André Nuno Carvalho Souto

Individual Information Measures: Applications to Computational Complexity

Departamento de Ciência de Computadores da
Faculdade de Ciências da Universidade do Porto

André Nuno Carvalho Souto

Individual Information Measures: Applications to Computational Complexity

Dissertation submitted to the Faculdade de Ciências da Universidade do Porto
in partial fulfillment of the requirements for the degree of Doutor em Ciência de Computadores

Thesis advisor:
Prof. Doutor Luís Filipe Antunes

External Promotion Commission:
Prof. Dr. L. Fortnow and
Prof. Dr. H.M. Buhrman

Departamento de Ciência de Computadores da
Faculdade de Ciências da Universidade do Porto
2010

To my Guardian

Acknowledgments

I am very grateful to several people and organizations.

For the financial support I thank the National Science Foundation: Fundação para a Ciência e Tecnologia (FCT), that made this dissertation and all my research possible by financing me directly with a PhD scholarship (SFRH/BD/28419/2006) and through the projects KCrypt (POSC/EIA/60819/2004) and CSI^2 (PTDC/EIA-CCO/099951/2008). I am also grateful to LIACC-Laboratório de Inteligência Artificial e Ciência de Computadores and Instituto de Telecomunicações, the two research laboratories where I had worked during my PhD for receiving me and giving me the best conditions for research.

Foremost, I would like to thank my advisor Professor Luís Filipe Antunes for all the support. He is font of inspiration and great scientific problems that allow me to grow as a scientist and researcher. You have been a good mentor. It has been a great pleasure working with you, not only for everything you had taught me scientifically but specially for friendship and great advices that you gave me. It means a lot to me having a great advisor that always stands there for me.

I would like to thank everyone that I ever met at DCC – Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto, but specially the fellows which I shared office with: Alexandre, Besik, Bruno, Claúdio, David, Hugo, Jorge, Marco, Nuno. I owe a huge and very special thank to dear friend Andreia for being in the next desktop always able to listen all my (crazy!) ideas, arguments and daily routines. I am also grateful to Professor Armando Matos for helpful discussions, his valuable knowledge, good mood and the revising this thesis.

I thank Professor Fernando Jorge Moreira, my Master thesis advisor, for permanent support and for weekly meetings for nonstandard ways of Birkhof's Theorem. It is great to see the "world" under a nonstandard vision!

All my scientific research have been obtained in collaborative efforts. I must thank all my coauthors, Luís Antunes, João Bernardes, Lance Fortnow, Armando Matos, Alexandre Pinto, Cristina Santos, Andreia Teixeira and Paul Vitányi, for their help and permission to use the results published in this dissertation.

From January to May 2007 I worked with Professor Harry Buhrman at CWI - Centrum voor Wiskunde en Informatica. I thank to all ISN4 members for receiving me and I specially thank to Harry and Ronald de Wolf for the hospitality and great advices. It has been a great pleasure meeting these great scientists. Thank you for teaching me Quantum Computation and letting me work on Quantum Mastermind. I learned a lot during this time and the (scientific!) advises that I got there are the most valuable lesson I took for my research career.

In February 2010, I joined Sophie Laplante at the University Paris Sud XI for a short visit. I thank her for the hospitality, sympathy and for the collaboration. The trip to Paris was very fruitful and allowed me to share new ideas in some problems we were working on. Hopefully we get it to the end.

In June 2010, I visited Professor Lance Fortnow at Northwestern University for few weeks. I would like to thank him for the hospitality, the great concert at Ravinia and specially for the brilliant ideas he shared. It was a great pleasure to work with someone like you. Your knowledge and wisdom are far more advanced which means that I learned a lot in those conversations we had. Complexity is sometimes complex to reach, but it definitely worth to give it a try!

I can not forget my family and friends. I thank to my mother, father, brother, Manuela, godfather, Fátima, uncles and cousins, Fernanda and Fernando Magalhães, Sameiro and Carlos Lima, Rui, Mafalda, Ricardo, Lena and kids and Ana Moura for all the support and for believing in me. Specially to my mother and my father and godfather that were always there to listen and support in those days that did not go that well. I love you all. (Especialmente à minha mãe, pai e padrinho que sempre estiveram presentes para ouvir, aconcelhar e apoiar naqueles dias que correram menos bem. Gosto muito de vocês.)

This doctoral thesis, is the first of its kind in the family and for that reason is indeed dedicated to them.

Finally, I would like to thank my lovely girlfriend for all kindness, comprehension, friendship and love. Lara, I am blessed to have someone like you right by my side. You are the one that is always there listening, supporting me in the most difficult times and making me believe that is possible to do this. Your friendship, support and belief made me stronger and a better person. My love for you is pure and means everything to me. Our “math” is very simple, “pure”, special, complete and it never ends. I love you so much...

Abstract

Kolmogorov complexity is a classical measure of information for individual objects – as the length of the shortest description of it – and it can be used to characterize many properties of computational complexity classes. In this dissertation, we explore this line of research in three fundamental ways:

1. In the first, we use Kolmogorov complexity to (re)define the notion of sophistication for infinite sequences.

2. In the second, we use Kolmogorov complexity, more precisely computational depth, to describe some **SAT** formulas for which we can find assignments probabilistically in polynomial time.

3. In the last one, we use a sublinear time bounded version of Kolmogorov complexity to describe classes of languages that have Complexity Cores, i.e., sets of hardest instances.

Random strings convey maximal information since they have higher Kolmogorov complexity. However, it is very unlikely that these strings have "useful" or meaningful information, as, in general, they are usually considered to be the result of noise. There are two known approaches to quantify the subjective notion of meaningful information: Sophistication and Computational Depth.

The Sophistication is usually based on the Kolmogorov structure function and essentially measures the regularity present in the string. In this dissertation we revisit this notion for infinite sequences, giving a meaningful definition of this measure and proving some relationships with classical notions of dimension (Hausdorff and Packing).

Computational Depth is based on the difference between the time bounded Kolmogorov complexity and its unbounded version and, as a consequence, it measures the effort required to produce the object from a short description. We studied the connection between computational depth and sophistication for infinite sequences proving that they are of different kind.

In recent years, much effort has been done to explore computational depth as a measure in computational complexity. We continue this sequel proving that if a Boolean formula has a satisfying assignment of computational depth $\mathtt{d}$, then we can probabilistically find an assignment for that formula in time exponential in $\mathtt{d}$. Regarding the converse question we show that under a standard hardness assumption the converse also holds and that under the unlikely, but open case, that $\mathbf{BPP} = \mathbf{FewP} = \mathbf{EXP}$, one can find formulas that have a single solution of high computational depth which can be found quickly using a probabilistic algorithm. Using similar techniques we show that one can not increase the depth of a string efficiently and if $\mathbf{BPP} = \mathbf{EXP}$ we show examples where one can produce a string of high depth from a string of very low depth.

In the last part of this dissertation we study the connection between computational depth and complexity cores. Informally, a polynomial complexity core of a language A is the kernel of hard instances, i.e., given some polynomial time algorithm it only decides a finite number of theses instances.

It is known that any recursive set that is not in $\mathbf{P}$ has an infinite polynomial complexity core and that if each algorithm for a language has a non-sparse set of "hard" inputs, then, in fact, the language has a non-sparse proper polynomial complexity core. In order to establish a connection between computational depth of the characteristic sequences of languages and the existence of (proper) complexity cores we study the computational depth of sets in the classes $\mathbf{P}$, $\mathbf{EXP}$, $\mathbf{FULL\text{-}P/log}$ and $\mathbf{P/poly}$. We give a characterization of the recursive sets not in $\mathbf{P}$ and thus admitting a polynomial complexity core and also prove that if a recursive set is not in $\mathbf{P/poly}$ then the depth of its characteristic sequence is high almost everywhere. We also prove that any Turing machine deciding a proper complexity core of exponential density can not recognize the language in average polynomial time when a time bounded version of the universal distribution is considered.

Resumo

A complexidade de Kolmogorov é uma medida clássica de informação para objectos individuais e pode ser usada para caracterizar muitas propriedades de clases de complexidade computacional. Nesta dissertação, exploramos esta linha de investigação de três formas fundamentais:

1. Na primeira, usamos a complexidade de Kolmogorov para redefinir a noção de sofisticação para sequências infinitas.

2. Na segunda, usamos a complexidade de Kolmogorov, mais precisamente, a profundidade computacional para descrever algumas fórmulas de **SAT** para as quais podemos encontrar atribuições em tempo probabilístico polinomial.

3. Na última, usamos a versão limitada por tempo sublinear da complexidade de Kolmogorov para descrever classes de linguagens que admitem *Complexity Cores*, i.e., conjuntos de instâncias difícies.

Os objectos aleatórios contêm informação máxima uma vez que têm complexidade de Kolmogorov máxima. Contudo, é muito improvável que estes objectos contenham informação útil ou com significado uma vez que estes são considerados como sendo ruído. Existem duas aproximações conhecidas para quantificar a noção subjectiva de informação útil: sofisticação e profundidade computacional.

A primeira é usualmente baseada na função de estrutura de Kolmogorov e essencialmente mede as regularidades presentes num objecto. Nesta dissertação revisitamos esta noção para sequências infinitas, dando uma definição desta medida e provando algumas relações com noções clássicas de dimensão (Dimensão de Hausdorff e *Packing*).

A última medida é baseada na diferença entre a versão limitada pelo tempo da complexidade de Kolmogorov e a sua versão ilimitada e, como consequência, esta mede o esforço necessário para produzir o objecto partindo de uma sua descrição mínima. Estudamos a relação entre profundidade computacional e sofisticação para sequências infinitas provando que são de natureza diferente.

Nos últimos anos, tem sido feito um grande esforço para explorar a profundidade computacional como uma medida para a complexidade computacional. Continuamos esta linha de investigação provando que se uma fórmula, que tem uma atribuição satisfazível de profundidade computacional $\mathtt{d}$, então podemos probabilisticamente encontrar uma atribuição que torna essa fórmula verdadeira em tempo exponencial em $\mathtt{d}$. No que diz respeito à implicação contrária, mostramos que sob assunções computacionais standard esta verifica-se e também que sob o caso improvável, mas ainda em aberto, em que $\mathbf{BPP} = \mathbf{FewP} = \mathbf{EXP}$, podemos encontrar fórmulas que têm uma única solução com alta profundidade computacional que pode ser encontrada usando um algoritmo probabilístico eficiente. Usando técnicas semelhantes mostramos que não podemos incrementar a profundidade computacional eficientemente e no caso de $\mathbf{BPP} = \mathbf{EXP}$ apresentamos exemplos em que podemos produzir objectos de alta profundidade computacional partindo de um com baixa profundidade computacional.

Na última parte desta dissertação estudamos a relação entre profundidade computacional e *complexity cores*. Informalmente, um *complexity core* polinomial de uma linguagem $\mathtt{A}$ é o núcleo de instâncias difíceis, ou seja, instâncias que, excepto um número finito, não podem ser decididas por um qualquer algoritmo em tempo polinomial.

É sabido que qualquer conjunto recursivo que não esteja em $\mathbf{P}$ admite um *complexity core* polinomial infinito e que se qualquer algoritmo para essa linguagem tem um conjunto de inputs "difíceis" que não é esparso então, de facto, a linguagem admite um *complexity core* polinomial próprio, também ele não esparso. Para estabelecer a relação entre a profundidade computacional da sequência característica das linguagens e a existência de complexity cores (próprios) estudamos a profundidade computacional de conjuntos que estão nas classes de complexidade como $\mathbf{P}$, $\mathbf{EXP}$, $\mathbf{FULL\text{-}P/log}$ e $\mathbf{P/poly}$. Estabelecemos uma caracterização de conjuntos recursivos que não estão em $\mathbf{P}$ e portanto que admitem um *complexity core* polinomial e provamos que se um conjunto recursivo não está em $\mathbf{P/poly}$ então a profundidade computacional da sua sequência característica é quase sempre alta. Provamos também que qualquer máquina de Turing que decide um *complexity core* de densidade exponencial não pode reconhecer a linguagem em tempo polinomial em média, quando consideramos uma versão limitada pelo tempo da distribuição universal.

Contents

Chapter 1

Introduction

"What is randomness?" This question has been capturing the attention of part of the scientific community for long time. There are a few ways to define randomness. One ancestor idea to rigorously define randomness was inspired by intuitions from gambling. In practice, flipping coins is a random game exactly because one can not come up with a gambling strategy for predicting if we will get head or tail in the next through. In this dissertation we explore randomness using Kolmogorov complexity. The intuition for this approach is that a random object has no easily discernible patterns or structures which we could take advantage of to obtain a shorter description than just giving the object itself.

Solomonoff [Sol64], Kolmogorov [Kol65] and Chaitin [Cha66] independently defined complexity of an individual object. This measure is of different kind when compared with computational complexity. Instead of looking to the evolution of the computational effort necessary to decide larger and larger instances of a problem, turns its attention to the description of individual objects, using the length of the shortest program describing the object.

Lets compare the string "00000000000000000000" with the string of same length "10010011001001111000". Which of them is more complex? The first can be easily described by a sequence of "20 zeros" while the second does not have a simple pattern, or structure, that could be exploited to give a shorter description than just print it out bit by bit. So, intuitively the second string seems to be more "random" than the first one, although both of them have the same probability to be generated if we toss a fair coin. It is easy to see that a randomly generated string has, with high probability, high Kolmogorov complexity and thus it contains lots of information (see Theorem 2.3.13). This information may not be meaningful or useful from a computational point of view,

as it is usually considered to be noise.

In the literature we can find two different approaches to define useful information: measuring the amount of planing necessary to construct the object (static resources) or measuring the computational effort (dynamic resources), usually time, required to produce the object.

The former was proposed by Koppel [Kop87, Kop95, KA91] and is based on Kolmogorov's structure function (see for example [VV03]) which divides the smallest program in two parts: one part accounting for the useful regularity (i.e., the amount of planning used to describe the string) and another accounting for the remaining accidental information present in the object (i.e., the information necessary to distinguish that object among the objects sharing the same regularities). Kolmogorov suggested that the useful information, i.e., the first part of the description, is a representation of a finite set in which the object is a typical element, so that the two-part description is as small as the shortest one-part description. Gács, Tromp and Vitányi [GTV01] generalized this approach to *computable probability mass functions*. Koppel [Kop87, Kop95, KA91], using monotonic Kolmogorov complexity, expressed the useful information as a recursive function and called the resulting measure *sophistication*. The regularity turns out to be the length of the total program $\mathtt{p}$; and the accidental information, i.e. information that is consider not to have structure, is expressed as the length of the data used by $\mathtt{p}$ to produce the string or sequence. However, as Koppel observed, not all infinite sequences are describable and thus, the notion of sophistication, is not properly defined. Later, Antunes and Fortnow [AF09a] revisited the notion of sophistication for finite strings and proposed an alternative definition based on total programs. They proved the existence of strings with maximum sophistication. In this dissertation we revisit this notion of sophistication for infinite sequences, giving a meaningful definition of this measure and proving some relationships with classical notions of dimension, namely with constructive Hausdorff and constructive Packing dimension (see for example [Lut00b, May02, Lut03]).

The latter approach was introduced by Bennett [Ben88] that called the effort required to produce the object, *logical depth*. The intuition behind the definition of this concept is that a computationally deep string should take a lot of effort to be recovered from its shortest description while incompressible strings are trivially constructible from their shortest description, and therefore computationally shallow. After some attempts, Bennett [Ben88] formally defined the $\mathtt{s}$-significant logical depth of an object $\mathtt{x}$ as the time required by a standard universal Turing machine to generate $\mathtt{x}$ by a program $\mathtt{p}$ that can not itself be obtained from a program that is $\mathtt{s}$ or more bits shorter than $\mathtt{p}$. Thus, an

object is logically deep if it takes a lot of time to be generated from any short description. Latter Antunes *et al.* [AFvMV06] introduced the notion of computational depth, for finite strings, as the difference between the time bounded and the traditional unrestricted Kolmogorov complexity. We have seen a number of results about computational depth such as a generalization of sparse and random sets [AFvMV06], a characterization of worst-case running time of problems that run quickly on average over all polynomial time samplable distributions [AF09b]. We study the connection between computational depth and sophistication for infinite sequences proving that they are of different kind.

In order to show the applicability of computational depth we study its relationship with the satisfiability problem. In this problem we are given a boolean formula and are asked if there exists an attribution to the variables that turns the formula true. Cook [Coo71] proved that this problem is **NP** complete i.e., if we can come up with an efficient algorithm to satisfiability then we can efficiently solve any other **NP** problem. We study the possibility of having a probabilistic efficient algorithm that produces a satisfying assignment, proving that if a formula has an attribution of depth d then we can use a probabilistic algorithm running in exponential time in d to find another true assignment for that formula.

As another application of computational depth we studied a relationship with complexity cores. Heuristically, a polynomial complexity core of a language A is the kernel of hard instances, i.e., the instances whose membership in A is hard to decide in polynomial time. Rather than being hard for just a machine, a complexity core is the set of instances that, any machine, can only decide a finite number of its instances. The definition of polynomial complexity core is due to Lynch [Lyn75], who showed that any recursive set not in **P** has an infinite polynomial complexity core. Later, Orponen and Schöning [OS84] showed that, if each algorithm for a language has a non-sparse set of "hard" inputs, then the language has a non-sparse proper polynomial complexity core. Much work has been done using complexity cores. For example, in [JL92] the authors showed that exponential space complete sets have small complexity cores and in [Mun99], the author establishes a connection between complexity cores and instance complexity, a complexity measure derived from Kolmogorov complexity and proved that **NP**-hard sets must have super-polynomially dense hard instances, unless $\mathbf{P} = \mathbf{NP}$.

The connection between computational depth of languages and the existence of complexity cores is achieved by quantifying the computational depth of sets in the computational classes **P**, $\mathbf{EXP} = \mathbf{DTIME}(2^{\mathrm{poly}(n)})$, **FULL-P/log** and **P/poly**. First, we give a characterization of recursive sets not in **P** and thus admitting a polynomial complexity

core based on computational depth. Then, we prove that if a recursive set is not in **P/poly** then the depth of its characteristic sequence is high infinitely often.

The complexity of a problem is usually measured in terms of the worst case behavior of algorithms. Several algorithms with a worst-case bad performance have a good performance in practice, since instances requiring a large running time rarely occur. This duality was studied by Levin [Lev86], who introduced the theory of average case complexity, giving a formal definition of Average Polynomial Time for a language L and a distribution μ. Some languages may remain hard in the worst case but can be solved efficiently in average polynomial time for all reasonable distributions. We show that a Turing machine deciding a proper complexity core of exponential density can not recognize the language in average polynomial time when a time bounded version of the universal distribution is used.

Thesis Overview

The rest of this dissertation is organized as follows:

- In Chapter 2 we introduce the basic concepts, terminology and notation necessary for the rest of this work.

- In Chapther 3 we (re)define sophistication for infinite sequences and establish a connection with constructive Hausdorff and Packing dimension and with variants of computational depth. We also show that sophistication and computational depth are measures of different kind.

- In Chapter 4 we explore computational depth as a tool for finding true assignments for formulas in **SAT**. In particular, we show how probabilistically find an assignment of a Boolean formula that has a satisfying assignment of low computational depth. We also study if the converse holds and show that this is the case under reasonable hardness assumptions. Using similar ideas we show that one can not significantly increase the depth of a string in polynomial time. Once again if $\mathbf{BPP} = \mathbf{EXP}$ we show examples where one can produce a string of high depth from a string of low depth. Finally, we explore the question as to whether a triangle inequality holds for conditional computational depth.

- In Chapter 5 we study the relationship between complexity cores of a language and the computational depth of the characteristic sequence of the language based on Kolmogorov complexity. Motivated by the results of Chapter 4 we study the

possibility of characterizing the set of hard instances for a probabilistic polynomial time algorithm of **SAT** using computational depth. This turns out not to be possible. Although, we prove that a recursive set A has a complexity core (respectively, proper complexity core) if for all constants c (respectively, all polynomial $p(n)$), the computational depth of the characteristic sequence of A up to length n is larger than c (respectively larger than $p(n)$) infinitely often. We also explore the connection with average case complexity. In particular, we show that if a language has a complexity core of exponential density, then it can not be accepted in average polynomial time, when the strings are distributed according to a time bounded version of the universal distribution.

Chapter 2

Preliminaries

We introduce the terminology, notation and basic concepts from computational complexity, Kolmogorov complexity as well as its main properties. For a deeper study on these themes we suggest the reading of the following books and surveys:

- *Computational complexity: [Pap85], [Sip97], [AB09], [BDG95];*
- *Kolmogorov complexity: [LV08], [Gác93];*

The contents of this chapter is based on several sources, including [Ant02], [Lap97], [For89], [Pin07], [Pap85], [Sip97], [AB09], [BDG95], [LV08], [Gác93], [Gol01], [Vad99] and a few Wikipedia and Wofgang pages.

2.1 Notation and computational model

In this dissertation we use the binary alphabet $\Sigma = \{0, 1\}$, where 0 and 1 are called symbols or bits. A string, or a finite word, is a finite "combination" of bits and usually is denoted by small letters, like x, y, z. The set of all strings is represented by Σ^*; the empty word in denoted by ϵ. A standard enumeration of the elements of Σ^* is

$$(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), (000, 7), ...$$

This method of enumerating all strings, first ordering by increasing length and then, when two strings have the same length, ordering by lexical order, is called lexicographic-length conditional ordering.

Given a string $x \in \Sigma^*$, we define $|x|$, the *length* of x, as the number of bits composing x. We denote the set of all strings of length n by Σ^n.

Given $x \in \Sigma^n$ and $y \in \Sigma^m$, the concatenation xy of x and y, is defined as the string of Σ^{n+m} with the bits of x followed by the bits of y.

Definition 2.1.1. *We say that a string* x *is a* prefix *of the string* y *if there is a string* z *such that* $y = xz$.

Given a string $x \in \Sigma^n$, x_i and $x_{[1:m]}$ denote, respectively, the i^{th} bit and the prefix of length m of x.

Sometimes, it is useful to encode the strings by a self-delimiting code. Their utility is due to the fact that it determines the end of code word. One way to construct $\bar{x}$, the self-delimiting code of x, is by considering $\bar{x} := 1^n 0x$. Notice that $|\bar{x}| = 2|x| + 1$.

A language over Σ is a subset of Σ^* and is denoted by capital letters like: A, L...

A sequence is a infinite "combination" of bits and they are represented by small Greek letters, e.g., α, β, ω.

Definition 2.1.2. *The* characteristic sequence *of a language* L *is a sequence defined by* $\chi_L = \chi_1\chi_2\chi_3...$, *where* $\chi_i = 1$ *if and only if the* i^{th} *string in the lexicographic-length conditional ordering is in* L.

2.2 Turing machines and computational complexity

As a basic model of computation, we use the Turing machine, proposed by Alan Turing in 1937 [Tur37].

In Alan Turing's own words:

> *"The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail. We may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. He has also an unlimited supply of paper on which he does his calculations."*

Formally, a Turing machine can be defined in the following way:

Definition 2.2.1 (Deterministic Turing machine)**.** *A* deterministic Turing machine *is a tuple* $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, *where:*

- Q *is the set of finite states.*
- Σ *is the input alphabet.*
- Γ *is the tape alphabet.*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ *is a partial function called transition function, where* L *and* R *indicates the movement of the head. If at some point the function is not defined the machine stops.*
- $q_0 \in Q$ *is the initial state.*
- $q_a \in Q$ *is the accepting state.*
- $q_r \in Q$ *is the rejecting state.*

In the literature one can find different definitions of Turing machines, specially with more than one accepting state and rejecting state, with more than one tape (with a working tape, with an auxiliary tape), but these computational models are all equivalent. Another formulation of Turing machine considers a finite state control with an input tape and work tape (see Figure 2.2) and a read/write head for each of them. The finite control is the set of states (including the initial and accepting state), a transition function δ and the tapes' heads. Each head is located over one of the cells of its tape and can be moved right or left accordingly to δ. The transition function δ take the values under the head of the input and work tape, the current state and describes whether to move the head right or left, specify a possible change in contents of the work tape cell under the head and switches to a new state.

