural Complexity I*. Springer-Verlag, 2nd edition, 1988, 1995.

[4] José Luis Balcázar and Montserrat Hermo. The structure of logarithmic advice complexity classes. *Theoretical Computer Science*, 207(1):217–244, 1998.

[5] Edwin Beggs, Pedro Cortez, José Félix Costa, and John V. Tucker. Classifying the computational power of stochastic physical oracles. page 122, 2016. Submited.

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

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

[8] Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. On the power of threshold measurements as oracles. In Giancarlo Mauri, Alberto Dennunzio, Luca Manzoni, and Antonio E. Porreca, editors, *Unconventional Computation and Natural Computation (UCNC 2013)*, volume 7956 of *Lecture Notes in Computer Science*, pages 6–18. Springer, 2013.

[9] Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. Computations with oracles that measure vanishing quantities. *Mathematical Structures in Computer Science*, page 149, 2016.

[10] Edwin Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. An Analogue-Digital Church-Turing Thesis. *International Journal of Foundations of Computer Science*, 25(4):373–389, 2014. in print.

[11] Edwin Beggs, José Félix Costa, and John V. Tucker. Limits to measurement in experiments governed by algorithms. *Mathematical Structures in Computer Science*, 20(06):1019–1050, 2010. Special issue on Quantum Algorithms, Editor Salvador Elías Venegas-Andraca.

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

[13] Edwin Beggs, José Félix Costa, and John V. Tucker. Three forms of physical measurement and their computability. *The Review of Symbolic Logic*, 7(4):618–646, 2014.

[14] Edwin Beggs and John 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.

[15] Olivier Bournez and Michel Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2):417–459, 1996.

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

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

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

[19] Kevin T. Kelly. *The Logic of Reliable Inquiry*. Oxford University Press, 1996.

[20] Hava T. Siegelmann. Computation Beyond the Turing Limit. *Science*, 268(5210):545–548, April 1995.

[21] Hava T. Siegelmann. *Neural networks and analog computation : beyond the Turing limit*. Birkhäuser, 1999.

[22] Alan Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[23] Damien Woods and Thomas J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(2005):227–258, April 2004.