Before the Turing machine starts its computation, an input string x is placed on the input tape, one symbol in each cell. The tapes' heads are positioned on the first cell of each tape. The Turing machine accepts if it reaches the accepting state.

It is possible to define other variants of the Turing machine model where, for example non-determinism (where the transition function δ is multivalued) and randomness are allowed. Those models of computation are equivalent to the Turing machine model.

Definition 2.2.2. *A Turing machine* accepts a language L *if it accepts, as input strings, exactly those strings that are in* L. *We denote by* $L(M)$ *the language accepted by the Turing machine* M.

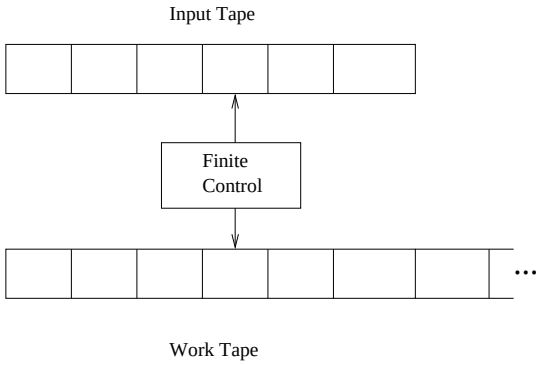

Figure 2.1: A Turing Machine.

A Turing machine runs in time t if for all input x, the number of times that the transition function is applied when the machine is initialized with x on the input tape, does not exceed $t(|x|)$. We will be denoting the running time of a Turing machine M on the input x by $\mathtt{time}_M(x)$. It is an important question in Computer Science, known as the "$\mathbf{P} = \mathbf{NP}$" question, whether a non-deterministic Turing machine can be simulated by a deterministic Turing machine using polynomial number of extra steps.

Definition 2.2.3. *Let $f : \Sigma^* \to \Sigma^*$ be a partial function and consider $t : \mathbb{N} \to \mathbb{N}$. A Turing machine* M computes f in time t *if for every input $x \in \Sigma^*$, the machine M when initialized with the input x, halts accepting x outputting $f(x)$ in $t(|x|)$ steps.*

Moreover, a Turing machine M computes a partial function f if on every input M produces $f(x)$. In this case we say that f is partial recursive. In particular, if f is a total function we say that f is recursive or computable.

The last definition allows us to define recursive and recursively enumerable sets.

Definition 2.2.4. *A set A is called recursive if the function χ_A is computable and is called recursively enumerable if the semi-characteristic function*

$$S_A(x) = \begin{cases} 1 & \text{if} \quad x \in A \\ \uparrow & \text{if} \quad x \notin A \end{cases}$$

is a partial recursive, where $S_A(x) = \uparrow$ means undefined.

One of the most well known example of a set that is recursively enumerable but not recursive is the so called *Halting Problem*.

Definition 2.2.5 (Halting Problem)**.** *Given the description of a Turing machine M and an input x, determine if M will halt on x.*

It is possible to enumerate effectively all Turing machines $T_1, T_2, ...$ This enumeration determines an effective enumeration of partial recursive function $\phi_1, \phi_2, ...$ where ϕ_i is the function computed by T_i for all i.

A Turing machine that can receive a codification of a pair of an encoding of any Turing machine T and an input x to T, and is able to simulate T on x is called a *Universal Turing machine*.

2.2.1 Computational complexity

One of the pioneer results in computational complexity, due to Hartmanis and Stearns [HS65] shows that if more time is allowed to a Turing machine then it can decide more languages. Once the model of computation is fixed we can group together languages that can be solved using similar amount of resources. Usually, a *complexity class* is defined by a model of computation and a resource bound for that model of computation.

When studying the computational complexity of a problem we are interested to know how the resources needed increase as the input size grows. The resources that are commonly studied are time and space. In the most part of this dissertation we will be interested in the assimptotic behavior of the growth rate of resources up to constant factors. So, we will be using the following notation:

Definition 2.2.6 (The Oh-notation)**.** *Let* $f, g : \mathbb{N} \to [0, \infty)$ *be two functions. We say that:*

- $f \in O(g)$ *if there is a constant* $c > 0$ *such that* $f(n) \leq c \cdot g(n)$*, for almost all* $n \in \mathbb{N}$*.*
- $f \in \Omega(g)$ *if there is a constant* $c > 0$ *such that* $f(n) \geq c \cdot g(n)$*, for almost all* $n \in \mathbb{N}$*.*
- $f \in \Theta(g)$ *if and only if* $f \in O(g)$ *and* $f \in \Omega(g)$*.*
- $f \in o(g)$ *if for any constant* $c > 0$ *and for sufficiently large* $n \in \mathbb{N}$*,* $f(n) \leq c \cdot g(n)$*.*
- $f \in \omega(g)$ *if for any constant* $c > 0$ *and for sufficiently large* $n \in \mathbb{N}$*,* $f(n) \geq c \cdot g(n)$*.*

Some of the most used bounds are: $O(1)$ called constant, $O(\log n)$ called logarithmic, $n^{O(1)}$ called polynomial, $2^{n^{o(1)}}$ called sub-exponential and $2^{n^{O(1)}}$ called exponential.

In this work all resource bounds t considered are time constructible, i.e., there is a Turing machine whose running time is exactly $t(n)$ on every input of size n.

Definition 2.2.7. *Let* $t : \mathbb{N} \to \mathbb{N}$ *be a constructible time bound. The class* **DTIME**[t] *is the set of languages that are accepted by some Turing machine running in time* $O(t(n))$*.*

As examples of deterministic time complexity classes we have:

1. $\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathbf{DTIME}[n^k]$, *the class of languages accepted in polynomial time;*
2. $\mathbf{EXP} = \bigcup_{k \in \mathbb{N}} \mathbf{DTIME}[2^{n^k}]$, *the class of languages accepted in exponential time;*
3. $\mathbf{E} = \bigcup_{k \in \mathbb{N}} \mathbf{DTIME}[2^{kn}]$, *the class of languages accepted in linear exponential time;*

As mentioned previously we can allow the Turing machine to make guesses, by defining the transition function as $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma)$, i.e., at each state the transition for a next state is not deterministic and, in fact, can be chosen between several alternatives. We say that the non-deterministic Turing machine M accepts an input string x if there is a choice of transitions that cause M to enter into an accepting state.

If no resource restrictions are considered then deterministic Turing machines decide exactly the same languages that non-deterministic Turing machines do. In other words, the models of computation are equivalent. When polynomial time constrains are considered then it is unknown if the two models are equivalent.

Consider a time constructible function t. We define the class $\mathbf{NDTIME}(t)$ similarly to $\mathbf{DTIME}(t)$ using non-deterministic Turing machines instead of using deterministic ones. Thus, the classes **NP**, **NEXP** and **NE** are the non-deterministic versions of **P**, **EXP** and **E** respectively.

The most important open problem in Computer Science is the question whether $\mathbf{P} = \mathbf{NP}$. By the way that theses classes were defined it is clear that $\mathbf{P} \subset \mathbf{NP}$ but it is unknown an efficient way (even lower than exponential time) of simulating a non-deterministic Turing machine in the deterministic model, where efficient means with a polynomial number of extra steps.

Definition 2.2.8 (Polynomial time many-to-one reduction)**.** *Let* $f : \Sigma^* \to \Sigma^*$ *be a function computable in polynomial time. The function* f reduces a language L_1 to a language L_2 *when* $x \in L_1$ *if and only if* $f(x) \in L_2$.

This type of reductions is also known as Karp reductions. There are other kinds of reductions as for example, Turing reductions[1], truth table reductions, but in this work we will be using only Karp reductions.

[1] A is Turing reducible or Cook reducible to B ($A \leq_T B$) if there is a Turing machine with oracle access to B that computes χ_A.

Definition 2.2.9 (**NP**-complete problems)**.** *A language* L *is called* **NP**-complete *if* L *is in* **NP** *and for all languages* $L' \in \mathbf{NP}$ *there is a polynomial time many-to-one reduction* f *from* L' *to* L*.*

Notice that if some polynomial time algorithm solves any **NP**-complete problem then $\mathbf{P} = \mathbf{NP}$. The first problem shown to be **NP**-complete was **SAT** by Cook [Coo71]. **SAT** is the set of boolean formulas that have at least one assignment that make the boolean formula true.

Definition 2.2.10 (**SAT**)**.** *A Boolean formula* $\phi(x_1, ..., x_n)$ *in the conjunctive normal form*[2] *of* n *variables is* satisfiable *if there is a Boolean-valued truth assignment* $x_1 = a_1, ..., x_n = a_n$ *such that* $\phi(a_1, ..., a_n)$ *is a true formula. The* **SAT** problem *is to decide whether or not a given Boolean formula has a satisfying assignment.*

Later, Karp [Kar72] proved that combinatorial problems like the Traveling Salesman problem and Vertex Cover and many other are also **NP**-complete. In [GJ79] we can find a large collection of **NP**-complete problems.

Another widely used model of computation in computational complexity is the probabilistic model. In this model, contrarily to non-deterministic Turing machines, instead of guessing the next move it is flipped a coin and then the model evolves according to the random outcome. This model is believed to be more efficient in practice when compared to its deterministic counterpart. Although, it is unknown if they are equivalent when time restrictions are considered.

Formally, we define a probabilistic Turing machine by equipping it with a special read only tape which content is the outcome of independent coin tosses of a fair coin.

Definition 2.2.11 (Probabilistic acceptance)**.** *We say that a* probabilistic Turing machine M accepts a language L *if for some* $\varepsilon > 0$ *and for all input* x *we have:*

- *if* $x \in L$, $\Pr(M \text{ accepts } x) \geq \frac{1}{2} + \varepsilon$*;*
- *if* $x \notin L$, $\Pr(M \text{ accepts } x) \leq \frac{1}{2} - \varepsilon$*;*

The class **BPP** is defined as the set of languages that are decidable by a probabilistic Turing machine running in polynomial time. It is unknown whether $\mathbf{P} = \mathbf{BPP}$ or $\mathbf{NP} = \mathbf{BPP}$. Although much effort and energy has been devoted to answer these questions, there is no significant progress in terms of collapsing or separating them.

[2] A Boolean formula is in the conjunctive normal form if it is a conjunction of clauses, where a clause is a disjunction of literals, where a literal and its complement can not appear in the same clause.

The last model that we will consider is the non-uniform model, where the Turing machine have access to an *advice* string, which is a small piece of information (normally logarithmic or polynomial in the size of the input) that the machine can use to decide the language. The term "non-uniform" is used to express the fact that for different input sizes the machine will have access to different advices. The definition of this model is due to Karp and Lipton [KL80].

Definition 2.2.12 (Non-uniform complexity classes)**.** *Let* $f : \mathbb{N} \to [0, +\infty)$ *be a function. A language* L *is in* $\mathbf{P/f}$ *if there is a Turing machine* M *and there is a sequence* $(a_n)_{n\in\mathbb{N}}$ *of advices with* $|a_n| \leq f(n)$*, such that, for all* $x \in \Sigma^n$*,* $M(a_n, x) = \chi_L(x)$ *in polynomial time. In particular, if* f *is a logarithmic (respectively polynomial) function we have the class* $\mathbf{P/log}$ *(respectively* $\mathbf{P/poly}$*).*

Notice that if f was a exponential function then any language would be $\mathbf{P/f}$ since in that case we could incorporate a table with the contents of L as advice. The most useful and commonly studied classes are $\mathbf{P/poly}$ and $\mathbf{P/log}$.

2.3 Kolmogorov complexity

Kolmogorov complexity is a different way of measuring complexity. Instead of analyzing the overall progression of the resources needed to solve larger and larger instances of a problem, it studies the resources needed for individual strings. This measure is based on the length of the description of strings.

Consider the experiment of independently flipping a fair coin several times and construct a binary string with the outcomes, by order. As it is the result of a truly random source we would say that any string produced by this method is random. However, we hardly expect to see strings like "000000000000000" or "110110110110110110" as outcomes since these strings, somehow, violate our intuition that a random string does not have regularities. Kolmogorov complexity formalizes this intuition by analyzing the existence of any type of regularities that allow a program to describe the string in a shorter way than just printing every bit of the string. Kolmogorov [Kol65], Solomonoff [Sol64] and Chaitin [Cha66] independently defined the complexity of an individual object, usually a string x, as the length of the shortest program that produces x. To avoid paradoxal questions we must choose carefully what we mean by "program" and "description".

Consider the effective enumeration of Turing machines $T_1, T_2, \cdots$ that induce an effective enumeration of partial recursive functions $\phi_1, \phi_2, \cdots$ such that T_i computes ϕ_i for

all i.

Definition 2.3.1 (Conditional Kolmogorov complexity)**.** *Let ϕ be a partial recursive function. Given any pair of strings $x, y \in \Sigma^*$, the* Kolmogorov complexity *of x conditional to y with respect to ϕ is*

$$C_\phi(x|y) = \min_p\{|p| : \phi(p, y) = x\}.$$

and $C_\phi(x|y) = \infty$ if there is no p such that $\phi(p, y) = x$.

Notice that the Kolmogorov complexity might depend on the choice of ϕ. The invariance Theorem, stated bellow, allows us to abstract from any particular partial computable function ϕ. It shows that Kolmogorov complexity is an intrinsic property of individual objects.

Theorem 2.3.2 (Invariance Theorem)**.** *There is a universal partial recursive function ϕ_0 such that, for any partial recursive function ϕ, there is a constant c_ϕ, depending only on ϕ such that, for all pairs of strings $x, y \in \Sigma^*$,*

$$C_{\phi_0}(x|y) \leq C_\phi(x|y) + c_\phi$$

Notice that the existence of a universal partial recursive function does not necessarily gives the shortest description of a string, however this invariance Theorem guaranties that no other description method can improve infinitely many times the universal description by more than a constant.

In the sequel of this thesis we fix once and for all a universal partial recursive function ϕ_0 and drop the argument writing $C(x|y)$ instead of $C_{\phi_0}(x|y)$.

The default value for y is ϵ, the empty string. When this is the case, it is usual to refer to the Kolmogorov complexity of x and denote it by $C(x)$.

In the next theorem, some of the basic properties of Kolmogorov complexity are enumerated.

Theorem 2.3.3. *The Kolmogorov complexity satisfies the following properties:*

1. *The Kolmogorov complexity is uncomputable.*
2. *There is no unbounded and non-decreasing recursive function f satisfying $C(x) \geq f(x)$ for all $x \in \Sigma^*$.*
3. *There is a constant $c > 0$, such that for all $x \in \Sigma^*$, $C(x) \leq |x| + c$.*

4. *If* f *is a partial computable function, then there is a constant* c *such that for all* $x \in \Sigma^*$, $C(f(x)) \leq C(x) + c$.

5. *There is a constant* c *such that for any string* $x \in \Sigma^*$, $C(|x|) \leq C(x) + c$.

Definition 2.3.4 (c-incompressibility)**.** *Let* c *be a fixed constant. A string* $x \in \Sigma^*$ *is called* c-incompressible (or c-algorithmically random or c-Kolmogorov random) *if* $C(x) \geq |x| - c$.

Using a simple pigeon-hole argument, it is easy to see that at least one string of length n has Kolmogorov complexity n. In fact, there are 2^n strings of length n but there are only $2^n - 1$ possible descriptions of length less than n.

A similar argument can be used to prove the following more general result:

Theorem 2.3.5. *Let* c *be a fixed constant,* y *a string and let* A *be a finite set of cardinality* m*. The number of strings* $x \in A$ *such that* $C(x|y) \geq m - c$ *is at least* $m(1 - 2^{-c})$.

The existence of such strings has been proven to be very useful, specially in computational complexity, since there are numerous results based on this fact.

2.3.1 Symmetry of information

The Kolmogorov complexity has an useful property called symmetry of information. Given a string x and a string y the amount of information that x gives about y, is the same (up to a logarithmic term) that y gives about x. To formally state this result we first need to define mutual information. The name "mutual information" came from its analogous on Shannon entropy [Sha48] and is justified by the following definition and results:

Definition 2.3.6 (Mutual information)**.** *The* mutual information *of* x *about* y *is*

$$I(x : y) = C(y) - C(y|x).$$

Since $C(y) \geq C(y|x) + O(1)$ it follows that $I(x : y) \geq O(1)$.

Let $C(x, y) = C(\langle x, y \rangle)$, where $\langle x, y \rangle$ is a representation of the pair $\langle x, y \rangle$. To describe $\langle x, y \rangle$ we can give a description of y, a description of x given y and a way of separating these two items. So, if $n = \max\{|x|, |y|\}$, then

$$C(x, y) \leq C(y) + C(x|y) + O(\log n)$$

Theorem 2.3.7 (Symmetry of information - Levin as suggested by [ZL70])**.** *For every* $x, y \in \Sigma^n$,

$$C(x, y) = C(y) + C(x|y) + O(\log n).$$

Proof. We have already seen that $C(x, y) \leq C(y) + C(x|y) + O(\log n)$. Now assume that $C(x, y) = a$. If a is known, then the set:

$$A = \{(x', y') : C(x', y') \leq a\}$$

is recursively enumerable, since we can dovetail over all programs of length at most a and insert a pair (x', y') in A if and only if one of those programs stops with (x', y') as output.

Consider now the set

$$A_x = \{y' : C(x, y') \leq a\}.$$

By definition, y is in A_x and A_x is also recursively enumerable when x and a are given. Notice that A_x has at most 2^{a+1} elements. So, to describe y we can give a, x and the index of y in A_x. Therefore,

$$C(y|x) \leq \log |A_x| + O(\log n) \tag{2.3.1}$$

Now, let k be the only number such that $2^k \leq |A_x| < 2^{k+1}$ and consider B_k, the set of all x for each there are more than 2^k strings y such that $C(x, y) \leq a$. Again, by definition, $x \in B_k$ and is recursively enumerable given k and a.

B_k has less than 2^{a-k+1} elements since A has less than 2^{a+1} elements. Thus, to describe x we only need to give a, k and the index of x in B_k. Thus,

$$C(x) \leq a - k + 1 + O(\log n) \tag{2.3.2}$$

Hence, putting together the inequalities 2.3.1 and 2.3.2 we get:

$$C(x) + C(y|x) \leq a - k + k + 1 + O(\log n) = a + O(\log n) \leq C(x, y) + O(\log n).$$

□

Corollary 2.3.8. *Given any two strings* $x, y \in \Sigma^*$,

$$I(x : y) = I(y : x) + O(\log \max\{|x|, |y|\}).$$

2.3.2 Prefix-free Kolmogorov complexity

Kolmogorov complexity as defined in the previous section does not satisfy the sub-additivity property, i.e., it does not satisfy $C(x, y) \leq C(x) + C(y) + c$, for some c that does not depend on x and y. This is not the only drawback of Kolmogorov complexity.

We can not use this definition as a universal prior probability for each binary string as Sollomonoff had conjectured in [Sol64]. So we need to consider another definition of Kolmogorov complexity. Levin [Lev74] solved this problem by considering as the formal model Turing machines in which the set of programs is a prefix-free set.

Definition 2.3.9. *A set of strings* A *is* prefix-free *if there are not two strings* x *and* y *in* A *such that* x *is a proper prefix of* y.

Theorem 2.3.10 (Kraft's Inequality [Kra49])**.** *Let* $n_1, n_2, \ldots$ *be a finite or infinite sequence of positive integer numbers. There is a prefix-free set with this sequence as lengths of its binary code words iff*

$$\sum_i 2^{-n_i} \leq 1.$$

One proof of this theorem relies on the fact that one can associate to each binary string an interval of length 2^{-n_i} which can represent a prefix-free code and vice-versa.

Similarly to what happens for regular Turing machines, there is also an effective enumeration of these Turing machines, usually called prefix-free Turing machines, and a universal prefix-free Turing machine that can simulate the behavior of any other prefix-free Turing machine. Consequently, we can define the prefix-free Kolmogorov complexity:

Definition 2.3.11 (Prefix-free Kolmogorov complexity [Lev74])**.** *Let* U *be a fixed universal Turing machine with a prefix-free domain. For any strings* $x, y \in \Sigma^*$*, the* prefix-free Kolmogorov complexity *of* x *given* y *is*

$$K(x|y) = \min_p \{|p| : U(p, y) = x\}.$$

The following theorem states some basic properties of prefix-free Kolmogorov complexity.

Theorem 2.3.12. *There is a constant* c *such that for all binary strings x and y,*

- $K(x, y) \leq K(x) + K(y) + c$;
- $K(x) \leq |x| + 2\log|x| + c$;
- $K(x) = K(x, K(x)) + c$.
- $K(x|y) \leq K(x) + c$.

With a pigeon-hole argument we can prove that there are at most 2^{n-c} strings x of length n such that $K(x) \leq n - c$. A more general result is:

Theorem 2.3.13.

- *There is a constant* c *such that, for each* n, $\max\{K(x) : |x| = n\} = n + K(n) + c$.
- *For each fixed constant* c, *the number of* $x \in \Sigma^n$ *with* $K(x) \leq n + K(n) - c$ *does not exceed* $2^{n-c+O(1)}$.

Similar to plain Kolmogorov complexity, the prefix-free Kolmogorov complexity satisfies the symmetry of information principle up to a logarithmic term.

Theorem 2.3.14 (Prefix-free symmetry of information)**.** *For all* $x, y \in \Sigma^n$,

$$K(x, y) = K(x) + K(y|x) + O(\log K(x)).$$

To avoid the logarithmic term, we can replace the conditional x by $\langle x, K(x)\rangle$ or equivalently by x^*, the first shortest program in lexicographic order that produces x. This result is attributed to Peter Gács in [GV04]. Formally,

Theorem 2.3.15. *Let* x *and* y *be two binary strings. Up to a fixed additive constant,*

$$K(x, y) = K(x) + K(y|x, K(x)) = K(x) + K(y|x^*)$$

2.3.3 Semi-measure based on Kolmogorov complexity

We can define a universal prior probability measure, based on Kolmogorov complexity. First we need the following definitions:

Definition 2.3.16 (Probability measures)**.** *Let* A *be a set and let* $\mathcal{P}(A)$ *denote the collection of all subsets of* A. *A function* $\mu : \mathcal{P}(A) \to [0, 1]$ *is called a* probability measure *over* A *if it satisfies the following conditions:*

- $\mu(\emptyset) = 0$;
- $\mu(A) = 1$;
- *If* $(A_n)_{n\in\mathbb{N}}$ *are mutual disjoint subsets of* A, *then* $\mu\left(\bigcup_{n\in\mathbb{N}} A_n\right) = \sum_{n\in\mathbb{N}} \mu(A_n)$.

The function μ *is called a semi-measure if it satisfies the first and the third items above and* $\mu(A) < 1$.

Notice that, due to Kraft's inequality, we can define a semi-measure over Σ^* by $\mu(x) = 2^{-K(x)}$, which assigns a weight to each string accordingly to its complexity. The more complex the string is to describe the less weight it has. This measure captures the intuition of the Occam's Razor principle which states that one should give more probability to simple explanations rather than complicated ones. An important result in Kolmogorov complexity is the existence of a semi-measure that multiplicatively dominates every other distribution. A semi-measure satisfying this property is called *universal*.

Theorem 2.3.17. *There is a universal enumerable discrete semi-measure which is denoted by* $\mathbf{m}$.

Proof. (Sketch) The proof consists in two steps. In the first step it is shown that the set of all enumerable discrete semi-measures can be enumerated. This is done by enumerating all enumerable functions and then effectively changing the enumerable functions to enumerable discrete semi-distributions, leaving the functions that were already discrete semi-distributions unchanged. Let $\mu_1, \mu_2, ...$ be such enumeration. The second step consists to show that $\nu(x) = \sum_{n\geq 1} \alpha(n)\mu_1(x)$ with $\sum_n \alpha(n) \leq 1$ and $\alpha(n) > 0$ multiplicatively dominates all μ_i and thus is universal. □

Theorem 2.3.18 (Coding Theorem (see [LV08]))**.** *There are constants* c_1 *and* c_2 *such that for every* x*,*

$$\mathbf{m}(x) = c_1 \cdot \sum_{p:U(p)=x} 2^{-|p|} = c_2 \cdot 2^{-K(x)}$$

A proof of this result can be found in [LV08], Chapter 4, page 273.

2.3.4 Time bounded Kolmogorov complexity

The notion of Kolmogorov complexity as presented in the previous sections does not take into account the computational effort that is necessary to produce the string from one of its shortest descriptions. In this section, we consider resource-bounded Kolmogorov complexity. Imposing restrictions on the time or space that the Turing machine is allowed to use, makes the Kolmogorov complexity a computable function. In this thesis, we will focus mainly on time bounded computations but space-bounded Kolmogorov complexity can be defined analogously.

Definition 2.3.19 (Time bounded Kolmogorov complexity)**.** *Let* U *be a fixed universal prefix-free Turing machine and* t *a time constructible function such that* $t(n) > n$*. Given*

two strings x *and* y*, the* t-time bounded Kolmogorov complexity *of* x *given* y *is:*

$$K^t(x|y) = \min_p\{|p| : U(p, y) = x \text{ in } t(|x| + |y|) \text{ steps}\}$$

Imposing time bounds to prefix-free Kolmogorov complexity changes very little its basic properties. In particular, $K^t(x|y) \leq |x|+2\log(|x|)+O(1)$. The time bounded version of incompressibility Theorem 2.3.13 also holds. In fact, $\lim_{t\to\infty} K^t(x) = K(x)$. However, it is unknown if time bounded Kolmogorov complexity differs from its unbounded version regarding symmetry of information, since it is not known if polynomial time bounded symmetry of information holds. This is an important open question with connections to complexity theory. Longpré and Mocas [LM93] and Longpré and Watanabe [LW95] have examined the conjecture of symmetry of information for polynomial time bounded Kolmogorov complexity. In particular, they proved that the conjecture is false if certain kinds of one-way functions exist. Intuitively, a one-way function is a function that is easy to compute, but its inverse is hard to compute. In [LR05], the authors explore the conjecture of polynomial time symmetry of information for other types of time bounded Kolmogorov related measures.

By analogy, we can define the t-time bounded universal semi-measure by $\mathbf{m}^t(x) = 2^{-K^t(x)}$. The function $\mathbf{m}^t(x)$ is computable in time $t(n) \cdot 2^{n+1} + O(1)$ by simulating all programs of size up to n for $t(n)$ steps. If p is a polynomial, $\mathbf{m}^p = 2^{-K^p(x)}$ sits somewhere between the polynomial time computable and polynomial time samplable distributions, since it dominates every polynomial time computable distribution and is dominated by a polynomial time samplable distribution (see [AFV03]). Later, Antunes and Fortnow in [AF09b] proved that if $\mathbf{E} = \mathbf{DTIME}\left(2^{O(n)}\right)$ does not have circuits of size $2^{o(n)}$ with Σ_2^p gates, i.e., if $\mathbf{E}$ has difficult functions for subexponential space, then for every polynomial time samplable distribution γ, there is a polynomial p such that $\mathbf{m}^p(x) \geq \gamma(x)/|x|^{O(1)}$. The next result states that $\mathbf{m}^t$ is universal among the distributions that have computable cumulative mass functions, i.e., for a probability distribution P, the function $P^*(x) = \sum_{y\leq x} P(y)$ is computable.

Theorem 2.3.20 ([LV08])**.** *The distribution* $\mathbf{m}^{t'}$ *dominates any distribution which cumulative mass function is* t*-time computable, where* $t'(n) = n \cdot t(n) \cdot \log(n \cdot t(n))$.

2.3.5 Computational depth

Kolmogorov random strings are the strings that have as their shortest description the string itself. Thus, random strings are objects containing lots of information. However,

this information may be not very useful from a computational point of view, since, with high probability, one can get another string as useful as the first one by flipping a fair coin. On the other hand, strings with lots of regularities can be highly compressed and thus, they have simple laws of construction. This simplicity may be instantly clear or very demanding in terms of computational resources. For example, if the object is the collection of results in a scientific book like [LV08], it has low Kolmogorov complexity since we can derive the proofs of all theorems using only a couple of initial definitions and inference rules. So, if we need to send the book to someone, we can use a small number of bits when compared with the size of the entire book, but the receiver will spend a long time reconstructing the proofs and the full book. On the other hand, we can send the entire book. In this case, the receiver could read the book and extract all the information easily. Thus, in both cases, we have the same information from Kolmogorov complexity point of view, but the difference between them is the tradeoff of computational difficulty and the number of bits communicated.

Informally, computational depth is a measure for the amount of "nonrandom" or "useful" information contained in a string. Notice that a computational deep string should have lots of regularities, i.e., it has low Kolmogorov complexity, but there is no efficient procedure to determine those regularities. In fact, in the previous example, from this point of view, the Li and Vitanyi's book [LV08] is a deep object.

A first attempt to formalize the concept of computational depth is due to Bennett. In [Ben88] he defined the s-significant logical depth of an object x as the time required by the reference universal Turing machine to generate x by a program that is no more than s bits longer than the shortest descriptions of x. Formally,

Definition 2.3.21 (Logical depth [Ben88])**.** *The* Logical Depth *of a string* x *at a significance level* s *is:*

$$\mathrm{ldepth}(x) = \min_{t} \left\{ t(|x|) : \sum_{p:U^t(p)=x} 2^{-|p|} > 2^{-s} \sum_{p:U(p)=x} 2^{-|p|} \right\}.$$

Later, Antunes, Fortnow, van Melkebeek and Vinodchandran [AFvMV06], to simplify this notion defined *computational depth* as a measure of nonrandom information in a string based on the difference between various kinds of time bounded Kolmogorov complexity and their unbounded counterparts. They have defined three types of computational depths:

1. *Basic computational depth*: measuring the gap between time bounded Kolmogorov complexity and traditional unrestricted Kolmogorov complexity.

This notion captures the intuitive concept of computational depth, much better than the technical notion of logical depth does. Note that x has large logical depth if it has short programs that require considerable computation time. It seems natural that computational depth has a similar property.

2. *Sublinear time computational depth*: gives an alternative way of defining shallow sets by analyzing its characteristic sequence. It also gives a way of proving that if **NP** is sparse or is reducible to a random set, then **NP** has polynomial-size circuits and the polynomial time hierarchy collapses.

3. *Distinguishing computational depth*: analyses the difference between polynomial time bounded distinguishing complexity, $CD^t(x)$, defined by the length of shortest program p running in time t such that $U^t(p, z) = 1$ if and only if $z = x$, and polynomial time bounded Kolmogorov complexity. Fortnow and Kummer [FK96] show that under some computational assumptions there are strings with high distinguishing computational depth.

In this thesis we are interested in *basic computational depth* and *sublinear computational depth.*

Definition 2.3.22 (Basic computational depth [AFvMV06])**.** *Let* t *be a constructible time bound. For any string* $x \in \Sigma^*$*, the* basic computational depth *is:*

$$\text{depth}^t(x) = K^t(x) - K(x).$$

Notice that this notion measures the effort that is necessary to reconstruct a string from its shortest description.

A computational deep string is not easy to identify, but can be constructed by diagonalization in time larger than 2^t for depth t. The next theorem proves that there are an exponential number of strings with large basic computational depth.

Theorem 2.3.23. *There is a constant* c *such that for any* $0 < \varepsilon < 1$*, there are at least* $2^{\varepsilon n}$ *strings* x *of length* n *satisfying:*

$$\text{depth}^{2^n}(x) \geq (1 - \varepsilon)n - c \log n.$$

Proof. Consider the set:

$$A = \{x \in \Sigma^n : \exists p \in \Sigma^{n-2}, U(p) = x \text{ in at most } 2^n \text{ steps}\}.$$

By the incompressible Theorem 2.3.13, $|A| < 2^{n-1}$. So if $B = \Sigma^n - A$, $|B| > 2^{n-1}$ and for any $0 < \varepsilon < 1$, there are more than $2^{\varepsilon n}$ strings in B. Let D be the lexicographically first $2^{\varepsilon n}$ strings in B. Since D is computable and any $x \in D$ can be specified by εn bits, we have that for every $x \in D$, $K(x) \leq \varepsilon n + O(\log n)$. We also have that for every $x \in D$, $K^{2^n}(x) \geq n-1$ since every program p of size at most $n-2$ such that $U(p)$ outputs x must run for at least 2^n steps. It follows that for any $x \in D$, $\text{depth}^{2^n}(x) \geq (1-\varepsilon)n - c \log n$ for some constant c. □

In order to define shallow set by analyzing the characteristic sequences we need to define K^t for sublinear time bound t, i.e., when $t(n) < n$. This is done by allowing the universal Turing machine U access to part of the description r of the string x and requiring only that each bit of x be generated in the allotted time.

Definition 2.3.24 (Sublinear time bounded Kolmogorov complexity)**.** *Let* t *be a time bound satisfying* $t(n) \leq n$ *and* x *a string. Define* sublinear time bounded Kolmogorov complexity *by*

$$K^t(x) = \min_{p,r} \{|p| + |r| : U^r(p, i) \text{ outputs } x_i \text{ in } t(|x|) \text{ steps for all } 1 \leq i \leq |x|\}.$$

Notice that this definition with $t(n) \geq n$ is essentially equivalent to the definition 2.3.19.

Definition 2.3.25 (Shallow strings)**.** *Let* k *be a fixed constant. A string* x *is* k-shallow *if* $\text{depth}^{\log^k}(x) \leq \log^k(|x|)$.

Definition 2.3.26 (Shallow sets)**.** *A set* A *is* shallow *if there is a constant* k *such that almost every initial segment of the characteristic sequence of* A*, is* k*-shallow.*

Chapter 3

Information measures for infinite sequences

The contents of this chapter are based on the following publications:

[AS08] L. Antunes and A. Souto, Sophisticated infinite sequences*, in Proceedings of Computability in Europe 2008, Athens, Grece, 2008;*

[AS10] L. Antunes and A. Souto, Information measures for infinite sequences*, in Theoretical Computer Science, Volume: 41, Issues: 26-28, pages:2602-2611, Elsevier;*

We explore the notion of sophistication defined in [AF09a] studying it for infinite sequences. A first definition of sophistication is due to Koppel ([Kop87, Kop95]). We show that our notion is properly defined for every sequence, and prove that the set of sequences with sophistication equal to zero has measure 1 *and that the set of sophisticated sequences is dense. We also prove that sophistication and computational depth of sequences are distinct complexity measures and that deep sequences are dense.*

3.1 Motivation

Random strings convey maximal information since they have almost maximum Kolmogorov complexity. However, it is very unlikely that these strings have "useful" or meaningful information, as they are usually considered to be noise. Thus, how do we formalize the notion of meaningful information?

We can do it in one of the two following ways: measuring the amount of planing necessary to construct the object (static resources) or measuring the computational effort (dynamic resources, usually time), required to produce the object.

As we have already discussed in the introduction, the former approach is based on the Kolmogorov structure function which divides the smallest program of a string in two parts: one part accounting for the useful regularity which can be exploited to describe the string and another accounting for the remaining accidental information present in the string. In [Kop87, Kop95, KA91], Koppel expressed the useful information as a recursive function and called the resulting measure *sophistication*. As a consequence, the regularity is just the length of a total program p that, together with the accidental information, i.e. information that is consider not to have structure, produces the string or sequence.

However, as Koppel observed, not all infinite sequences are describable and thus, the notion of sophistication, is not properly defined. We redefine sophistication for infinite sequence based on [AF09a], where the authors revisited the notion of sophistication for finite strings, introducing a new definition, and proving the existence of strings with maximum sophistication.

The latter approach was introduced in [Ben88] who called *logical depth* to the effort required to produce the object. Thus, an object is logically deep if a lot of time is needed to recover it from any of its shortest descriptions. In the sequel, Antunes *et al.* [AFvMV06] introduced the notion of computational depth for finite strings, as the difference between the time bounded and unbounded Kolmogorov complexities.

Concerning Kolmogorov's question (see [V'y99]) on the existence of "absolutely non-random" or highly sophisticated objects Gács *et al.* [GTV01] and Antunes and Fortnow [AF09a] independently, proved that the answer is affirmative for strings. We address Kolmogorov's question for sequences. We start by redefining sophistication for sequences, introducing the lower and upper sophistication as the $\liminf$ and $\limsup$, respectively, of the ratio between the sophistication of the initial segments (as defined in [AF09a]) and the length of that initial segments. Notice that these two notions of sophistication are always well defined, solving one of the problems of Koppel's definition. Using these new definitions, we prove that the set of sequences with lower sophistication equal to $\mathsf{0}$ has measure $\mathsf{1}$ and the set of sequences with upper sophistication equal to $\mathsf{1}$ is dense. So, the answer to Kolmogorov's question, regarding infinite sequences, is affirmative if we use upper sophistication and probably negative if we use the lower sophistication. We also prove that the set of deep infinite sequences is dense. The study of these measures for infinite sequences may be useful as a complexity measure to detect attack behaviors in

network traffic as it is usually considered to be an infinite source of information.

Koppel claimed that sophistication and logical depth are equivalent information measures. For the finite case, Antunes and Fortnow [AF09a], gave an example where the equivalence is not valid. In this work, we give two examples of infinite sequences for which sophistication differs from a variant of computational depth.

3.2 Preliminaries

We present some specific definitions and results that are necessary to understand the sequel of this chapter.

3.2.1 Sophistication

The Kolmogorov structure function divides the smallest program for an object x in two parts: one part accounting for the useful regularities to compress the string and another for the remaining accidental information present in the object. The former is the structure of x, i.e, it essentially describes the set of all objects that share the same structure with x, and the latter can be considered as the index of x in that set. To formalize this notion, Koppel [Kop87] used total functions to represent the useful information. He defined sophistication based on (monotonic) process complexity.

Definition 3.2.1. *A* description *of a string* x *is a pair* (p, d) *such that* p *is a self-delimiting total program, and* x *is an initial segment of* $U(p, d)$.

Koppel also defined the *complexity* of x by:

$$H(x) = \min\{|p| + |d| : (p, d) \text{ is a description of } x\}.$$

Definition 3.2.2. *A description* (p, d) *of a string* x *is* c*-minimal if* $|p| + |d| \leq H(x) + c$.

Definition 3.2.3. *A program* p *is a* c*-minimal program for* x *if:*

1. *for some* d*, the pair* (p, d) *is a* c*-minimal description of* x.
2. *for any* c*-minimal description* (p', d') *of* x*, we have* $|p| \leq |p'|$.

Definition 3.2.4. *The* c*-sophistication of a string* x*, denoted by* $\mathrm{soph}_c(x)$*, is the length of a* c*-minimal program for* x.

Bringing together the two previous definitions we obtain the following definition for c-sophistication of a string $x \in \Sigma^n$:

$$\mathrm{soph}_c(x) = \min_p \left\{ |p| : \begin{array}{l} \text{exists } d \text{ such that } (p,d) \text{ is a description} \\ \text{of } x \text{ and } |p| + |d| \leq H(x) + c \end{array} \right\}.$$

Koppel used a similar approach for sequences. First, he defined a notion of compression and minimal description and, then he defined sophistication.

Definition 3.2.5. *A program p is called a c-compression program for a sequence α if:*

1. *For all n there is d_n such that (p, d_n) is a c-minimal description of $\alpha_{[1:n]}$.*
2. *$d_{n-1} \leq d_n$, i.e., d_{n-1} is an initial segment of d_n.*

Definition 3.2.6. *The sophistication of a sequence α is*

$$\mathrm{soph}_c(\alpha) = \min_p \{|p| : p \text{ is a } c\text{-compression program for } \alpha\}.$$

If such p does not exist then $\mathrm{soph}_c(\alpha) = \infty$.

A sequence is describable if it has a compression program. Koppel in [Kop95] remarked that not every sequence is describable and thus $\mathrm{soph}_c(\alpha)$ is not properly defined. For example, if for all c, $\limsup_n \mathrm{soph}_c(\alpha_{[1:n]}) = \infty$, then α is not describable. In order to avoid this problem, Koppel defined a weaker version of sophistication based on "weak" compression programs for α, i.e., program for which the data considered for each n is not necessarily a prefix of the next one. Antunes and Fortnow [AF09a] revisited the notion of sophistication and, using Kolmogorov complexity, adapted Koppel's definition for finite strings.

Definition 3.2.7. *Let c be a constant, x a string of length n and U the universal reference Turing machine. The c-sophistication of x is*

$$\mathrm{soph}_c(x) = \min_p \left\{ |p| : \begin{array}{l} p \text{ is total and there is a string } d \text{ such that} \\ U(p,d) = x \text{ and } |p| + |d| \leq K(x) + c \end{array} \right\}.$$

By definition, it is always true that $\mathrm{soph}_c(x) \leq K(x) + c$. It is not known if sophistication is a robust measure. Indeed it is unknown if slight variations on the parameter c, especially when c is small, largely affects the value of sophistication.

In the sequel, we will need the following generalization of Theorem 2 of [AF09a], on the existence of highly sophisticated finite strings.

Theorem 3.2.8. *Let* x *be a string of length* $\log n$ *and* $c > 0$ *a constant. There is* $y \in \Sigma^{n-\log n}$ *such that* $\mathrm{soph}_c(xy) \geq n - 10\log n - c$.

Proof. For all programs p such that $|p| \leq n - 10\log n - c$ define

$$r_p = \begin{cases} 0 \text{ if } \exists d : |d| < n - |p| - c \text{ such that } U(p,d) \text{ diverges} \\ \max\limits_{d:|d|<n-4\log n-|p|-c} \text{running time of } U(p,d) \end{cases}$$

Consider $S = \max r_p$. Given n and p that maximizes r_p we can compute S. Consider the following set

$$V = \left\{ xy \in \Sigma^n : \begin{array}{l} \text{there exists } p, d \text{ satisfying } |p| \leq n - 10\log n - c, \\ |d| \leq n - 4\log n - |p| - c \text{ such that } U^S(p,d) = xy \end{array} \right\}.$$

and $\bar{V} = \{xy \in \Sigma^n : xy \notin V\}$. Notice that if $xy \in V$ then $K(y) \leq K(xy) + K(x) \leq |p| + |d| + K(x) \leq |p| + n - 4\log n - |p| - c + K(x) \leq n - 4\log n - c + 3\log n = n - \log n - c$. Then, using the incompressible Theorem 2.3.13, there must exist at least one string $y \in \Sigma^{n-\log n}$ such that $xy \notin V$ and thus $\bar{V} \neq \emptyset$. Let z be the first string in lexicographic order in $\bar{V}$. Since given n, x and p that maximizes r_p we can compute V, then we conclude that:

$$\begin{aligned} K(z) &\leq K(x) + K(p) + K(n) + K(V|n,p) \\ &\leq 3\log n + n - 10\log n - c + 3\log n \\ &= n - 4\log n - c \end{aligned}$$

Assume that $\mathrm{soph}_c(z)$ is small, i.e., $\mathrm{soph}_c(z) \leq n - 10\log n - c$. Then, by definition

$$\exists p^*, d^* : p^* \text{ is total}, |p^*| \leq n - 10\log n - c, |p^*| + |d^*| \leq K(z) + c$$

but then we have $|d^*| \leq K(z) + c - |p^*| \leq n - 4\log n - c - |p^*|$ and thus $U(p^*, d^*)$ stops and, by construction of S, it stops in time $\leq S$, i.e., $z \in V$. But, by construction, $z \notin V$, which is a contradiction. So $\mathrm{soph}_c(z) > n - 10\log n - c$. □

3.2.2 Hausdorff and Packing dimension

In mathematics, the notion of measure is largely used. Intuitively, a measure defined over a set is a systematic way to assign a number to each suitable subset of a given set, which mean to be interpreted as the size of the subset. In this sense, a measure is a generalization of the concepts of length, area and volume. Hausdorff [Hau19] augmented Lebesgue measure theory with a concept of dimension. This measure assigns to every subset X of a given metric space a real number $\dim(X)$, called the *Hausdorff dimension*

of X. Lutz [Lut00a, Lut00b] proved there is a gale characterization of Hausdorff dimension. This characterization gives an exact relationship between Hausdorff dimension of a set X consisting of infinite binary sequences, and growth rates achievable by martingales betting on the sequences in X. This gale characterization of Hausdorff dimension was a breakthrough as it enabled the definition of *effective versions* of Hausdorff dimension by imposing various computational and complexity constraints on the gales.

Definition 3.2.9. *An s-supergale is a function $d : \Sigma^\infty \to [0, \infty[$ that satisfies the following condition:*

$$d(x) \geq \frac{d(0\alpha) + d(1\alpha)}{2^s}.$$

An s-supergale is called an s-gale if the equality holds. A gale is called a martingale if $s = 1$.

Definition 3.2.10. *Let α be a sequence and d an s-supergale. We say that d succeeds on α if:*

$$\limsup_n d(\alpha_{[1:n]}) = \infty.$$

We define S_d as the set of sequences where d succeeds.

Definition 3.2.11. *Let X be a set of sequences.*

- *$\mathcal{G}(X)$ is the set of all $s \in [0, \infty[$ such that there is an s-gale d for which $X \subset S_d$.*
- *$\hat{\mathcal{G}}(X)$ is the set of all $s \in [0, \infty[$ such that there is an s-supergale d for which $X \subset S_d$.*
- *$\hat{\mathcal{G}}_{\mathrm{constr}}(X)$ is the set of all $s \in [0, \infty[$ such that there is a lower semi-computable s-supergale d for which $X \subset S_d$.*
- *The Hausdorff dimension of X is $\dim_H(X) = \inf \mathcal{G}(X) = \inf \hat{\mathcal{G}}(X)$.*
- *The constructive dimension of X is $\mathrm{cdim}(X) = \inf \hat{\mathcal{G}}_{\mathrm{constr}}(X)$.*
- *The constructive dimension of a sequence α is $\dim(A) = \mathrm{cdim}(\{\alpha\})$.*

Intuitively, a martingale d is a strategy for betting on the successive bits of a sequence α.

In [May02], it is shown that constructive Hausdorff dimension can be fully characterized in terms of Kolmogorov complexity.

Theorem 3.2.12 (Constructive Hausdorff dimension)**.** *For every sequence α,*

$$\dim(\alpha) = \liminf_{n\to\infty} \frac{K(\alpha_{[1:n]})}{n}.$$

The proof of this result can be verified conjugating [May02] and [Lut00b].

Proof. The "$\geq$" inequality follows directly from the fact that the dimension can be used to describe the sequence.

To prove the inequality "$\leq$", let α be a sequence and s and s' two rational numbers such that $s > s' > \liminf_{n\in\infty} \frac{K(\alpha_{[1:n]})}{n}$. The set:

$$B = \{x \in \Sigma^* : K(x) \leq s'|x|\}$$

is recursively enumerable and for all n, $B^{=n}$ has at most $2^{s'n-K(n)+O(1)}$ elements. Using Kraft's inequality we know that

$$d(x) = 2^{(s-s')|x|}\left(\sum_{w:xw\in B} 2^{-s'|w|} + \sum_{w:w\in B, w\leq x} 2^{(s'-1)}(|x|-|w|)\right)$$

is a lower semi-computable s-gale and by construction, if $x \in B$, $d(w) \geq 2^{(s-s')|x|}$. By assumption, there are infinitely many n for which $\alpha_{[1:n]} \in B$ and thus $\alpha \in S_d$ which implies that $\dim(\alpha) \leq 2^s$. Since this result is true for every $s > \liminf_{n\in\infty} \frac{K(\alpha_{[1:n]})}{n}$ the inequality follows. □

Packing dimension was introduced independently by Tricot [Tri82] and Sullivan [Sul84]. Later, Athreya *et al.* [AHLM07] proved how to characterize packing dimension in terms of gales, a dual of the gale characterization of the Hausdorff dimension. By imposing computational constrains on the gales they obtained a variety of *effective strong dimensions* which are exactly duals of the effective Hausdorff dimension. In particular, the following characterization was proved.

Theorem 3.2.13 (Constructive packing dimension)**.** *For every sequence α,*

$$\mathrm{Dim}(\alpha) = \limsup_{n\to\infty} \frac{K(\alpha_{[1:n]})}{n}.$$

The proof of this result can be found in [AHLM07] and is dual of the previous one.

Theorems 3.2.12 and 3.2.13 are not valid for the plain version of Kolmogorov complexity.

3.2.3 Topological results

To formally present the results of the next sections, we use the standard metric in the Cantor space Σ^∞ and apply the well known Baire's result for complete metric spaces.

Definition 3.2.14. *In the Cantor set* Σ^∞*, given* $\alpha, \beta \in \Sigma^\infty$*, the* standard metric *is:*

$$d(\alpha, \beta) = \max_i\{2^{-i} : \alpha_i \neq \beta_i\}$$

It is well known that (Σ^∞, d) is a complete metric space. Notice that, the less the distance between α and β, the bigger the initial segment common to α and β.

Definition 3.2.15. *Let* (X, d) *be a metric space. We say that a set* A *is* open *if, for every* $x \in A$*, there is a* $\varepsilon > 0$ *such that the ball* $B(x, \varepsilon) = \{y \in X : d(x, y) < \varepsilon\}$ *is contained in* A*. The complement of an open set is called a* closed set*.*

A set D *is called* dense *if for all* $x \in X$ *and every* $\varepsilon > 0$ *there is* $y \in D$ *such that* $d(x, y) < \varepsilon$*.*

Theorem 3.2.16 (Baire's Theorem)**.** *Let* (X, d) *be a complete metric space and let* $(A_n)_{n\in\mathbb{N}}$ *be a sequence of open dense subsets of* X*. Then* $\bigcap_{n\in\mathbb{N}} A_n$ *is dense.*

3.3 The existence of highly sophisticated sequences

We propose a new and simpler definition of sophistication of infinite sequences. The definitions are based on the limits of the sophistication of the initial segments and formalize the idea that if we analyze the sequence considering larger and larger initial segments, then we can describe better the emerging structure of the sequence. We prove the existence of highly sophisticated sequences, a connection with constructive Hausdorff dimension and constructive packing dimension.

Keeping a similar flavor to the Hausdorff and Packing dimension characterizations, we propose the following definition of sophistication for infinite sequences.

Definition 3.3.1. *The* lower sophistication *of a sequence* $\alpha \in \Sigma^\infty$ *is defined as:*

$$\underline{\mathrm{soph}}_c(\alpha) = \liminf_n \frac{\mathrm{soph}_c(\alpha_{[1:n]})}{n}$$

and the upper sophistication *as:*

$$\overline{\mathrm{soph}}_c(\alpha) = \limsup_n \frac{\mathrm{soph}_c(\alpha_{[1:n]})}{n}$$

Notice that the lim inf and lim sup of real numbers are always defined. Hence, the lower and the upper sophistication of a sequence are well defined, solving one of the problems of Koppel's definition. We now prove some properties of the new measure and establish a connection with constructive Hausdorff dimension and with constructive packing dimension.

Proposition 3.3.2. *For all sequence* α *and constant* $c > 0$, $\underline{\mathrm{soph}}_c(\alpha) \leq \dim(\alpha)$ *and* $\overline{\mathrm{soph}}_c(\alpha) \leq \mathrm{Dim}(\alpha)$.

Proof.

$$\begin{aligned}
\underline{\mathrm{soph}}_c(\alpha) &= \liminf_n \frac{\mathrm{soph}_c(\alpha_{[1:n]})}{n} \\
&\leq \liminf_n \frac{K(\alpha_{[1:n]}) + c}{n} \\
&\leq \liminf_n \frac{K(\alpha_{[1:n]})}{n} + \liminf_n \frac{c}{n} \\
&= \liminf_n \frac{K(\alpha_{[1:n]})}{n} \\
&= \dim(\alpha)
\end{aligned}$$

The proof that $\overline{\mathrm{soph}}_c(\alpha) \leq \mathrm{Dim}(\alpha)$ is similar. □

A sharper result for the lower sophistication is presented next. It proves the existence of sequences for which the lower sophistication is strictly smaller than the constructive Hausdorff dimension.

Proposition 3.3.3. *For any sufficiently large constant* c*, there are sequences* α *such that* $\underline{\mathrm{soph}}_c(\alpha) = 0$ *and* $\dim(\alpha) = 1$.

Proof. The idea of the proof is to use a sequence with high Kolmogorov complexity. Chaitin [Cha66] and Martin-Löf [ML71] observed that there are α such that from some n_0 onwards $K(\alpha_{[1:n]}) \geq n - \log n - \log\log n$. In fact, almost all sequences α have this property, since $\sum_{n\in\mathbb{N}} 2^{-\log n - \log\log n}$ converges. Thus,

$$\begin{aligned}
\dim(\alpha) &= \liminf_n \frac{K(\alpha_{[1:n]})}{n} \\
&\geq \liminf_n \frac{n - \log n - \log\log n}{n} \\
&= 1
\end{aligned}$$

On the other hand, as proved in [YDD04], the following set has measure 1:

$$A = \{\alpha \in \Sigma^\infty : \text{for infinitely many } n,\ K(\alpha_{[1:n]}) = n - c\}$$

where c is a fixed constant. So, the set

$$B = \{\alpha \in \Sigma^\infty : \alpha \in A \text{ and } \mathbf{dim}(\alpha) = 1\}$$

has also measure 1. But, by definition of A, for any $\alpha \in B$, there are infinitely many n such that $\mathbf{soph}_c(\alpha_{[1:n]}) \leq O(1)$, since the program p that prints $\alpha_{[1:n]}$ when $\alpha_{[1:n]}$ is given

as data satisfies $|p| + |\alpha_{[1:n]}| \leq |p| + n = c + K(\alpha_{[1:n]})$, where c is the constant describing `print` on the universal Turing machine. So,

$$\begin{aligned}\underline{\text{soph}}_c(\alpha) &= \liminf_n \frac{\text{soph}_c(\alpha_{[1:n]})}{n} \\ &\leq \liminf_n \frac{O(1)}{n} \\ &= 0.\end{aligned}$$

□

Notice that the constant c of the previous proposition only needs to be larger than the description of the program `print` for the universal Turing machine. From the proof of previous Proposition we can conclude that:

Theorem 3.3.4. *For some constant $c > 0$, the set of sequences α such that $\underline{\text{soph}}_c(\alpha) = 0$ has measure 1.*

We now show that the set of sequences with upper sophistication equal to 1 is dense. For each natural number n_0 consider the following set:

$$V_{n_0} = \{\alpha \in \Sigma^\infty : (\forall n \geq n_0)\ \text{soph}_c(\alpha_{[1:n]}) \leq n - 10\log n - c\}$$

where c is a fixed constant. V_{n_0} is the set of sequences that from its n_0^{th} bit their initial segments are not highly sophisticated. Then we have:

1. $V_i \subset V_{i+1}$.

   If $\alpha \in V_i$ then for all $n \geq i$, $\text{soph}_c(\alpha_{[1:n]}) \leq n - 10\log n - c$. In particular, for all $n \geq i + 1$, we have that $\text{soph}_c(\alpha_{[1:n]}) \leq n - 10\log n - c$. So, $\alpha \in V_{i+1}$.

2. V_i is non empty.

   For example, the sequence such that all bits are equal to 0 has low sophistication since it has low Kolmogorov complexity.

3. For all sufficiently large n_0, $V_{n_0} \neq \Sigma^\infty$.

   Considering $y = \epsilon$ in Theorem 3.2.8, it follows that there is $x \in \Sigma^n$ that satisfies $\text{soph}_c(x) \geq n - 10\log n - c$. So, the sequence $\alpha = x000\ldots$ satisfies $\text{soph}_c(\alpha_{[1:n]}) \geq n - 10\log n - c$ and thus $\alpha \notin V_{n_0}$.

4. All sets V_{n_0} are closed subsets of Σ^∞.

   To prove this fact we show that $\Sigma^\infty - V_{n_0}$ are open subsets of (Σ^∞, d).

   If $\alpha \in \Sigma^\infty - V_{n_0}$ then there is n such that $\text{soph}_c(\alpha_{[1:n]}) \geq n - 10\log n - c$. Set $\varepsilon = 2^{n-1}$. Then, if $d(\alpha, \beta) < \varepsilon$ it implies that for all $i \leq n$, $\alpha_i = \beta_i$. So, $\text{soph}_c(\beta_{[1:n]}) = \text{soph}_c(\alpha_{[1:n]}) \geq n - 10\log n - c$, which proves that $\beta \in \Sigma^\infty - V_{n_0}$.

So if we prove that $\Sigma^\infty - V_{n_0}$ are dense then we prove that the set of all highly sophisticated sequences is dense in Σ^∞ since

$$\bigcap_{n_0 \in \mathbb{N}} \Sigma^\infty - V_{n_0} = \Sigma^\infty - \bigcup_{n_0 \in \mathbb{N}} V_{n_0}.$$

Notice that if $\alpha \in \Sigma^\infty - \bigcup_{n_0 \in \mathbb{N}} V_{n_0}$ then α satisfies

$$\overline{\text{soph}}_c(\alpha) \geq \lim_n \frac{n - 10\log n - c}{n} = 1.$$

To prove that each $\Sigma^\infty - V_{n_0}$ is dense it is sufficient to show that given $\varepsilon > 0$ and $\alpha \in V_{n_0}$ there is a sequence $\beta \in \Sigma^\infty - V_{n_0}$ such that $d(\alpha, \beta) < \varepsilon$.

Intuitively, this fact is true since we can consider the first bits of α (to ensure that $d(\alpha, \beta) < \varepsilon$) and construct a sophisticated string with that prefix of a reasonable size.

Proposition 3.3.5. *Each set $\Sigma^\infty - V_{n_0}$ is dense.*

Proof. Let α be an element in V_{n_0} and $\varepsilon > 0$ a real number. We construct β as follows:

Let i_0 be the index such that $2^{-i_0} \leq \varepsilon/2$. Set $\beta_i = \alpha_i$ for all $i \leq i_0$. With this condition we guarantee that for all $\omega \in \Sigma^\infty$, $d(\alpha, \beta_{[1:i_0]}\omega) < \varepsilon$.

Considering $x = \beta_{[1:i_0]}$ in Theorem 3.2.8 of page 43, it follows that there is $y \in \Sigma^{2^{i_0} - i_0}$ that satisfies $\text{soph}_c(\beta_{[1:i_0]}y) \geq |xy| - 10|x| - c$. Then the sequence $\beta = \beta_{[1:i_0]}y000\ldots$ satisfies $\text{soph}_c(\beta_{[1:2^{i_0}]}) \geq 2^{i_0} - 10\log 2^{i_0} - c$. So, $\beta \in \Sigma^\infty - V_{n_0}$. □

Thus, using Theorem 3.2.16 and Proposition 3.3.5 we have:

Theorem 3.3.6. *For some $c > 0$, the set of sequences α such that $\overline{\text{soph}}_c(\alpha) = 1$ is dense.*

3.4 Computational depth for sequences

There are several recent results about computational depth for strings, see [AF09b], [AFPS07], [AMSV09], [Sou10]. In this section, we study the density of the set of deep sequences and prove that, although it has measure 0, it is dense.

Definition 3.4.1. *The infinite series* $\sum 2^{-f(n)}$ *is recursively convergent if there is a recursive sequence* $(n_i)_{i\in\mathbb{N}}$ *such that* $\sum_{n=n_m}^{\infty} 2^{-f(n)} \leq 2^{-m}$ *for all* m.

Theorem 3.4.2 (Theorem 2.5.4 in [LV08])**.** *Let* $f(n)$ *be a recursive function such that* $\sum 2^{-f(n)}$ *is recursively convergent. If an infinite binary sequence* ω *is Martin-Löf random, then from some* n *onward,* $K(\omega_{[1:n]}|n) \geq n - f(n)$.

The idea of the proof is to show that the sets of the form $[0.\omega_{[1:n]}, 0.\omega_{[1:n]+2^{-n}})$ are a sequential Marti-Löf test.

Let t be any polynomial. From the last theorem, we can show that the set of sequences ω such that $K^t(\omega_{[1:n]}|n) \geq n - c\log n$ for some n onwards has also measure 1.

Theorem 3.4.3. *Let* t *be any fixed polynomial. The set*

$$A = \{\omega \in \Sigma^\infty : \textit{exists } n_0 \textit{ such that for all } n \geq n_0, K^t(\omega_{[1:n]}|n) \geq n - c\log n\}$$

where $c \geq 2$*, has measure* 1.

Proof. Since all random sequences satisfies $K^t(\omega_{[1:n]}|n) \geq n - c\log n$ for some n_0 onwards, then A contains the set of random sequences by Theorem 3.4.2. Thus, A has measure 1. □

What can we say about the complement of this set? We know that this set has measure 0, but what about its density? Adapting the arguments of Theorem 3.3.6, we show that the set of sequences having high depth for infinitely many initial segments is dense.

Theorem 3.4.4. *Let* t *be any polynomial. The set of sequences* ω *such that for infinitely many* n*,* $K^t(\omega_{[1:n]}|n) \leq n - c\log n$ *for some* $c \geq 2$ *is dense.*

Proof. For each natural number n_0, consider the sets:

$$A_{n_0} = \{\omega \in \Sigma^\infty : \forall n \geq n_0, K^t(\omega_{[1:n]}|n) > n - c\log n\}$$

It follows from the results above that $\bigcup_{n_0} A_{n_0}$ has measure 1 since it contains the set of Martin-Löf random sequences.

1. $\Sigma^\infty - A_{n_0}$ are open sets.

   Let α be a sequence in $\Sigma^\infty - A_{n_0}$. Then, by the definition of A_{n_0}, there is a $n \geq n_0$ such that $K^t(\alpha_{[1:n]}|n) \leq n - c\log n$. Set $\varepsilon = 2^{-n}$. If $d(\alpha, \beta) < \varepsilon$ then for all $i \leq n$, $\alpha_i = \beta_i$. In particular, it follows that $K^t(\beta_{[1:n]}|n) = K^t(\alpha_{[1:n]}|n) \leq n - c\log n$, which implies that $\beta \notin A_{n_0}$, i.e., $\beta \in \Sigma^\infty - A_{n_0}$.

2. $\Sigma^\infty - A_{n_0}$ are dense sets.

   We must prove that given an $\alpha \in A_{n_0}$ and any $1 > \varepsilon > 0$ there is $\beta \in \Sigma^\infty - A_{n_0}$ such that $d(\alpha, \beta) < \varepsilon$. Set $n = -\log \varepsilon$ and define β as follows: for all $i \leq n$, $\beta_i = \alpha_i$ and for $i > n$, $\beta_i = 0$.

   It follows, by definition of n, that $d(\alpha, \beta) < \varepsilon$ and it is clear that there is $n' >>> n \geq n_0$ such that $K^t(\beta_{[1:n']}|n') \leq n' - c \log n'$which implies that $\beta \in \Sigma^\infty - A_{n_0}$.

So, using Baire's Theorem we conclude that $\bigcap_{n\in\mathbb{N}} \Sigma^\infty - A_n = \Sigma^\infty - \bigcup_{n\in\mathbb{N}} A_n$ is dense.

Notice that $\omega \notin \bigcup_{n\in\mathbb{N}} A_n$ means that for all n_0 there is $n \geq n_0$ such that $K^t(\omega_{[1:n]}|n) \leq n - c \log n$, i.e., for infinitely many n, $K^t(\omega_{[1:n]}|n) \leq n - c \log n$. □

So, although the complement of the set of polynomial time $c \log(n)$-Kolmogorov-random sequences has measure 0 it is dense. As a corollary, the complement of the set of random sequences is also dense. Since deep objects are not random, this leaves open the possibility of the existence of some deep sequences. Notice that from Theorem 3.4.2, the set of deep sequences has measure 0, but, as proved in the next result, it is dense and so, in fact, deep sequences do exist. The density of deep sequences follows from the following result:

Theorem 3.4.5. *Let* t *be a fixed polynomial. Given* $y \in \Sigma^n$ *and* $0 < \delta < 1$*, there is a string* x *of length* $2^n + n$ *such that* y *is a prefix of* x *and* $\text{depth}^t(x) \geq (1-\delta)2^n - c \cdot n$ *for some constant* c*.*

Proof. Consider a string x of length $2^n + n$ of the form $x = yz$. We claim that there is a string z such that $\text{depth}^t(x) \geq (1-\varepsilon)2^n - c \cdot n$ for some constant c.

It is known (see Theorem 5 in [AFvMV06]) that, for every $0 < \varepsilon < 1$, there is a string z of length 2^n such that $\text{depth}^t(z) \geq (1-\varepsilon)2^n - c \cdot n$, where c is some fixed constant. Then, $\text{depth}^t(yz) \geq (1-\varepsilon)2^n - c \cdot n$.

$$\begin{aligned}
\text{depth}^t(yz) &= K^t(yz) - K(yz) \\
&\geq K^t(z) - K(yz) \\
&\geq K^t(z) - K(z) - K(y) \\
&= \text{depth}^t(z) - K(y) \\
&\geq (1-\varepsilon)2^n - c \cdot n - K(y) \\
&\geq (1-\varepsilon)2^n - c' \cdot n
\end{aligned}$$

the last inequality follows from the fact that $K(y) \leq n + 3 \log n$. □

Theorem 3.4.6. *Let t be a fixed polynomial. The set of sequences ω such that for infinitely many n, $\text{depth}^t(\omega_{[1:n]}|n) \geq (1-\delta)n - O(\log n)$ is dense, where $0 < \delta < 1$ is a fixed constant.*

The proof consists again of adapting the arguments of Theorem 3.3.6 for the particular case of computational depth measure.

Proof. For any natural number n_0 consider the sets:

$$V_{n_0} = \{\omega \in \Sigma^\infty : \exists n \geq n_0, \text{depth}^t(\alpha_{[1:n]}|n) \geq (1-\delta)n - O(\log n)\}$$

If $\alpha \in \bigcap_{n_0} V_{n_0}$ then for all n_0 there is $n \geq n_0$ such that $\text{depth}^t(\omega_{[1:n]}|n) \geq (1-\delta)n - O(\log n)$. So, for infinitely many n, $\text{depth}^t(\omega_{[1:n]}|n) \geq (1-\delta)n - O(\log n)$.

1. V_{n_0} are open sets.

   Let α be a sequence in V_{n_0}. Then, there is an $n \geq n_0$ such that

   $$\text{depth}^t(\alpha_{[1:n]}|n) \geq (1-\delta)n - O(\log n)$$

   Set $\varepsilon = 2^{-n}$. If $d(\alpha, \beta) < \varepsilon$ then for all $i \leq n$, $\alpha_i = \beta_i$. In particular, it follows that $\text{depth}^t(\beta_{[1:n]}|n) = \text{depth}^t(\alpha_{[1:n]}|n) \geq (1-\delta)n - O(\log n)$, which implies that $\beta \in V_{n_0}$.

2. V_{n_0} are dense sets.

   Fix $\varepsilon > 0$ and $\alpha \notin V_{n_0}$. We consider β such that for all $i \leq n = -\log \varepsilon$, $\beta_i = \alpha_i$. This implies that, for any $\omega \in \Sigma^*$, $d(\alpha, \beta_{[1:n]}\omega) < \varepsilon$.

   From Theorem 3.4.5, we can extend $\beta_1 ... \beta_n$ to a string x of length $2^n + n$ such that $\text{depth}^t(x) \geq (1-\delta)2^n - O(n) = (1-\delta)|x| - O(\log |x|)$. It then follows that $\beta = x000... \in V_{n_0}$ and $d(\alpha, \beta) < \varepsilon$.

So, using Baire's Theorem 3.2.16, we conclude that $\bigcap_{n_0} V_{n_0}$ is dense. □

Remark 3.4.7. *The use of polynomial time bounds is not crucial for all the results of this section. In fact, Theorem 5 of [AFvMV06], allows us to have a similar result considering any time bound up to $t(n) = 2^n$.*

3.5 Sophistication vs depth of sequences

Koppel [Kop87] claimed that, for all infinite sequences, sophistication and logical depth are equivalent. However, the proof uses a different definition of Bennett's logical depth since totality in the functions is imposed. In fact, the claimed equivalence would be an unexpected result, as sophistication measures the program length which does not exceed the length of the string and logical depth measures running times which can be arbitrarily large.

In [AF09a], the authors proved that computational depth and sophistication are distinct for finite strings, contradicting Koppel's intuition. In this section, we reinforce the distinctness of these two measures for infinite sequences by proving the existence of sequences that are deep but not very sophisticated. We define packing and Hausdorff dimensional depth instantiating the probability distribution of randomness deficiency to be the time bounded universal distribution $\mathbf{m}^t(x) = 2^{-K^t(x)}$.

Definition 3.5.1. *The packing dimensional depth of a sequence α is:*

$$\mathrm{depth}^t_{\mathrm{Dim}}(\alpha) = \limsup_{n\to\infty} \frac{\delta(\alpha_{[1:n]}|2^{-K^t(\alpha_{[1:n]})})}{n}.$$

where $\delta(x|\mu)$ is the randomness deficiency of x with respect to the semi-measure μ which is defined by $\delta(x|\mu) = \left\lfloor \log \frac{2^{-K(x)}}{\mu(x)} \right\rfloor$.

The difference of the two measures, sophistication and computational depth, follows from the following theorem.

Theorem 3.5.2 (Theorem 7.1.4 in [LV08])**.** *Let t be a time constructible function. There is a recursive sequence ω such that $K^t(\omega_{[1:n]}|n) \geq n - \log(n)$ infinitely often.*

Sketch of the Proof. The sequence ω results from a diagonalization. Consider the recursive function defined by $g(1) = 1$ and $g(n) = 2^{g(n-1)}$.

Set $\omega_1 = 0$. Define $\omega_{[g(n-1)+1:g(n)]}$ recursively as follows: simulate all prefix-free programs of size less than $g(n) - g(n-1)$ for $t(g(n))$ steps each. Extend $\omega_{[1:g(n-1)]}$ to $\omega_{[1:g(n)]}$ such that $\omega_{[1:g(n)]}$ is not the initial segment of any of the previous simulations.

The string obtained by this construction, satisfies $K^t(\omega_{[1:g(n)]}|g(n)) \geq g(n) - g(n-1)$. Since $\log(g(n)) = g(n-1)$ then for infinitely many n, $K^t(\omega_{[1:n]}|n) \geq n - \log(n)$. □

Observe that from the fact that ω is recursive we have $K(\omega_{1:n}|n) \leq \log n$ for all n. So, for all n, $\mathrm{soph}_c(\omega_{1:n}) \leq O(\log n)$ and thus

$$\underline{\mathrm{soph}}_c(\omega_{1:n}) \leq \overline{\mathrm{soph}}_c(\omega_{1:n}) \leq \lim_n \frac{O(\log n)}{n} = 0.$$

On the other hand, for infinitely many n, $\mathrm{depth}^t(\omega_{1:n}) \geq n - O(\log n)$. So,

$$\begin{aligned}
\mathrm{depth}^t_{\mathrm{Dim}}(\omega) &= \limsup_{n\to\infty} \frac{\delta(\omega_{[1:n]}|2^{-\mathrm{K}^t(\omega_{[1:n]})})}{n} \\
&= \limsup_{n\to\infty} \frac{\log(2^{-\mathrm{K}(\omega_{[1:n]})}/2^{-\mathrm{K}^t(\omega_{[1:n]})})}{n} \\
&= \limsup_{n\to\infty} \frac{\mathrm{K}^t(\omega_{[1:n]}) - \mathrm{K}(\omega_{[1:n]})}{n} \\
&= \limsup_{n\to\infty} \frac{\mathrm{depth}^t(\omega_{[1:n]})}{n} \\
&= 1
\end{aligned}$$

It is possible to obtain a similar result for depth defined with "lim inf" but the separation is not so strong, since we prove that the difference is not 1 but a constant smaller than 1 that depends on the time t.

Definition 3.5.3. *The Hausdorff dimensional depth of a sequence α is:*

$$\mathrm{depth}^t_{\mathrm{dim}}(\alpha) = \liminf_{n\to\infty} \frac{\delta(\alpha_{[1:n]}|2^{-\mathrm{K}^t(\alpha_{[1:n]})})}{n}.$$

Theorem 3.5.4 (Theorem 7.1.3 in [LV08])**.** *Let t be a time constructible function and k the index of the Turing machine that computes t. There is a recursive enumerable set A such that, the characteristic sequence χ_A satisfies $\mathrm{K}^t(\chi_{A[1:n]}|n) \geq \frac{n}{2^{2^k+1}}$ for all n.*

Since A is recursively enumerable we know that for all n, $\mathrm{K}(\chi_{A[1:n]}) \leq O(\log n)$. So, for all n, $\mathrm{soph}_c(\chi_{A[1:n]}) \leq O(\log n)$ and then:

$$\underline{\mathrm{soph}}_c(\chi_A) \leq \lim_n \frac{O(\log n)}{n} = 0.$$

On the other hand, from last theorem, we know that for all n, $\mathrm{K}^t(\chi_{A[1:n]}) \geq \frac{n}{2^{2^k+1}} - \log n$. Thus,

$$\begin{aligned}
\mathrm{depth}^t_{\mathrm{dim}}(\chi_A) &= \liminf_{n\to\infty} \frac{\delta(\chi_{A[1:n]}|2^{-\mathrm{K}^t(\chi_{A[1:n]})})}{n} \\
&= \liminf_{n\to\infty} \frac{\log(2^{-\mathrm{K}(\chi_{A[1:n]})}/2^{-\mathrm{K}^t(\chi_{A[1:n]})})}{n} \\
&= \liminf_{n\to\infty} \frac{\mathrm{K}^t(\chi_{A[1:n]}) - \mathrm{K}(\chi_{A[1:n]})}{n} \\
&= \liminf_{n\to\infty} \frac{\mathrm{depth}^t(\chi_{A[1:n]})}{n} \\
&\geq \frac{1}{2^{2^k+1}}
\end{aligned}$$

Chapter 4

Low depth witnesses of SAT

The content of this chapter is based on the following publication:

[AFPS07] *L. Antunes, L. Fortnow, A. Pinto and A. Souto,* Low depth witnesses are easy to find*, in Proceedings of 22nd Annual of IEEE Conference on Computational Complexity 2007, pages:46-51, San Diego, United States of America;*[1]

We show unconditionally how to probabilistically find satisfying assignments for formulas that have at least one assignment of logarithmic depth. The converse holds under a standard hardness assumption though fails if **BPP** = **FewP** = **EXP**. *We also prove that assuming the existence of good pseudorandom generators one can not increase the depth of a string efficiently.*

4.1 Motivation

We investigate a computational depth characterization of the set of formulas in **SAT** that are solved by a probabilistic algorithm and continue the study of computational depth.

Let ϕ be a Boolean formula having a satisfying assignment w of computational depth d. We show how to probabilistically find an assignment w', not necessarily equal to w, in time exponential in d such that $\phi(w') = \mathsf{True}$.

We show that under a standard hardness assumption the converse also holds, manely that, if exponential time is not infinitely often in subexponencial space, then any formula ϕ for which we can find a satisfying assignment w in probabilistic time admits a satisfying

[1] This paper has been recently accepted for publication in the Computational Complexity journal

assignment of low computational depth. On the other hand, under the unlikely, but open case, that **BPP** = **FewP** = **EXP**, one can find formulas that have a single solution of high computational depth that can be found quickly using a probabilistic algorithm. The class **FewP** is the set of problems accepted by a polynomial time nondeterministic Turing machine which has at most a fixed polynomial number of accepting paths for each input.

We also look at the question of whether one can increase computational depth of a string efficiently. We show that under the same hardness assumption, one can not significantly increase the depth of a string in polynomial time. Once again, if **BPP** = **EXP** we show examples where one can produce a string of high depth from a string of very low depth. Finally, we explore the question as to whether a triangle inequality holds for conditional depth.

4.2 Preliminaries

4.2.1 Pseudorandom generators

Pseudorandom generators are efficiently computable functions which stretch a seed into a long string, so that, apart from not being random in the classical sense, testing this fact requires an unrealistic amount of time, i.e., for a random input its output looks perfectly random for a resource-bounded machine. The existence of pseudorandom generators has many computational complexity implications. For example, Impagliazzo, Levin and Luby, in [ILL89], proved that their existence is equivalent to the existence of one-way functions and thus it implies a separation of **P** and **NP**.

In this work we need some pseudorandom generators based on hard functions.

The following lemma is implicit in the work of Nisan and Wigderson [NW94].

Lemma 4.2.1 (Nisan-Wigderson [NW94])**.** *Suppose we have a set $\mathcal{H}$ of functions from Σ^k to Σ^m, a polynomial p and a parameter n satisfying $\log(m) \leq k \leq p(n)$ with the following properties:*

1. *At least $3/4$ of all possible functions mapping Σ^k to Σ^m are in $\mathcal{H}$;*
2. *For some a, there is a $\Sigma_a^{p(n)}$-machine with oracle access to a function H which on input 1^n will accept exactly when H is in $\mathcal{H}$.*

Then there is a function $H'(x, r)$ with $x \in \Sigma^k$ and $|r|$ polynomial in n, such that each output bit is computed in polynomial time (in n) and for at least $2/3$ of the possible r, $\hat{H}_r(x) = H'(x, r)$ is in $\mathcal{H}$.

Proof. View a function H as a binary string of length $M = m2^k$ and the $\Sigma_a^{p(n)}$-machine as a constant depth circuit C of size $2^{n^{O(1)}}$. Nisan and Wigderson [NW94] show how to create a pseudorandom generator G, based on the parity function, that maps a seed of size polynomial in n to M bits that fools C. Each output bit of the generator G can be computed in time polynominal in n. H' is easily constructed from G. □

One of the most important applications of pseudorandom generators is derandomization. This procedure has as ultimate goal to prove that $\mathbf{P} = \mathbf{BPP}$ but, that conclusion was not yet proved.

Impagliazzo and Wigderson [IW96] strengthen the work of Nisan and Wigderson to show how to achieve full derandomization based on strong hardness assumptions. Klivans and van Melkebeek [KvM02] generalize Impagliazzo-Wigderson by showing that the results hold for relativized worlds in a strong way.

Lemma 4.2.2 (Impagliazzo-Wigderson, Klivans-van Melkebeek)**.** *For any oracle A, suppose that there are languages in $\mathbf{DTIME}(2^{O(n)})$ that for some $\varepsilon > 0$, can not be computed by circuits of size $2^{\varepsilon n}$ with access to an oracle for A. Then there is a k and a pseudorandom generator $g : \Sigma^{k \log n} \to \Sigma^n$ computable in time polynomial in n such that for all relativizable circuits C of size n*

$$\left| \Pr_{s \in \Sigma^{k \log n}} (C^A(g(s)) = 1) - \Pr_{r \in \Sigma^n} (C^A(r) = 1) \right| = o(1).$$

We need the following hardness hypothesis for many of the derandomization results in this work.

Hypothesis 4.2.3. *There is a language L in $\mathbf{DTIME}(2^{O(n)})$ that for some $\varepsilon > 0$, L can not be computed on infinitely many inputs lengths by circuits of size $2^{\varepsilon n}$ with Σ_2^p gates.*

Miltersen [Mil01] shows that Hypothesis 4.2.3 follows from a uniform statement of time versus space. He proved that if $\mathbf{DTIME}(2^{O(n)})$ is not contained in $\mathbf{DSPACE}(2^{o(n)})$ then $\mathbf{DTIME}(2^{O(n)})$ contains a language that does not have circuits of size $2^{o(n)}$ with Σ_2^p gates or even **PSPACE** gates.

Lemma 4.2.4 (Miltersen)**.** *If $\mathbf{DTIME}(2^{O(n)})$ is not contained in $\mathbf{DSPACE}(2^{o(n)})$ for infinitely many input lengths then, for every language A in **PSPACE**, there is some $\varepsilon > 0$ such that $\mathbf{DTIME}(2^{O(n)})$ contains a language that does not have circuits of size $2^{\varepsilon n}$ with access to A for infinitely many input lengths.*

4.3 Finding low-depth witnesses

We study the relationship between the depth of a solution for a Boolean formula and the existence of a probabilistic algorithm to find a satisfying assignment for the formula. For this section, we assume that n represents the number of variables in the Boolean formula considered.

Theorem 4.3.1. *Let c be a constant and t some polynomial. There is a probabilistic polynomial time algorithm such that, if ϕ is a Boolean formula over n variables with a satisfying assignment w of $\mathrm{depth}_t(w|\phi) \leq c \log n$, then on input ϕ the algorithm outputs a satisfying assignment of ϕ with probability close to one.*

Proof. Let $m = K(w|\phi)$. Since $\mathrm{depth}^t(w|\phi) = K^t(w|\phi) - K(w|\phi) \leq c \log n$ we can write

$$m' = K^t(w|\phi) \leq m + c \log n \tag{4.3.1}$$

Now consider the following set:

$$A = \left\{ z | \phi(z) = \mathsf{True} \text{ and } K^t(z) \leq m' \right\}$$

where we have used $\phi(z)$ to denote the value of ϕ for the assignment z. The second condition of the definition of A says that if $z \in A$, there is a program p such that $|p| \leq m'$ and p generates z in time t. By construction, given ϕ and m', A is a computable set and $w \in A$. From the fact that $w \in A$, we get $m = K(w|\phi) \leq K(w|\phi, m') + K(m') \leq \log |A| + O(\log n)$. Using Inequality 4.3.1 we can write, for some constant c' depending on t and on c:

$$|A| \geq \frac{2^m}{c'n} \geq \frac{2^{m'}}{\mathrm{poly}(n)}.$$

where $\mathrm{poly}(n) = c' n^{c'}$. The following probabilistic algorithm M produces a satisfiable assignment for ϕ:

Input: Formula ϕ;

Output: z an assignment for ϕ;

1. Guess $m' \leq n + c \log n$;
2. Generate randomly a program p of length at most m';
3. Run the universal Turing machine U with program p and the formula ϕ for t steps and let z be the output;

4. If z is a satisfiable assignment accept, otherwise reject.

The probability of this algorithm generating an assignment for ϕ is at least:

$$\frac{1}{n + c \log n} \times \frac{|A|}{2^{m'}} \geq \frac{2^{m'}/\mathrm{poly}(n)}{2^{m'}} = \frac{1}{\mathrm{poly}(n)}.$$

If we run the algorithm a polynomial number of times, with high probability, one of those runs will produce a satisfying assignment of ϕ. □

Consider now the converse problem. Given a probabilistic algorithm which finds valid assignment for some Boolean formula ϕ in time t, can we say that the Boolean formula has a witness w with low computational, i.e., satisfying $\mathrm{depth}^t(w|\phi) \leq O(\log n)$, where n is the number of variables occurring in ϕ?

The answer to this problem depends on the assumptions we make. The answer is false if we assume that **BPP** = **FewP** = **EXP** and is true if we assume that good pseudorandom generators exist.

Theorem 4.3.2. *If* **BPP** = **FewP** = **EXP** *then we can find a witness for every satisfiable formula in probabilistic polynomial time but for every polynomial* q *there are infinitely many Boolean formulas* ϕ *and some* $\varepsilon > 0$ *such that:* $\mathrm{depth}^q(w|\phi) \geq n^\varepsilon$*, for all satisfying assignments* w *of* ϕ*.*

Proof. For any formula ϕ we can find a satisfying assignment in exponential time and since **EXP** = **BPP** we can find a witness probabilistically quickly, i.e., in probabilistic polynomial time.

Fix a complete language L for **EXP** and since L is in **FewP**, let M be an **NP** machine accepting L with at most $p(n)$ accepting paths on each input.

Let x be a string in L with $|x| = n$. By Cook's reduction of the **NP**-completeness of **SAT**, we can compute a ϕ that has the same number of satisfying assignments as $M(x)$ has accepting computations. Let w be any witness of ϕ. We have:

- $K(w|\phi) = O(\log n)$, since we can search for the satisfying assignments w of ϕ and identify one of them by its index.

- Let q be any polynomial. If $K^q(w|\phi) \leq n^{o(1)}$ then we can compute whether x is in L in deterministic time $2^{n^{o(1)}}$ by trying all small programs and seeing if any of them produce a witness.

By the time hierarchy theorem[2], **EXP** is not contained in deterministic time $2^{n^{o(1)}}$, so there must be infinitely many ϕ with $K^q(w|\phi) \geq n^\delta$ for some $\delta > 0$. Thus, we get $\text{depth}^q(w|\phi) \geq n^\varepsilon$ for any $\varepsilon < \delta$. □

We now prove that if good pseudorandom generators exist then the converse proposition holds, i.e., under the assumption that exponential time is not infinitely often in subexponential space (which is sufficient for the existence of good pseudorandom generators that stretch a seed of length $O(\log n)$ to a string of length 2^n), all the satisfying assignments w of ϕ with n variables produced by a probabilistic algorithm in time $t(n)$ have $\text{depth}^t(w|\phi) \leq O(\log n + \log(t(n)))$.

Proposition 4.3.3. *Let c be a constant, t some polynomial and A a probabilistic algorithm that on an input Boolean formula with n variables runs in time $t(n)$. Under Hypothesis 4.2.3, for any n and formula ϕ over n variables if the algorithm outputs a satisfying assignment of ϕ with probability at least $2/3$ then there is a satisfying assignment w of ϕ with $\text{depth}_t(w|\phi) \leq c(\log n + \log t)$.*

Notice that the probabilistic algorithm may output different witnesses on different random coin tosses used by the algorithm.

Proof. Since there is a probabilistic algorithm that finds a valid assignment w for ϕ in time $t(n)$, there is a random string r satisfying $|r| \leq t(n)$ and such that $\text{depth}^t(w|\phi, r) = O(1)$. Now, given a seed of length $O(\log r + \log t)$ we can derandomize the **BPP** algorithm using the pseudorandom generator. So $K^t(w|\phi) \leq O(\log r + \log t) \leq O(\log n + \log t)$ and then $\text{depth}^t(w|\phi) \leq O(\log n + \log t)$. □

4.4 Depth can not increase rapidly

We show that if f is an honest polynomial time computable function then it can not significantly increase the depth of its argument, i.e., deep objects are not possible to be quickly produced from shallow ones. A function f is honest if the size of the output is polynomially related with the size of the input.

We start showing that it holds for honest efficiently computable functions with few inverses.

[2]This theorem formalizes the intuition that if we allow more time to a Turing machine of computation, then it can decide more languages. It has been proved, as mentioned in the Preliminaries Chapter, by Hartmanis and Stearns in [HS65].

Proposition 4.4.1. *Let* $f : \Sigma^* \to \Sigma^*$ *be a polynomial time computable function that is at most* m *to* 1*. If* $y = f(x)$ *and* x *is shallow, then* $\text{depth}^t(y) \leq \log m + O(1)$*.*

Proof. Since we assume that f is a fixed polynomial time function and it is independent from the strings involved, it is clear that f has a constant time bounded Kolmogorov complexity.

Since $f(x) = y$, we can write:

$$K^t(y) \leq K^t(f) + K^t(x|f) \leq O(1) + K(x)$$

Now, consider the following set:

$$A_y = \{w : f(w) = y\}$$

Since f is computable and is at most m to 1, A_y is recursive (given y) and has at most m elements. Thus:

$$K(x) \leq K(y) + \log |A_y| = K(y) + \log m$$

Then using the last two inequalities we get:

$$K^t(y) \leq K(x) + O(1) \leq K(y) + \log m + O(1) \Leftrightarrow \text{depth}^t(y) \leq \log m + O(1)$$

□

Theorem 4.4.2. *Let* $f : \Sigma^* \to \Sigma^*$ *be a computable* $1-1$ *function that requires superpolynomial time, but has the property that with an advice string* s *becomes polynomial. Then the advice string must have length at least equal to the difference in depth between* x *and* $y = f(x)$*.*

Proof. Let t be a fixed polynomial such that f with the advice s is computable in time at most t. We have that $K^t(y) \leq K^t(x) + K^t(y|x, s) + |s| = K^t(x) + O(1) + |s|$. Then, $K^t(y) \leq \text{depth}^t(x) + K(x) + |s| + O(1)$ and since $K(x) = K(y)$, we get

$$\text{depth}^t(y) \leq \text{depth}^t(x) + |s| + O(1)$$

□

In the next result we show that, relative to a random oracle for every honest efficiently computable f, the depth of $f(x)$ can not be much greater than the depth of x. Later on we will replace the random oracle by a pseudorandom generator.

We say that a statement holds relative to a random oracle if, when we choose an oracle R uniformly at random, the statement is true with probability one, when all computations have access to the oracle R.

Lemma 4.4.3. *For most oracles* R *the following holds: If* $f : \Sigma^* \to \Sigma^*$ *is an honest polynomial time computable function relative to* R *then, for any polynomial* t *there is a polynomial* t' *so that* $\mathrm{depth}^{t'}(f(x)|f,R) \leq \mathrm{depth}^{t}(x|f,R) + O(\log|x|)$ *for all* $x \in \Sigma^*$.

Proof. Fix x, let $y = f(x)$ and define the set A_y as the set of strings $z' \in \Sigma^{K^t(x|f,R)}$ for which there is a prefix z of z' such that $U^{f,R}(z) = x'$ in time t and $f(x') = y$.

By construction, x can be computed from a string in A_y (in fact, from x^* the first string in lexicographic order that, with oracle access to f and R, produces x in time t). Since f is computable in polynomial time in R, A_y can be computed (given $K^t(x|f,R)$ and y) by enumerating all programs of size up to $K^t(x|f,R)$, if a program z outputs a string x' in time t with $f(x') = y$ then we output all $z' \in \Sigma^{K^t(x|f,R)}$ that extend z. So,

$$K(x|y,R) \leq K(\chi_{A_y}|y,f,R) + \log|A_y| + O(\log n) = \log|A_y| + O(\log n)$$

which implies $|A_y| \geq 2^{K(x|y,f,R)-O(\log n)}$. By the relativized symmetry of information we have:

$$K(x|y,f,R) = K(y|x,f,R) + K(x|f,R) - K(y|f,R) \pm O(\log n).$$

As $y = f(x)$ and f is known, we have that $K(y|x,f,R) = O(1)$ and so

$$K(x|y,f,R) = K^t(x|f,R) - \mathrm{depth}^t(x|f,R) - K(y|f,R) \pm O(\log n)$$

implying that

$$|A_y| \geq 2^{K(x|y,f,R)-O(\log n)} = \frac{2^{K^t(x|f,R)}}{2^{\mathrm{depth}^t(x|f,R)+K(y|f,R)+c\log n}}$$

for some constant c.

If we randomly pick a program of size $K^t(x|f,R)$, the probability that it is in A_y is at least

$$\frac{1}{2^{\mathrm{depth}^t(x|f,R)+K(y|f,R)+c\log n}}.$$

Since f is honest there is some q such that $|z| \leq |y|^q$ for all z such that $f(z) = y = f(x)$. Let $m = |y|^q + 1$. Note that since f is polynomial time computable and honest, m is bounded by a polynomial in $|x|$.

Let c' be a constant to be specified later. Define a function $g : \Sigma^k \to \Sigma^\ell$ where $k = \mathrm{depth}^t(x|f,R) + K(y|f,R) + c'\log n$ and $\ell = K^t(x|f,R)$, by letting the j^{th} bit of the output of $g(w)$ be $R(\langle 1^m, w, j\rangle)$. When R is chosen at random the output of $g(y)$ is completely independent from the part of R used to define A_y since A_y depends on strings

in R only of length less than m. Thus,

$$\begin{aligned}
\Pr[\exists w : g(w) \in A_y] &= 1 - \Pr[\forall w : g(w) \notin A_y] \\
&\geq 1 - \left(1 - \frac{1}{2^{\mathrm{depth}^t(x|f,R)+K(y|f,R)+c\log n}}\right)^{2^{\mathrm{depth}^t(x|f,R)+K(y|f,R)+c'\log n}} \\
&= 1 - e^{-2^{(c'-c)\log n}} \\
&= 1 - e^{-n^{c'-c}} \\
&\geq 1 - 2^{-n^{c'-c}}
\end{aligned}$$

Then,

$$\begin{aligned}
\Pr[\forall y \exists w : g(w) \in A_y] &= 1 - \Pr[\exists y \forall w_y : g(w_y) \notin A_y] \\
&\geq 1 - 2^n 2^{-n^{c'-c}} \\
&= 1 - 2^{-n^{c'-c}+n}
\end{aligned}$$

For the appropriate choice of c' we can find a w_y for every y, with high probability, such that $g(w_y)$ is in A_y, i.e., we can find a w_y for which there is a prefix z of $g(w_y)$ satisfying $f(U(z)) = y$. So, for some polynomial t' we have:

$$K^{t'}(y|f,R) \leq |w_y| + O(\log n) = \mathrm{depth}^t(x|f,R) + K(y|f,R) + O(\log n).$$

and thus conclude that

$$\mathrm{depth}^{t'}(y|f,R) \leq \mathrm{depth}^t(x|f,R) + O(\log n).$$

By the Kolmogorov zero-one law [Kol50] this high probability result implies these statements must hold over the choice of R with probability one. □

We now improve the previous result to more general terms under a standard hardness assumption, namely exponential time is not infinitely often in subexponential space. The idea behind the proof is to compose the pseudorandom generator in Lemma 4.2.2 with the pseudorandom generator in Lemma 4.2.1, as done by Antunes and Fortnow [AF09b].

Theorem 4.4.4. *Let $f : \Sigma^* \to \Sigma^*$ be an honest polynomial time computable function and $x \in \Sigma^n$. Then, under Hypothesis 4.2.3, for any polynomial $t(n)$ larger than the computation time of f there is a polynomial $t'(n)$ so that $\mathrm{depth}^{t'}(y|f) \leq \mathrm{depth}^t(x|f) + O(\log n)$, where $y = f(x)$.*

Proof. Since $\mathrm{depth}^t(.)$ decreases as t increases, we assume without loss of generality that t is much larger than the running time of f.

Notice that this result is true without any assumptions if $K(f(x)) \geq K(x) - O(\log n)$ as, $\mathrm{depth}^t(f(x)|f) \leq K^t(f(x)|f) - K(f(x)|f) \leq K^t(f(x)|f) - K(x|f) + O(\log n) \leq K^t(x|f) -$

$K(x|f) + O(\log n) = \text{depth}^t(x|f) + O(\log n)$. So we will focus on the case $K(f(x)) < K(x) - c \log n$ for all constant c.

Continuing the proof of Lemma 4.4.3, consider the set $\mathcal{H}$ of functions $h : \Sigma^k \to \Sigma^m$ where $k = \text{depth}_t(x|f) + K(y|f) + c' \log n$ and $m = K^t(x|f)$ such that for all y there is a w_y and a prefix z of $h(w_y)$ satisfying $f(U(z)) = y$. Notice

1. By the same argument as in the proof of Lemma 4.4.3, a randomly chosen h will fall into $\mathcal{H}$ with probability very close to one.

2. By definition of $\mathcal{H}$, we can determine whether h sits in $\mathcal{H}$ in $\Pi_2^{p(n)} \subseteq \Sigma_3^{p(n)}$ with oracle access to h.

With the above conditions and the fact that $\log m < k < 3n$ for sufficiently large c', the set $\mathcal{H}$ fulfills the requirements of Lemma 4.2.1 so we can use a polynomially-long random seed to describe an h in $\mathcal{H}$. Notice that in the worst case we will need to describe a function that it characterized by a string of length $\text{poly}(n) \cdot 2^n$ that requires a seed of length $\text{poly}(\log(\text{poly}(n) \cdot 2^n))$ which is polynomial in n. Given a good pseudorandom generator (Lemma 4.2.2) we can use an $O(\log n)$ bit random string to generate the seed for the first generator. Notice that to achieve $\text{poly}(n)$ bits using the pseudorandom generator of Lemma 4.2.2 we need a seed of $O(\log(\text{poly}(n))) = O(\log n)$. We call the result of the composition of the two pseudorandom generators G.

Composing this procedure with the procedure of the previous theorem, we have a way to describe y by the following program for a fixed y:

Input: a seed s and a witness w_y

Output: y

1. Compute $s' = G(s)$.
2. Consider s' as a function h that maps a sequence of length $\text{depth}^t(x) + K(y) + O(\log n)$ into a program of size at most $K^t(x)$, as per Lemma 4.4.3.
3. Compute $p = h(w_y)$, giving a program in the set A_y.
4. Run the universal Turing Machine with the appropriated prefix of the program p and call the output x'.
5. Compute $f(x')$. (By the construction in the Lemma 4.4.3, this is y.)
6. Output y.

Since all constructions are independent of any given instance, we have a description for y requiring only the description of s, w_y and an $O(\log n)$ term to account for the information needed to consider the prefix, that can be computed in time $t'(n)$, a polynomial depending on the running time of the function h, depending on t, and the running time of p, again depending on t. Therefore,

$$\begin{aligned} K^{t'}(y|f) &\leq |s| + |w_y| + O(\log n) \\ &= O(\log n) + \text{depth}^t(x|f) + K(y|f) \end{aligned}$$

So, $\text{depth}^{t'}(y|f) \leq \text{depth}^t(x|f) + O(\log n)$. □

However if **BPP** = **EXP** the previous result does not hold.

Theorem 4.4.5. *Assume* **BPP** = **EXP**. *For all polynomial* t*, there is a polynomial time computable honest function* f *such that, for all polynomials* t'*, there is a polynomial* q *such that for every natural number* n *there are strings* y *and* x *with* $|y| = n$ *and* $|x| = q(n)$ *satisfying:*

1. $y = f(x)$,
2. $\text{depth}^t(y) \geq n - O(\log n)$, *and*
3. $\text{depth}^{t'}(x) \leq O(\log n)$.

Proof. Fix n. Let y be the lexicographically least string such that $K^t(y) \geq n$. We can compute y so $K(y) \leq O(\log n)$ and $\text{depth}^t(y) \geq n - O(\log n)$.

We can find y in time $2^{O(n)}$ given n and $t(n)$ so by the hypothesis **BPP** = **EXP** there is a probabilistic algorithm A that will compute y given 1^n. Let t' be the running time of A.

Let $m = q(n) > n$ be the number of random bits used by A. Let $f(r)$ simulate A using r as the random coins. Let x be Kolmogorov random of length m.

We have $f(x) = y$ since the set of strings that cause f to give the wrong answer will be small and all such strings will have low Kolmogorov complexity.

Finally we have $\text{depth}^{t'}(x) \leq O(\log n)$ because x is random. □

4.5 Properties of conditional depth

Bennett [Ben88] noticed that the impossibility of rapid growth of depth can not be extended to a transitive law relative to shallowness, i.e., if x is shallow relative to y

and y is shallow relative to z, this does not necessarily imply that x is shallow relative to z. Bennett considered z to be a Kolmogorov random string of size n, $y = 0^n$ and $x = z \oplus d$ where d is some deep string.

We conjecture that the transitive law relative to shallowness does not hold. However, assuming that pseudorandom generators exist, we show that depth satisfies an analog of a triangular inequality, which informally states that the depth of y is bounded by the depth of x and the depth of y given x. In fact we prove something slightly stronger (Corollary 4.5.2).

Theorem 4.5.1. *Under the Hypothesis 4.2.3, given a polynomial* $t(n)$ *there is a polynomial* $t'(n)$ *such that for any* $x, y, z \in \Sigma^*$ *and* $n = \max(|x|, |y|, |z|)$

$$\mathrm{depth}^{t'}(y|z) \leq \mathrm{depth}^{t}(x|z) + \mathrm{depth}^{t}(y|x,z) + O(\log n)$$

Proof. Define $a = K^t(x|z)$, $b = K^t(y|x,z)$, $\mathrm{depth}^t(x|z) = r$ and $\mathrm{depth}^t(y|x,z) = s$. Then,

$$K(x|z) = a - r \text{ and } K(y|x,z) = b - s.$$

Consider the set A consisting of all w such that there are a $|u| \leq a$ and $|v| \leq b$ with $U^t(u,z) = w$ and $U^t(v,w,z) = y$, i.e.,

$$A = \left\{ w | \, K^t(w|z) \leq a \wedge K^t(y|w,z) \leq b \right\}.$$

By construction, x is an element of A and A is computable given y and z. Then, $K(x|y,z) \leq \log |A| + c$, i.e., A has at least $2^{K(x|y,z)-c}$ elements. By symmetry of information, we have that:

$$\begin{aligned} K(x|y,z) &\geq K(y|x,z) + K(x|z) - K(y|z) - O(\log n) \\ &= b - s + a - r - K(y|z) - O(\log n) \end{aligned}$$

Thus,

$$|A| \geq \frac{2^{a+b}}{2^{r+s+K(y|z)+c' \log n}}$$

for some constant c'. The probability of a random program p of size smaller than $a + b + O(\log n)$ to generate y is at least $\frac{1}{2^{r+s+K(y|z)+c' \log n}}$. Using a similar construction of Theorem 4.4.4 we can find a seed of size $r+s+K(y|z)+O(\log n)$ that generates a program producing y within polynomial time t'. So,

$$K^{t'}(y|z) \leq K(y|z) + r + s + O(\log n)$$

and then

$$\mathrm{depth}^{t'}(y|z) \leq r + s + O(\log n).$$

□

Corollary 4.5.2. *Under the Hypothesis 4.2.3, given a polynomial* $t(n)$ *there is a polynomial* $t'(n)$ *such that for any* $x, y \in \Sigma^*$ *and* $n = \max(|x|, |y|)$

$$\text{depth}^{t'}(y) \leq \text{depth}^{t}(x) + \text{depth}^{t}(y|x) + O(\log n).$$

However if we replace the existence of pseudorandom generators assumption by the assumption $\mathbf{BPP} = \mathbf{EXP}$ the previous results do not hold. In fact, in the next result we show a pair of strings x and y that under the assumption $\mathbf{BPP} = \mathbf{EXP}$, satisfy $\text{depth}^{t}(x)$ and $\text{depth}^{t}(y|x)$ are small and $\text{depth}^{t}(y)$ is big.

Theorem 4.5.3. *If* $\mathbf{BPP} = \mathbf{EXP}$*, then there are* $x, y \in \{0,1\}^*$ *such that* $\text{depth}^{t}(x) \leq O(1)$ *and* $\text{depth}^{t}(y|x) \leq O(1)$ *but* $\text{depth}^{t}(y) \geq n - O(1)$*, with* $n = \max(|x|, |y|)$ *and* t *a polynomial in* n*.*

Proof. Let x be a Kolmogorov random string, i.e., such that $K(x) \geq |x|$, and y the lexicographically least string satisfying $K^{t}(y) \geq |y|$. Since all random strings are shallow, $\text{depth}^{t}(x) \leq O(1)$.

The string y can be computed by the following procedure: enumerate all binary strings in lexicographic order and for each string w, compute $K^{t}(w)$. If this number is bigger $|y|$, then output w and stop. Thus, $K(y) = O(1)$. On the other hand, by construction, $K^{t}(y) \geq |y|$, so $\text{depth}^{t}(y) \geq |y| - O(1)$.

To finish we prove that $\text{depth}^{t}(y|x)$ is a constant. Since $K(y) \leq O(1)$, then $K(y|x) \leq O(1)$. The program for y given above enumerates at most $2^{|y|}$ strings before outputting a result, and for each of them it executes up to a polynomial number of steps. Since by assumption $\mathbf{BPP} = \mathbf{EXP}$, there is a probabilistic algorithm that takes a certain random input, runs in polynomial time and outputs y. We let this random input be x. Taking the size of the probabilistic algorithm to be a constant, $K^{t}(y|x) = O(1)$ and thus $\text{depth}^{t}(y|x) = O(1)$. □

Chapter 5

Complexity cores

The contents of this chapter is based on the following publication:

[Sou10] A. Souto, Kolmogorov compexity cores*, in Proceeding of Computability in Europe, volume 1658 of Lecture Notes in Computer Science, pages:376-385, Springer-Verlag Berlin Heidelberg, 2010, Azores, Portugal;*

We study the relationship between complexity cores of a language and the descriptional complexity of the characteristic sequence of the language based on Kolmogorov complexity.

Intuitively, a complexity core is a set of hard instances of a language, i.e. instances which can not be decided in polynomial time.

In the last chapter we showed that we can efficiently find satisfying assignments for formulas that have at least one assignment of logarithmic depth. This result made us conjecture that formulas having only high depth witnesses form a complexity core, i.e., they are the hardest formulas to decide. However this turns out not to be the case.

Nevertheless, we prove that a recursive set $\mathtt{A}$ *has a complexity core (respectively, proper complexity core) if for all constants* $\mathtt{c}$ *(respectively, all polynomial* $\mathtt{p}(\mathtt{n})$*), the computational depth of the characteristic sequence of* $\mathtt{A}$ *up to length* $\mathtt{n}$ *is larger than* $\mathtt{c}$ *(respectively larger than* $\mathtt{p}(\mathtt{n})$*) infinitely often. We also explore the connection with average case complexity. In particular, we show that if a language has a complexity core of exponential density, and the strings are distributed according to a time bounded version of the universal distribution, then it can not be accepted in average polynomial time.*

5.1 Motivation

A polynomial complexity core of a language A is the set of strings for which any Turing machine deciding A requires time larger than a polynomial for almost all strings. Lynch [Lyn75] proved that any recursive set that is not in **P** admits an infinite polynomial complexity core. Later, Orponen and Schöning [OS84] proved that, if each algorithm for a language has a non-sparse set of "hard" inputs, then, in fact, the language has a non-sparse proper polynomial complexity core.

In this chapter we study a connection between the computational depth of the characteristic sequences of languages and the existence of (proper) complexity cores. We address this issue by studying the computational depth of sets in the computational classes **P**, **EXP**, **FULL-P/log** and **P/poly**. First, we give a characterization of the recursive sets not in **P**, thus admitting a polynomial complexity core, based on computational depth. Then we prove that if a recursive set is not in **P/poly** the depth of its characteristic sequence is larger than any polynomial almost everywhere. We also establish a similar relationship between sets of logarithmic depth and the class of **FULL-P/log** .

The complexity of a problem is usually measured in terms of the worst case behavior of algorithms. Many algorithms with a bad worst case performance have a good performance in practice, since instances requiring a large running time rarely occur. Thus, in some cases, the average case complexity of a problem is a more significant measure than its worst case complexity. In order to address this problem, Levin [Lev86] introduced the theory of average case complexity, giving a formal definition of Average Polynomial Time for a language L and a distribution μ. Some languages may remain hard in the worst case but can be solved efficiently in average polynomial time for all reasonable distributions. We prove that any Turing machine deciding a proper complexity core of exponential density can not recognize the language in average polynomial time when a time bounded version of the universal distribution is considered.

5.2 Preliminaries

5.2.1 Complexity Cores

The concept of polynomial complexity core was introduced by Lynch [Lyn75] and was developed to capture the intuition of what should be the set of hard instances of a set.

Definition 5.2.1 (Complexity Core)**.** *Let* M *be a Turing machine and* t *a time con-*

structible function. We denote the set of t*-hard inputs for* M *by*

$$H(M,t) = \{x \in \Sigma^* : time_M(x) > t(|x|)\}.$$

Let $\mathcal{F}$ *be a class of functions. A set* $C \subset \Sigma^*$ *is a* $\mathcal{F}$ *complexity core for a language* $L \subset \Sigma^*$*, if given any Turing machine* M *accepting* L *and any function* p *in* $\mathcal{F}$*,* C *is almost everywhere contained in* $H(M,p)$*. A complexity core* C *for* L *is proper if* $C \subset L$*.*

In particular, if $\mathcal{F} = \mathbf{P}$*, the set of hard instances is called polynomial complexity core.*

Rather than having a set of difficult instances for each machine, this notion captures the idea of hard instances of a language in the sense that it is independent from the choice of the Turing machine deciding the language.

Lynch proved that given a recursive language L not in $\mathbf{P}$ there is an infinite polynomial complexity core C for L.

Theorem 5.2.2 (Lynch)**.** *If* A *is a recursive set not in* $\mathbf{P}$*, then there is an infinite recursive set* C *that is a polynomial complexity core for* A*.*

The idea of the proof is to recursively eliminate strings that are decided by some Turing machine M_i in some polynomial time p_k, where p_k is a collection of increasing polynomials and where $i \leq k$ and we cycle over the lexicographic order strings, the machines and the polynomials.

Proof. Let $\{M_i\}_{i\in\mathbb{N}}$ be a standard enumeration Turing machines and let $time_{M_i}$ be the running times of the machine M_i. Consider the increasing family of polynomials $\{p_k\}_{k\in\mathbb{N}}$, where $p_k(n) = n^k + k$.

The construction of C is the following:

Construction 5.2.3.

Stage 0:

- $y = \varepsilon$*, the empty string;*

Stage $k \geq 1$**:**

- *For all* i*,* $1 \leq i \leq k$ *that have not be eliminated so far test if* $time_{M_i}(y) \leq p_k(|y|)$ *and* $M(y) \neq \chi_A(y)$*. Eliminate all* i *that validate both conditions.*
- *For all* i*,* $1 \leq i \leq k$ *not eliminated test if* $time_{M_i}(y) > p_k(|y|)$*.*

– **If it is true** *then put* y *in* C *and set* y *to be the next string in the lexicographic order and proceed to next stage.*

– **Otherwise** *set* y *to be the next string in the lexicographic order and repeat this stage.*

C is infinite since all stages terminate. If this was not the case, then it would exist a stage k that is reached but does not terminate. Then for sufficiently long strings y, there must exist $i < k$ such that $\mathtt{time}_{M_i}(y) \leq p_k(|y|)$ and $M_i(y) = \chi_A(y)$. Thus, if we dovetail the computations of all machines M_i, where $1 \leq i \leq n$, for which i is never canceled, we obtain an algorithm for χ_A on sufficiently long strings y, which runs in polynomial time. A patch for the shorter strings shows $A \in \mathbf{P}$, contrary to our assumption. □

In order to study the frequency of intractable problems Meyer and Peterson [MP79], defined a class of problems that are almost everywhere polynomial, i.e., the class of decision problems for which only a sparse set of the inputs are allowed to run in non-polynomial time.

Definition 5.2.4. *Let* A *be a set. Consider the function* $\mathtt{dens}_A : \mathbb{N} \to \mathbb{N}$ *defined by* $\mathtt{dens}_A(n) = \#\{x \in A : |x| \leq n\}$. *The set* A *is called* sparse *if for some polynomial* p, $\mathtt{dens}_A(n) \leq p(n)$. *A set is* co-sparse *if its complement is sparse. We say that* A *has* exponential density *if, for some* $\varepsilon > 0$ *and infinitely many* n, $\mathtt{dens}_A(n) \geq 2^{\varepsilon n}$.

Definition 5.2.5. *(***APT***) The class of almost polynomial time languages is:*

$$\mathbf{APT} = \{L(M) : H(M, p) \text{ is sparse for some polynomial } p\}.$$

Later, Orponen and Schöning [OS84] proved an interesting connection of this class with polynomial complexity cores. They proved that **APT** identifies exactly the set of recursive languages that have a proper non-sparse polynomial complexity core. Formally,

Theorem 5.2.6. *A recursive set* A *has a non-sparse polynomial proper complexity core if and only if* $A \notin \mathbf{APT}$.

The idea is similar to the one used by Lynch for proving Theorem 5.2.2.

5.2.2 Average Case Complexity

The analysis of the efficiency of an algorithm is typically based on a worst case scenario of the running time. This analysis is important since this information might be needed to

ensure that the algorithm would always finish on a specific time. However many common algorithms are bad in terms of worst case complexity, they perform well in practice. In middle eighties, Levin [Lev86], proposed a different measure of complexity based on the average of the performance of the algorithm over all inputs.

Definition 5.2.7 (Average case complexity)**.** *Let* μ *be a probability distribution. A function* $f : \Sigma^* \to \mathbb{N}$ *is polynomial on* μ*-average if there is an* $\epsilon > 0$ *such that*

$$\sum_x \frac{f(x)^\epsilon}{|x|}\mu(x) < \infty.$$

In [Sch90] the author, using Markov's inequality and classical algebraic manipulations, proved that a function f is polynomial on μ-average in Levin's sense if and only if there is a polynomial $p : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_0^+$, such that for all $m > 0$, $Pr_\mu[f(x) > p(|x|, m)] \leq \frac{1}{m}$. Observe that the definition takes into account that the average case measure depends not only on the given instance x but also on the probability $\mu(x)$ of the occurrence of x. If the instance x does not appear, i. e. $\mu(x) = 0$, it has no effect on the average case analysis. On the other hand, if for some string x, $\mu(x) > 0$ then $f(x) < p(|x|, \lceil 1/\mu(x) \rceil)$.

A result that will be used later was proved in [AFV03] and gives a connection between average running time of a Turing machine and computational depth of each individual instance for that Turing machine:

Theorem 5.2.8 (Antunes *et al.* [AFV03])**.** *Let* T *and* t *be two constructible time bounds. The following conditions are equivalent:*

1. $T(x) \in 2^{O(\mathrm{depth}^t(x)+\log|x|)}$.
2. $T(\cdot)$ *is polynomial on* $\mathbf{m}^t$*-average.*

Proof. $(1 \to 2)$ Let $T(x) \in 2^{O(\mathrm{depth}^t(x)+\log|x|)}$. Note that $2^{c\log n} = n^c$ is polynomial on $\mathbf{m}^t$-average. Then, by the closure property of the polynomial on $\mathbf{m}^t$-average functions, all we have to do is to prove that $2^{\mathrm{depth}^t(x)}$ is polynomial on $\mathbf{m}^t$-average.

$$\sum_x \frac{2^{\mathrm{depth}^t(x)}}{|x|}\mathbf{m}^t(x) \;=\; \sum_x \frac{2^{K^t(x)-K(x)}}{|x|}2^{-K^t(x)} = \sum_x \frac{2^{-K(x)}}{|x|} \leq \sum_x 2^{-K(x)} < \infty$$

The last inequality follows from Kraft's inequality (see Theorem 2.3.10).

$(2 \to 1)$ Let $T(x)$ be a time constructible function which is polynomial on $\mathbf{m}^t$-average. Then for some $\varepsilon > 0$ we have

$$\sum_x \frac{T(x)^\varepsilon}{|x|}\mathbf{m}^t(x) < 1$$

Define $S_{i,j,n} = \{x \in \Sigma^n : 2^i \leq T(x) < 2^{i+1} \wedge K^t(x) = j\}$. Let 2^r be the approximate size of $S_{i,j,n}$. Then the Kolmogorov complexity of elements in $S_{i,j,n}$ is r, up to an additive $O(\log n)$ factor.

Consider the above sum restricted to elements in $S_{i,j,n}$. We have

$$\sum_{x \in S_{i,j,n}} \frac{T(x)^\varepsilon}{|x|} \mathbf{m}^t(x) < 1$$

$T(x) \geq 2^i$, $\mathbf{m}^t(x) = 2^{-j}$ and there are at least 2^r elements in the above sum. Hence the above sum is lower-bounded by the expression $\dfrac{2^r \times 2^{i\varepsilon} \times 2^{-j}}{|x|^c}$ for some constant c. This gives us

$$1 \;>\; \sum_{x \in S_{i,j,n}} \frac{T(x)^\varepsilon}{|x|} \mathbf{m}^t(x) \geq \frac{2^r \times 2^{i\varepsilon} \times 2^{-j}}{|x|^c} = 2^{i\varepsilon + r - j - c \log n}$$

Then $i\varepsilon + r - j - c \log n < 0 < 1$ and then there is a constant d, such that for all $x \in S_{i,j,n}$, $i\varepsilon \leq \mathrm{depth}^t(x) + d \log |x|$. Then we conclude that $T_M(x) \in 2^{O(\mathrm{depth}^t(x) + \log |x|)}$. □

The above Theorem has an interesting connection to a result due to Li and Vitányi [LV92] relating the average case complexity and the worst-case complexity. They proved that when the inputs to any algorithm are distributed accordingly to universal distribution, the algorithm's average case complexity is of the same order of magnitude as its worst case complexity. Rephrasing this result in the setting of average polynomial time we can make the following statement.

Theorem 5.2.9 (Li-Vitányi [LV92])**.** *Let T be some constructible time bound. The following conditions are equivalent:*

1. *$T(x)$ is bounded by a polynomial in $|x|$.*
2. *$T(\cdot)$ is polynomial on $\mathbf{m}$-average.*

5.2.3 The class FULL-P/log and the class P/poly

Karp and Lipton [KL80] initiated the study of advice functions as a tool to provide a connection between uniform models of computation and non-uniform ones. The idea is to give the program for each string length access to a string that "helps" to compute some language and thus incorporating the non-uniformity in the "advice". The two most natural functions to be considered for the length of the advice words are polynomials and logarithms. The class **P/poly** has been widely studied in the literature (see, for example [Adl78, BFNW93, KL82, Ko82]).

In this dissertation we also use a class that derives from **P/log**. It is known that the class **P/log** is not closed under polynomial time Turing reductions (see, for example, [BH98, BHM92, Ko87]). To have a more robust complexity class, Ko [Ko87] introduced the class **FULL-P/log** . The robustness derives from the fact that the advice not only helps to decide strings of a fixed length, but it is also required that it decides all strings of length smaller or equal to a fixed length. Formally,

Definition 5.2.10. *A set* A *is in the class* **FULL-P/log** *if:*

$$(\forall n)(\exists w : |w| \leq c \log n)(\forall x : |x| \leq n) : x \in A \Leftrightarrow \langle x, w\rangle \in B$$

where $B \in \mathbf{P}$ *and* c *is a constant.*

In the literature it is possible to find several results relating this complexity class with $\mathbf{K}[\log, poly]$[1] and with Tally sets (see [BHM92, BH98]).

We use this complexity class to determine the class of recursive language that are deep.

5.3 Kolmogorov complexity and complexity cores

We show that if the computational depth of the characteristic sequence of a set is larger than a constant almost everywhere then it has a polynomial complexity core, and if the computational depth is larger than a polynomial almost everywhere then it has a proper polynomial complexity core. All the results could be stated using the time bounded Kolmogorov complexity instead of computational depth. However as we would like to remove the condition of the sets being recursive and establish a bridge between the previous chapter and complexity cores, we prefer to use computational depth.

We assume that all time bounds t considered in this section are polynomial on the logarithm of the size of the string. Notice that by using this convention we are considering sublinear time bounds, i.e., when analyzing $\chi_{A_{[1:N]}}$ with $N = 2^{n+1} - 1$ we are only using time polynomial in n.

Following the work of Juedes and Lutz [JL92, JL93] we begin by proving that a set A with polynomial complexity cores with low density can not be complex with respect to time bounded Kolmogorov complexity.

[1] $\mathbf{K}[\log, poly]$ is the class of languages A for which there is a constant c and a polynomial p such that $K^p(x) \leq c \log |x|$ for all $x \in A$

Proposition 5.3.1. *Let A be a set, $\varepsilon > 0$ a constant and $g : \mathbb{N} \to [0, \infty[$ a function. If every polynomial complexity core C of A satisfies $dens_C(n) \leq 2^n - g(n)$ infinitely often then $K^f(A^{=n}) < 2^n - n^{-\varepsilon} g(n) + O(\log n)$ infinitely often, where $f(n) = 2^n \cdot poly(n)$ for some polynomial $poly$.*

Proof. Let p be a fixed polynomial and set $k = \lceil 1/\varepsilon \rceil$. Consider M_1, M_2, ..., a standard enumeration of the deterministic Turing machines that decide A. For each m define the following sets:

$$H_m = \{x : time_{M_m}(x) \leq p(|x|)\}$$

$$B_m = H_m - \Sigma^{\leq m^k}$$

$$B = \bigcup_m B_m$$

$$C = \Sigma^* - B$$

It is easy to see that $H_m \cap C = H_m - B \subset H_m - B_m \subset \Sigma^{\leq m^k}$.

Thus, $H_m \cap C$ is finite and so C is a polynomial complexity core. Define the set $S = \{n : \#C^{=n} \leq 2^n - g(n)\}$. It follows that $S = \{n : \#B^{=n} \geq g(n)\}$. Hence, for each $n \in S$,

$$\begin{aligned} g(n) &\leq \#B^{=n} = \# \left(\bigcup_m B_m \right)^{=n} \\ &\leq \sum_m \#B_m^{=n} = \sum_{m^k < n} \#(B_m)^{=n} \\ &\leq \sum_{0<m\leq n^\varepsilon} \#(B_m)^{=n} \leq \sum_{0<m<n^\varepsilon} \#(H_m)^{=n} \end{aligned}$$

So, there is m such that $0 < m \leq n^\varepsilon$ and $\#(H_m)^{=n} > n^{-\varepsilon} g(n)$. Let $\beta(m)$ be a binary representation of m and w_1, w_2, ..., be the lexicographic ordering of Σ^*. Consider M as the Turing machine implementing the following algorithm:

Algorithm 5.3.2. *Let y be a finite string.*

For $i = 0$ to 2^n do:

1. *Simulate $M_m(w_i)$ for at most $p(n)$ steps.*
2. *If $M_m(w_i) = 1$ or $M_m(w_i) = 0$ then set $z = 0$ or $z = 1$ respectively.*
3. *Otherwise set $z_i = y_1$ and $y = y_2 y_3 ....$*

Output z.

It is easy to check that for all $x \in \Sigma^n$, $\mathtt{time}_M(x) \leq 2^n \times \mathtt{poly}(n)$ for some polynomial $\mathtt{poly}$. So if, for each $n \in \mathbb{N}$, we choose $m \in \mathbb{N}$ and $y \in \Sigma^*$ such that $0 \leq m < n^\varepsilon$, $\#H_m^{=n} \geq n^{-\varepsilon} g(n)$ and y is the successive $2^n - \#H_m^{=n}$ bits consisting of $\chi_A(w_i)$ with $w_i \in \Sigma^n - H_m$, the output of M is exactly $A^{=m}$. Setting $f(n) = 2^n \cdot \mathtt{poly}(n)$, by the assumptions of the Theorem we have, infinitely often, that:

$$\begin{aligned} K^f(A^{=n}|n) &\leq |\langle \beta(m), y\rangle| + O(1) \\ &\leq |y| + 2\beta(m) + O(1) \\ &\leq 2^n - \#H_m^{=n} + O(\log n) \\ &\leq 2^n - n^{-\varepsilon} g(n) + O(\log n) \end{aligned}$$

□

Lemma 5.3.3. *For every* $\leq_m^p$*-hard language* H *for* $\mathbf{E}$*, there are sets* $B, D \in \mathbf{DTIME}(2^n)$ *such that* D *has exponential density and* $D \cap H = B$.

Proof. Notice that the construction due to Meyer [Mey77] and reported in [BH77] used in [JL92] (Theorem 12) also works for time bounds instead of space bounds. In fact there is a set $A \in \mathbf{DTIME}(2^n)$ such that, any $\leq_m^p$ reduction is incompressible i.e., for every polynomial reduction f, $\{x : (\exists y < x) f(x) = f(y)\}$ is finite.

Let H be $\leq_m^p$-hard for $\mathbf{E}$ and let f be a $\leq_m^p$-reduction from A to H. Let $B = \{y : \exists x \in A, y = f(x) \text{ and } |y| \geq |x|\}$ and $D = \{y : \exists x \in \Sigma^*, y = f(x) \text{ and } |y| \geq |x|\}$. Notice that since f is polynomial time computable and $A \in \mathbf{E}$, then $B, D \in \mathbf{E}$.

To prove that D is dense, let p be some polynomial such that $|f(x)| \leq p(|x|)$ and let $\varepsilon > 0$ be a fixed real number such that $p(n^{2\varepsilon}) \leq n$ almost everywhere. Define $W = \{x : |f(x)| < |x|\}$ and for sufficiently large n define $m = \lfloor n^{2\varepsilon} \rfloor$. Thus:

$$\begin{aligned} f(\Sigma^{\leq m}) - \Sigma^{\leq m} &\subset f(\Sigma^{\leq m}) - f(W^{\leq m}) \\ &\subset \{f(x) : x \in \Sigma^{\leq m} \text{ and } |f(x)| \geq |x|\} \\ &\subset D^{\leq p(m)} \\ &\subset D^{\leq n} \end{aligned}$$

Thus

$$\begin{aligned} |D^n| &\geq |f(\Sigma^{\leq m})| - |\Sigma^{<m}| \\ &\geq |\Sigma^{\leq m}| - |\{x|(\exists y < x) f(x) = f(y)\}| - |\Sigma^{<m}| \\ &= 2^m - |\{x|(\exists y < x) f(x) = f(y)| \end{aligned}$$

Since the set $\{x|(\exists y < x) f(x) = f(y)\}$ is finite, for sufficiently large n, we have: $|D^{\leq n}| \geq 2^{n^\varepsilon}$, so D is exponentially dense.

Notice that $B = \{y : \exists x \in A, y = f(x) \text{ and } |y| \geq |x|\} = \{y : \exists x \in f^{-1}(H), y = f(x) \text{ and } |y| \geq |x|\} = H \cap \{y : \exists x \in \Sigma^*, y = f(x) \text{ and } |y| \geq |x|\} = H \cap D$. □

Lemma 5.3.4. *Any* $\mathbf{DTIME}(2^n)$*-core of every* $\leq_m^p$ *hard language* H *for* $\mathbf{E}$ *has a exponential dense complement.*

Proof. Let C denote the $\mathbf{DTIME}(2^n)$-core of H and consider B and D as in the previous lemma. Since $D \in \mathbf{DTIME}(2^n)$, by definition of complexity core, $C \cap D$ is finite. Since D is exponentially dense it follows that $D - C$ is exponentially dense and thus the complement of C is exponentially dense. □

Theorem 5.3.5. *For every* $\leq_m^p$ *hard language* H *for* $\mathbf{E}$*, there is a* $\varepsilon > 0$ *such that* $K^f(H^{=n}) \leq 2^n - n^{-\varepsilon}2^{\varepsilon n} + O(\log n)$*, where* $f(n) = 2^n \cdot \mathtt{poly}(n)$ *for some polynomial* $\mathtt{poly}$*.*

Proof. If H is $\leq_m^p$ hard, by last Lemma, there is $\varepsilon > 0$ such that every $\mathbf{DTIME}(2^n)$-core C of H satisfies $\mathtt{dens}_C(n) < 2^n - 2^{n^\varepsilon}$. Thus by Proposition 5.3.1 it follows, for $f(n) = 2^n \cdot \mathtt{poly}(n)$ for some polynomial $\mathtt{poly}$, that $K^f(H^{=n}) \leq 2^n - n^{-\varepsilon}2^{\varepsilon n} + O(\log n)$. □

Theorem 5.3.6. *Let* t *be a fixed polylogarithmic function. A recursive set* A *is not in* $\mathbf{P}$ *if and only if for all constant* c*, for infinitely many* n *and* $N = 2^{n+1} - 1$*,* $\mathrm{depth}^t(\chi_{A_{[1:N]}}|n) > c$*.*

Proof. Assume that there is a polynomial t and a constant $c > 0$ such that, for all n, $\mathrm{depth}^t(\chi_{A_{[1:N]}}|n) < c$. Since we have assumed that A is recursive, this means that for all n, $K(\chi_{A_{[1:N]}}|n) < c$. So, if $\mathrm{depth}^t(\chi_{A_{[1:N]}}|n) < c$ this implies $K^t(\chi_{A_{[1:N]}}|n) < c$. Let I be the set of all pairs (p, r) such that $|p| + |r| \leq c$ and (p, r) is the least pair in the lexicographic order corresponding to $K^t(\chi_{A_{[1:N]}}|n)$ for some n. Notice that:

1. $\#I \leq 2^c$;
2. If $N_1 < N_2$ and (p_1, r_1) and (p_2, r_2) correspond to $K^t(\chi_{A_{[1:N_1]}}|n_1)$ and $K^t(\chi_{A_{[1:N_2]}}|n_2)$ then, by definition of K^t, $|p_1| + |r_1| \leq |p_2| + |r_2|$. Also, for all $x \in \Sigma^{n-1}$, if $U^{r_2}(p_2, x)$ stops in time $t(N_1)$ then $U^{r_2}(p_2, x) = U^{r_1}(p_1, x) = \chi_A(x)$.

Consider now the following program p: on input x of length n, p runs each pair (p_i, r_i) in lexicographic order for t steps and outputs the same answer which the largest program in the lexicographic order outputted. From the construction, the running time of p is at most $2^c t(N) + O(1)$, which is polynomial on n. The correctness of the result follows from item 2 above. Thus $A \in \mathbf{P}$.

To prove the converse, assume that $A \in \mathbf{P}$, so there is an algorithm g that, given x, produces $\chi_A(x)$ in polynomial time on the length of x. Since the algorithm does not depend on x, its length is a constant and then $\mathrm{depth}^t(\chi_{A_{[1:N]}}|n) < c$. □

Notice that, by Lynch's characterization of complexity cores (Theorem 5.2.2) the previous result establishes a connection between recursive sets having a polynomial complexity core and its computational depth.

Corollary 5.3.7. *Let* t *be a fixed polylogarithmic function. A recursive set* A *has a polynomial complexity core if for all constant* c *and, for infinitely many* n, $\text{depth}^t(\chi_{A_{[1:N]}}|n) > c$, *where* $N = 2^{n+1} - 1$.

Proposition 5.3.8. *Let* t *be a fixed polylogarithmic function. The set* A *is recursive not belonging to* $\mathbf{P/poly}$ *if and only if for all polynomial* p *and for infinitely many* n, $\text{depth}^t(\chi_{A_{[1:N]}}|n) > p(n)$, *where* $N = 2^{n+1} - 1$.

Proof. Assume, by contradiction, that there is a polynomial q such that there exists an n for which $\text{depth}^t(\chi_{A_{[1:N]}}|n) \leq q(n)$ onwards. Since A is recursive this means that $K^t(\chi_{A_{[1:N]}}|n) \leq q(n)$. Thus, there is p and r such that $|p|+|r| \leq q(n)$ such that $U^r(p, x) = \chi_A(x)$ in polynomial time for all $x \in \Sigma^{\leq n}$. So, giving p and r as advice we can decide $A^{\leq n}$, so A belongs to $\mathbf{P/poly}$.

To prove the converse, assume that A is a recursive set belonging to $\mathbf{P/poly}$. Then there is a polynomial q such that, given n, there are p and r such that $|p|+|r| \leq q(n)$ and $U^r(p, x) = \chi_A(x)$ in polynomial time for all $x \in \Sigma^{\leq n}$. Thus, for all n, $K^t(\chi_{A_{[1:N]}}|n) \leq q(n)$ and then $\text{depth}^t(\chi_{A_{[1:N]}}|n) \leq q(n)$. □

In [Sch86], the author proved that $\mathbf{APT} \subsetneq \mathbf{P/poly}$. Since all recursive sets not in $\mathbf{APT}$ have a proper complexity core (see Theorem 5.2.6), as a consequence of the last result, we conclude that:

Corollary 5.3.9. *Let* t *be a fixed polylogarithmic function. A recursive set* A *has a proper complexity core if for all polynomial* p *and for infinitely many* n, $\text{depth}^t(\chi_{A_{[1:N]}}|n) > p(n)$ *where* $N = 2^{n+1} - 1$.

An similar result to the last two Theorems establishes that all recursive sets that are not in $\mathbf{FULL\text{-}P/log}$ have logarithmic depth. It is unknown an inclusion relationship between $\mathbf{APT}$ and $\mathbf{FULL\text{-}P/log}$. Notice that if $\mathbf{APT} \subset \mathbf{FULL\text{-}P/log}$ then this result would give a sharper characterization of sets admitting a proper complexity core.

Proposition 5.3.10. *Let* t *be a fixed polylogarithmic function. A set* A *is a recursive set not in* $\mathbf{FULL\text{-}P/log}$ *if and only if there is a constant* c *such that, for infinitely many* n, $\text{depth}^t(\chi_{A_{[1:N]}}|n) > c \log n$, *where* $N = 2^{n+1} - 1$.

The proof of this result is similar to the proof of Proposition 5.3.8.

We now characterize complexity cores based on computational depth:

Definition 5.3.11 (Kolmogorov complexity core)**.** *We say that a set* A *has a* Kolmogorov complexity core *for the uniform class* $\mathcal{C}$ *if for all* $c \in \mathbb{N}$ *and for all* $f \in \mathcal{C}$ *and for infinitely many* n*,*

$$\text{depth}^{\log^k(f)}(\chi_{A_{[1:N]}}|n) > c.$$

In order to better understand the previous definition we apply it within the **EXP** class.

Theorem 5.3.12. *Let* A *be a recursive set.* A *has a complexity core relatively to the class* **EXP** *if and only if for all constant* c*, for all sufficiently large function* $f \in \mathbf{EXP}$ *and infinitely many* n*,* $\text{depth}^{\log^k(f)}(\chi_{A_{[1:N]}}|n) > c$*, where* $N = 2^{n+1} - 1$*.*

Proof. Assume that for some $f \in \mathbf{EXP}$, $\text{depth}^{\log^k(f)}(\chi_{A_{[1:N]}}|n) \leq O(1)$ for all n. Then, since A is recursive this implies that $K^f(\chi_{A_{[1:N]}}|n) \leq O(1)$. Thus, since there are a finite number of possible programs for all n, we can construct one[2] deciding the problem the membership in A for all $x \in \Sigma^*$ in time f, i.e., there is an algorithm G such that $G(x) = \chi_A(x)$ and $|G| \leq c$, where c is some constant. Thus $A \in \mathbf{EXP}$ which implies that A has not a complexity core with respect to the class **EXP**.

On the other hand, if $A \in \mathbf{EXP}$ then there is an algorithm G such that $G(x) = \chi_A(x)$ for all $x \in \Sigma^*$ in an exponential number of steps. Since G does not depend on x, its length is a constant. So, $\text{depth}^{\log^k(f)}(\chi_{A_{[1:N]}}|n) \leq O(1)$, where $N = 2^{n+1} - 1$. □

We already proved that if A is recursive and admits a Kolmogorov complexity core for the polynomial time class then it satisfies the condition on Definition 5.2.1. In the next result we show how to construct a complexity core.

Proposition 5.3.13. *Let* $k > 0$ *be a fixed integer. If* A *is a recursive set and for all* $c \in \mathbb{N}$ *and for all polynomial* p *and for almost all* $n \in \mathbb{N}$*,* $\text{depth}^{\log^k}(\chi_{A_{[1:N]}}|n) > c\log(n)$*, where* $N = 2^{n+1} - 1$ *then* A *has a polynomial complexity core.*

Proof. We construct a polynomial complexity core inductively.

Let n_0 be the first index such that $\text{depth}^{\log^k}(\chi_{A_{[1:N]}}|n) > c\log(n)$ for all $n \geq n_0$.

Since A is recursive, we know that $K(\chi_{A_{[1:N]}}|n) = O(1)$, so $\text{depth}^{\log^k}(\chi_{A_{[1:N]}}|n) > c\log(n)$ is equivalent to $K^{\log^k}(\chi_{A_{[1:N]}}|n) > c\log(n)$.

[2]Similar to the one constructed in Theorem 5.3.6.

If for all c, $K^{\log^k}(\chi_{A_{[1:N]}}|n) > c\log(n)$ then there is no p and r such that $|p| + |r| \leq c\log(n)$ and for all $1 \leq i \leq N$, $U^r(p,i) = \chi_{A_{[i,i]}}$ in time $\log^k(N) = \mathtt{poly}(n)$.

So, there is at least one i, $1 \leq i \leq N$ such that for all p and r satisfying $|p|+|r| \leq c\log n$ either $U^r(p,i) \neq \chi_{A_{[i,i]}}$ or $U^r(p,i) = \chi_{A_{[i,i]}}$ but $\mathtt{time}_p(i) > \mathtt{poly}(n)$. Since this is true for all c, there is x of size at most n that can not be decided by programs of length at most $O(\log n)$ in polynomial time. Such x is an element of the core of A.

To construct the next element in the core we consider the characteristic sequence up to strings of size $n_1 = 2^{2^{n_0+1}}$, i.e., $N = 2^{n_1+1} - 1$. Notice that now we allow programs and oracles up to size $\log n_1 = 2^{n_0+1}$. In particular, we can allow the oracle to be the characteristic sequence of A up to strings of length n_0; considering the program printing the i^{th} bit of r we conclude that in the alloted time, $\mathtt{poly}(n_1)$ we can decide x. But, by assumption, there is $i_1 \neq i$ with $1 \leq i_1 \leq N_1$ such that no program and oracle of size at most 2^{n_0+1} (logarithmic on the size of the strings we are considering) can decide it in time $\mathtt{poly}(n_1)$. The x_1 corresponding to the i^{th} element in the lexicographic order is the next element in the complexity core.

Notice that we can go on forever with this argument and construct an infinite polynomial complexity core although this will take a long time. Also, a program that decides A has, in particular, to decide all strings up to a certain size. □

5.4 Complexity cores and average case complexity

We now relate the existence of polynomial complexity cores with the universal distribution $\mathbf{m}$. From Theorem 5.2.9 we have:

Theorem 5.4.1. *Let A be a recursive language such that $A \notin \mathbf{P}$. If M is a Turing machine deciding A, then M is not polynomial on $\mathbf{m}$-average.*

Proof. Since A is a recursive set not in $\mathbf{P}$ then given a Turing machine M that decides A and any polynomial p there is an n and a string $x \in \Sigma^n$ such that $\mathtt{time}_M(x) > p(n)$. Then, by the characterization of polynomial average case of Theorem 5.2.9, M can not run in polynomial time on $\mathbf{m}$-average. □

We can go further and prove that in fact A is not $\mathbf{m}^t$ polynomial on average where t is any super polynomial time constructible function.

Theorem 5.4.2. *If A is recursive and not in $\mathbf{P}$, then A is not $\mathbf{m}^t$ polynomial on average where $t(n) = n \cdot t'(n)\log(n \cdot t'(n))$ and t' is any sufficiently large super polynomial time constructible function.*

Proof. If A is recursive and is not in $\mathcal{P}$ then it admits a polynomial complexity core H that is recognizable in time t, where t is any super polynomial time constructible function, as proved in [OS86]. Let $dens_H(\cdot)$ be the density function of H. Consider the following probability distribution over Σ^*:

$$\mu(x) = \begin{cases} \dfrac{1}{dens_H(n) \cdot 2^n} & \text{if } x \in H^n \\ \\ 0 & \text{otherwise} \end{cases}$$

By definition of polynomial complexity core we know that, for all polynomial p almost all $x \in H$ satisfies $time_M(x) > p(|x|)$. Thus, there is n_p such that M can not decide any element of H of length larger than n_p. Thus,

$$Pr_\mu[time_M(x) \geq p(|x|)] \geq 1 - \frac{1}{dens_H(n_p)}.$$

Hence, according to the equivalence of polynomial time average complexity proved in [Sch90], $time_M$ is not μ-polynomial on average. Notice that μ is computable in some super polynomial time t_1. So, by Theorem 2.3.20, $\mathbf{m}^{t'}$, where $t'(n) = n \cdot t_1(n) \log(n \cdot t_1(n))$ dominates μ and we conclude that for any sufficiently large super polynomial time constructible function t', A is not polynomial on $\mathbf{m}^{t'}$ average. □

If the polynomial complexity core has exponential density we can improve the last result by using the distribution $\mathbf{m}^p$, where p is a polynomial. Notice that $\mathbf{m}^p$ assigns a smaller probability to strings than $\mathbf{m}^t$ where t is some super polynomial time bound and thus the result is not a corollary of the previous results.

Proposition 5.4.3. *If a set A admits a exponential density polynomial complexity core then A is not $\mathbf{m}^{poly}$ polynomial on average.*

Proof. If A admits a polynomial complexity core of exponential density then given any Turing machine M that decides A, there is a $\varepsilon > 0$ such for all polynomial p and sufficiently large n

$$\#H^{=n} = \#\{x \in \Sigma^n : time_M(x) > p(n)\} > 2^{\varepsilon n}$$

Let q be a fixed polynomial time bound and consider the set:

$$D = \{x \in \Sigma^* : depth^q(x) \leq c \log n\}$$

In [AFvMV06] it is shown that there are approximately $2^{\varepsilon' n}$ strings of length n such that $\text{depth}^{2^n}(x) \geq (1-\varepsilon')n - c \log n$. Thus $D^{=n} \approx 2^n - 2^{\varepsilon' n}$. So, if $\varepsilon > \varepsilon'$, by a Pigeon-hole principle, $D^{=n} \cap H^{=n} \neq \emptyset$. In fact:

$$\#(D^{=n} \cap H^{=n}) \geq 2^n - 2^{\varepsilon' n} + 2^{\varepsilon n} - 2^n = 2^{\varepsilon n} - 2^{\varepsilon' n} > 0$$

For any $x \in D^{=n} \cap H^{=n}$ we know that $O(2^{\text{depth}^q(x)+\log|x|})$ is polynomial in the length of x and $\mathtt{time}_M(x) > q(|x|)$. Since this works for any polynomial q we conclude by Theorem 5.2.8 that $\mathtt{time}_M$ is not polynomial time on average with respect to $\mathbf{m}^q$. □

Corollary 5.4.4. *If* **NP** *does not have measure* 0 *in* **EXP** *then every* **NP***-hard language, with respect to polynomial time many to one reductions, is not in* $\mathbf{m}^{\mathtt{poly}}$ *polynomial on average.*

Proof. In [JL93] it is proved that if **NP** does not have measure 0 in **EXP** then any **NP** hard language has a polynomial complexity core of exponential density (see Corollary 4.10 in [JL93]). So, by the previous Proposition the result follows. □

Chapter 6

Conclusions and future work

Computational complexity and Kolmogorov complexity are different kind of measures. While the first studies the difficulty of of solving all instances of a problem the latter has its focus on the difficulty of describing individual instances. Nevertheless it is interesting to study the interplay between these two measures.

In this thesis we continued to explore the applications of Kolmogorov complexity and more precisely, the computational depth (difference between time bounded and unrestricted versions of Kolmogorov complexity) as a tool for computational complexity.

We applied computational depth in complexity theory to two directions:

1. Comparing it with sophistication;

2. Using it to derive a characterization on complexity classes and boolean formulas for which we can find in probabilistic polynomial time satisfying assignments.

For the first item, we revisited the notion of sophistication for sequences. We proposed new definitions of this concept based on the definition of sophistication for strings proposed in [AF09a] with a similar flavor of [Lut03, May02]. We proved that the set of sequences with lower sophistication equal to 0 has measure 1 and the set of sequences with upper sophistication equal to 1 is dense. We showed a connection with classical notions of dimension, namely constructive Hausdorff and Packing dimension, and aproved that computational depth and sophistication are measures of different kind.

For the second item we firstly studied the relationship of computational depth with the satisfiability problem. We proved that if a formula has an attribution of depth d then we can use a probabilistic time algorithm running in time exponential in d to find another true assignment for that formula. We also showed that if exponential time is not infinitely

often in subexponencial space, then the converse holds, i.e., if a probabilistic algorithm finds a valid assignment for a Boolean formula ϕ in time t, there is a witness w for ϕ such that $\text{depth}^t(w|\phi) \leq O(\log n)$. On the other hand if $\mathbf{BPP} = \mathbf{FewP} = \mathbf{EXP}$, one could find formulas that have a single solution of high computational depth that could be found quickly using a probabilistic algorithm.

We looked at the question of whether one can increase the depth of a string efficiently and showed that under the assumption of the existence of good psudorandom generators one can not significantly increase the depth of a string in polynomial time, while if $\mathbf{BPP} = \mathbf{EXP}$ there are strings of high depth produces from string of very low depth.

Following this idea we asked if one could characterize the complexity core of **SAT** with computational depth. This was not achieved, yet we established a connection between computational depth of languages and the existence of complexity cores. This was done by quantifying the computational depth of sets in the computational classes $\mathbf{P}$, $\mathbf{EXP} = \mathbf{DTIME}(2^{\text{poly}(n)})$, **FULL-P/log** and **P/poly**.

There are many different future research directions one can take from here.

Concerning sophistication and computational depth for sequences, it would be interesting to know if a stronger result relatively to the difference between dimensional depth and lower sophistication holds. One candidate for a sharper relation is the characteristic sequence of the diagonal of Halting problem. These results may be applicable not only in Computer Science but also in Dynamical Systems Theory. Some work has been done in this direction using Kolmogorov complexity, see for example [Bru83, Whi91, Zwe06]. It might be interesting to study if the measures presented in this paper can be used to characterize some of the properties of the dynamics.

The paper [AFPS07] gave some insight for what should be a complexity cores in terms of Kolmogorov complexity. The paper [Sou10] was the result of those thoughts. In Chapter 5 we had proved some results relating computational depth with existence of complexity cores. It would be interesting to have a pure Kolmogorov complexity characterization of the core itself instead of having the characterization of the languages that have a (proper) complexity cores similar to what is done in [OKSW94] with instance complexity. It also would be interesting to relax the condition of recursive languages to recursive enumerable sets and explore a generalization of the definition of complexity cores using Kolmogorov complexity or any other Kolmogorov complexity based measure.

Another line of research we intend to follow is the use of time bounded Kolmogorov complexity or computational depth in analysis of some cryptographic primitives. Since it is easy to impose time constrains to Kolmogorov complexity it might be suitable to

address some questions regarding one-way functions and zero knowledge protocols. In particular, we are interested to study the notion of individual witness hiding proofs and Kolmogorov one-way functions.

We plan to characterize and study individual witness hiding proofs, based on the knowledge conveyed on individual communications between two parties by measuring the difference between two time bounded Kolmogorov complexities, one when the verifier uses the communication and the other when he does not use it.

Intuitively, the function f is one-way if is easy to evaluate but hard to invert. Thus x and f give all the information needed to compute in polynomial time $f(x)$ and, on the other hand, the value $f(x)$ does not convey, in polynomial time, useful information about x. We conjecture that if f is a one-way function then the length of a shortest program computing x given $f(x)$, $|x|$, and f should be approximately equal to the length of a shortest program computing x without any auxiliary input.

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 2009.

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 75–83, Washington, DC, USA, 1978. IEEE Computer Society.

[AF09a] L. Antunes and L. Fortnow. Sophistication revisited. *Theory of Computing Systems*, 45(1):150–161, Springer–Verlag, 2009.

[AF09b] L. Antunes and L. Fortnow. Worst-case running times for average-case algorithms. *Proceedings of the Conference on Computational Complexity*, pages 298–303, IEEE Computer Society, 2009.

[AFPS07] L. Antunes, L. Fortnow, A. Pinto, and A. Souto. Low-depth witnesses are easy to find. In *Proceedings of the Conference on Computational Complexity*, pages 46–51, Washington, DC, USA, 2007. IEEE Computer Society.

[AFV03] L. Antunes, L. Fortnow, and N. Vinodchandran. Using depth to capture average-case complexity. In *Proceedings of the International Symposium Fundamentals of Computation Theory*, volume 2751 of *Lecture Notes in Computer Science*, pages 303–310, Springer–Verlag, 2003.

[AFvMV06] L. Antunes, L. Fortnow, D. van Melkebeek, and N. Vinodchandran. Computational depth: concept and applications. *Theoretical Computer Science*, 354(3):391–404, Elsevier Science Publishers Ltd., 2006.

[AHLM07] K. Athreya, J. Hitchcock, J. Lutz, and E. Mayordomo. Effective strong dimension in algorithmic information and computational complexity. *SIAM Journal on Computing*, 37(3):671–705, Society for Industrial and Applied Mathematics, 2007.

[AMSV09] L. Antunes, A. Matos, A. Souto, and P. Vitányi. Depth as randomness deficiency. *Theory of Computing Systems*, 45(4):724–739, Springer–Verlag, 2009.

[Ant02] L. Antunes. *Useful information.* PhD thesis, Faculdade de Ciências da Universidade do Porto, 2002.

[AS08] L. Antunes and A. Souto. Sophisticated infinite sequences. In *Proceedings of the Computability in Europe 2008*, pages 25–34, Springer–Verlag, Athens, Greece, 2008.

[AS10] L. Antunes and A. Souto. Information measures for infinite sequences. *Theoretical Computer Science*, 41(26-28):2602–2611, Elsevier Science Publishers Ltd., 2010.

[AST10] L. Antunes, A. Souto, and A. Teixeira. A characterization of one-way functions based on time-bounded komogorov complexity-extended abstract. In *Proceedings of the Logical Approaches to Barriers in Computing and Complexity*, volume 4, pages 3–5. Greiwswald University, Germany, 2010.

[BDG95] J. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity 1.* Springer-Verlag, Berlin, 1995.

[Ben88] C. Bennett. Logical depth and physical complexity. In *A half-century survey on The Universal Turing Machine*, pages 227–257, New York, NY, USA, 1988. Oxford University Press, Inc.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, Birkhauser Verlag, 1993.

[BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, Society for Industrial and Applied Mathematics, 1977.

[BH98] J. Balcázar and M. Hermo. The structure of logarithmic advice complexity classes. *Theoretical Computer Science*, 207(1):217–244, Elsevier Science Publishers Ltd., 1998.

[BHM92] J. Balcázar, M. Hermo, and E. Mayordomo. Characterizations of logarithmic advice complexity classes. In *Proceedings of the World Computer Congress on Algorithms, Software, Architecture - Information Processing*, pages 315–321. North-Holland Publishing Co., 1992.

[Bru83] A. Brudno. Entropy and the complexity of the trajectories of a dynamical system. *Transactions of the Moscow Mathematical Society*, 2:127–151, Moskovskoe matematicheskoe obshchestvo, 1983.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of ACM*, 13(4):547–569, ACM Press, 1966.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Symposium on Theory of Computing*, pages 151–158. AMC, 1971.

[FK96] L. Fortnow and M. Kummer. On resource-bounded instance complexity. *Theoretical Computer Science*, 161:123–140, Elsevier Science Publishers Ltd., 1996.

[For89] L. Fortnow. *Complexity-theoretic aspects of interactive proof systems.* PhD thesis, Massachusetts Institute of Technology, 1989.

[Gác93] P. Gács. Lecture notes on descriptional complexity and randomness, 1993.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NPCompleteness.* W.H. Freeman and Company, 1979.

[Gol01] O. Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

[GTV01] P. Gacs, J. Tromp, and P. Vitanyi. Algorithmic statistics. *IEEE Transactions on Information Theory*, 47:2443–2463, IEEE Computer Society, 2001.

[GV04] P. Grunwald and P. Vitanyi. Shannon information and kolmogorov complexity, 2004.

[Hau19] F. Hausdorff. Dimension und äusseres mass. *Mathematics Annuals*, 79:157–179, 1919.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, American Mathematical Society, 1965.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Symposium on Theory of Computing*, pages 12–24. ACM Press, 1989.

[IW96] R. Impagliazzo and A. Wigderson. P=BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma (preliminary version). In *Proceedings of the Symposium on Theory of Computing*, pages 220–229. ACM Press, 1996.

[JL92] D. Juedes and J. Lutz. Kolmogorov complexity, complexity cores, and the distribution of hardness. *Kolmogorov Complexity: Theory and Relations to Computational Complexity*, pages 43–65, Springer–Verlag, 1992.

[JL93] D. Juedes and J. Lutz. The complexity and distribution of hard problems. *Journal on Computing*, 24:177–185, Society for Industrial and Applied Mathematics, 1993.

[KA91] M. Koppel and H. Atlan. An almost machine-independent theory of program-length complexity, sophistication, and induction. *Information Scinces*, 56(1-3):23–33, Elsevier Science Publishers Ltd., 1991.

[Kar72] R. Karp. Reducibility among combinatorial problems. pages 85–103, New York, 1972. Plenum Press.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the Symposium on Theory of Computing*, pages 302–309, New York, NY, USA, 1980. ACM Press.

[KL82] R. Karp and R. Lipton. Turing machines that take advice. In *Logic and Algorithmic, L'Enseignement Mathématique*, 1982.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Information Processing Letters*, 14(1):39–43, Elsevier Science Publishers Ltd., 1982.

[Ko87] K. Ko. On helping by robust oracle machines. *Theoretical Computer Science*, 52(1-2):15–36, Elsevier Science Publishers Ltd., 1987.

[Kol50] A. Kolmogorov. *Foundations of the Theory of Probability.* Chelsea Publishing, 1950.

[Kol65] A. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, Springer–Verlag, 1965.

[Kop87] M. Koppel. Complexity, depth, and sophistication. *Complex Systems*, 1:1087–1091, Complex Systems Publication Inc., 1987.

[Kop95] M. Koppel. Structure. *The universal Turing machine (2nd ed.): a half-century survey*, pages 403–419, Springer–Verlag, 1995.

[Kra49] L. Kraft. *A device for quantizing, grouping and coding amplitude modulated pulses.* Masters thesis, Department of Electrical Engineering, M.I.T., 1949.

[KvM02] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, Society for Industrial and Applied Mathematics, 2002.

[Lap97] S. Laplante. *Kolmogorov Techniques in Computational Complexity Theory.* PhD thesis, University of Chicago, 1997.

[Lev74] L. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Problems Information Transmission*, 10:206–210, Russian Academy of Sciences, 1974.

[Lev86] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, Society for Industrial and Applied Mathematics, 1986.

[LM93] L. Longpré and S. Mocas. Symmetry of information and one-way functions. *Information processing Letters*, 46(2):95–100, Elsevier Science Publishers Ltd., 1993.

[LR05] T. Lee and A. Romashchenko. Resource bounded symmetry of information revisited. *Theoretical Computer Science*, 345(2-3):386–405, Elsevier Science Publishers Ltd., 2005.

[Lut00a] J. Lutz. Dimension in complexity classes. In *Proceedings of the Conference on Computational Complexity*, pages 158, IEEE Computer Society, Washington, DC, USA, 2000.

[Lut00b] J. Lutz. Gales and the constructive dimension of individual sequences. In *Proceedings of the Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 902–913. Springer-Verlag, 2000.

[Lut03] J. Lutz. The dimensions of individual strings and sequences. *Information and Computation*, 187(1):49–79, Academic Press, Inc., 2003.

[LV92] M. Li and P. Vitányi. Average case complexity under the universal distribution equals worst-case complexity. *Information Processing Letters*, 42(3):145–149, Elsevier Science Publishers Ltd., 1992.

[LV08] Ming Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, 2008.

[LW95] L. Longpré and O. Watanabe. On symmetry of information and polynomial time invertibility. *Information and Computation*, 121(1):14–22, Academic Press, Inc., 1995.

[Lyn75] N. Lynch. On reducibility to complex or sparse sets. *Journal of ACM*, 22(3):341–345, ACM Press, 1975.

[May02] E. Mayordomo. A kolmogorov complexity characterization of constructive hausdorff dimension. *Information Processing Letters*, 84(1):1–3, Elsevier Science Publishers Ltd., 2002.

[Mey77] A. Meyer. reported in [BH77]. 1977.

[Mil01] P. Miltersen. Derandomizing complexity classes. *Handbook of Randomized Computing, Kluwer*, 2001.

[ML71] P. Martin-Löf. Complexity oscillations in infinite binary sequences. *Probability Theory and Related Fields*, 19(3):225–230, Springer–Verlag, 1971.

[MP79] A. Meyer and M. Paterson. With what frequency are apparently intractable problems difficult? In *MIT technical report TM-126*. MIT, 1979.

[MTS08] A. Matos, A. Teixeira, and A. Souto. On a relationship between nondeterministic communication complexity and instance complexity. In *Proceedings of the Computability in Europe 2008*, 2008.

[MTS10] A. Matos, A. Teixeira, and A. Souto. On the largest monochromatic combinatorial rectangles with an application to communication complexity. In *Proceedings of the Computability in Europe 2010*, pages 264–273, University of Azores, 2010.

[Mun99] M. Mundhenk. On hard instances. *Theoretical Computer Science*, 242, Elsevier Science Publishers Ltd., 1999.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, Elsevier Science Publishers Ltd., 1994.

[OKSW94] P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of ACM*, 41(1):96–121, ACM Press, 1994.

[OS84] P. Orponen and U. Schöning. The structure of polynomial complexity cores (extended abstract). In *Proceedings of the Mathematical Foundations of Computer Science*, pages 452–458, London, UK, 1984. Springer-Verlag.

[OS86] P. Orponen and U. Schöning. The density and complexity of polynomial cores for intractable sets. *Information and Control*, 70(1):54–68, Academic Press Professional, Inc., 1986.

[Pap85] C. Papadimitriou. *Computational Complexity*. Wiley, 1985.

[Pin07] A. Pinto. *Applications of Kolmogorov Complexity to Cryptography*. PhD thesis, Faculdade de Ciências da Universidade do Porto, 2007.

[PSMA09] A. Pinto, A. Souto, A. Matos, and L. Antunes. Commitment and authentication systems. *Designs, Codes & Cryptography*, 53(3):175–193, Springer–Verlag, 2009.

[SASB10] C. Santos, L. Antunes, A. Souto, and J. Bernardes. Assessment of disagreement: a new information based approach. *Annals of Epidemiology*, 20:555–561, Elsevier Science Publishers Ltd., 2010.

[Sch86] U. Schöning. Complete sets and closeness to complexity classes. *Theory of Computing Systems*, 19(1):29–41, Springer–Verlag, 1986.

[Sch90] R. Schapire. The emerging theory of average case complexity. In *MIT technical report 431*. MIT, 1990.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, Bell labs, 1948.

[Sip97] M. Sipser. *Introduction to the Theory of Computation*. PWS Publisher, 1997.

[Sol64] R. Solomonoff. A formal theory of inductive inference, Part I. *Information and Control*, 7(1):1–22, Academic Press Inc., 1964.

[Sou10] A. Souto. Kolmogorov complexity cores. In *Computability in Europe 2010*, volume 6158 of *Lecture Notes in Computer Science*, pages 376–385. Springer-Verlag, 2010.

[STP10] A. Souto, A. Teixeira, and A. Pinto. One-way functions using komogorov complexity. In *Proceedings of the Computability in Europe*, pages 346–356, University of Azores, 2010.

[Sul84] D. Sullivan. Entropy, hausdorff measures old and new, and limit sets of geometrically finite kleinian groups. *Acta Mathematica*, 153(1):259–277, Springer–Verlag, 1984.

[Tri82] C. Tricot. Two definitions of fractional dimension. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 91, pages 57–74. ACM Press, 1982.

[TSAM10] A. Teixeira, A. Souto, L. Antunes, and A. Matos. Entropy measures vs. algorithmic information. In *Proceedings of the International Symposium on Information Theory*, pages 1413–1417, IEEE Computer Society, June 2010.

[Tur37] A. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, volume 42, pages 230–265. Springer-Verlag, 1937, Correction: 43: 544-546 (1937).

[Vad99] S. Vadhan. *A study of statistical zero-knowledge proofs*. PhD Thesis, Massachusetts Institute of Technology, 1999.

[VV03] N. Vereshchagin and P. Vitanyi. Kolmogorov's structure functions and model selection. *IEEE Transactions on Information Theory*, 50:3265–3290, Computer Society, 2003.

[V'y99] V. V'yugin. Algorithmic complexity and stochastic properties of finite binary sequences. *Computational Journal*, 42(4):294–317, Elsevier Science Publishers Ltd., 1999.

[Whi91] H. White. *On the algorithmic complexity of the trajectories of points in dynamical systems.* PhD Thesis, University of North Carolina at Chapel Hill, 1991.

[YDD04] L. Yu, D. Ding, and R. Downey. The kolmogorov complexity of random reals. *Annals of Pure and Applied Logic*, 129(1-3):163 – 180, Elsevier Science Publishers Ltd., 2004.

[ZL70] A. Zvonkin and L. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 256:83–124, Russian Academy of Sciences, 1970.

[Zwe06] R. Zweimuller. Asymptotic orbit complexity of infinite measure preserving transformations. *Discrete and Continuous Dynamical Systems*, 15:353–366, AIMS, 2006.

Appendix A

Other work

During my research, I also developed other scientific work, collaborating with other researchers. I participated in several research projects in different areas:

- The work developed in the area of medicine with C. Santos, L. Antunes and J. Bernardes [SASB10], where we proposed a new information based measure of agreement between two observers and studied the advantages of the use of this new measure.

- In cryptography, I have worked with A. Pinto, A. Matos and L. Antunes [PSMA09], where we analyzed some cryptographic schemes namely, authentication and commitment. Also, together with L. Antunes, A. Teixeira and A. Pinto in [AST10] and in [STP10], we discussed the classical notion of one-way functions based on Kolmogorov complexity.

- In computational complexity, I also contributed in the papers [AMSV09], [MTS08] and [MTS10]. One of the most gratifying collaboration end up with the paper [TSAM10] at ISIT 2010, where together with A. Texeira, A. Matos and L. Antunes, we studied if the known result of having the Shannon entropy being equal to the expected value of the Kolmogorov complexity also holds when Shannon entropy is replaced by Tsalis and Rényi entropies, two generalizations of the Shannon entropy.

- With H. Burhman, I did some work in quantum computation field, during the visit to CWI in 2007. We studied the advantage of the use of a quantum computer when we play a generalization of the classical Mastermind game with n pegs and k colors, showing upper and lower bounds for its quantum query problem. This work has not been published yet.

- Together with F. Moreira, my Master thesis advisor, during my PhD, I developed some work on Dynamical Systems and Nonstandard Analysis. We investigated a generalization of Birkhoff's Theorem for non-invariant measures and studied a new proof of Oseledets' Theorem, showing the existence of Lyapunov exponents. This is a work in progress, hopefully to be publish it soon.

Assessment of disagreement: a new information based approach

Cristina Costa Santos, MSc[1], Luís Antunes PhD[2], André Souto MSc[2], João Bernardes PhD[3]

1. Biostatistics and Medical Informatics Department – CINTESIS, Faculty of Medicine, University of Porto
2. Instituto de Telecomunicações. Faculty of Sciences, Faculty of Science, University of Porto
3. Obstetrics and Gynaecology Department, Faculty of Medicine, University of Porto

Corresponding author:

Cristina Costa Santos

Biostatistics and Medical Informatics Department,

Faculty of Medicine, University of Porto.

Al. Prof. Hernâni Monteiro

4200-319 Porto, Portugal

Tel: +351 22 551 3622 Fax:+351 22 551 3623 E-mail: csantos@med.up.pt

Running title: A new approach to assess disagreement

Word count: 3839

Abstract word count: 154

Number of tables: 4

ABSTRACT

Purpose

Disagreement on the interpretation of diagnostic tests and clinical decisions remains an important problem in Medicine. As no strategy to assess agreement seems to be fail-safe to compare the degree of agreement, or disagreement, we propose a new information based approach to assess disagreement.

Methods

The sum over all logarithms of possible outcomes of a variable is a valid measure of the amount of information contained in the variable. So the proposed measure is based on the amount of information contained in the differences between two observations. This measure is normalized and satisfies the flowing properties: it is a metric, scaled invariant and it has differential weighting.

Results

Two real examples and a simulation were used to illustrate the usefulness of the proposed measure to compare the degree of disagreement.

Conclusions

Used as complement to the existing methods, our approach can be useful to compare the degree of disagreement among different populations.

MeSH heading key words: Observer Variation; Information Theory; Reproducibility of Results

Introduction

Measurement is essential both for clinical care and for epidemiologic research; however measurement always implies some degree of variability[1]. Ideally the only source of variability in measurements should be the variability within subjects, however often observer variability and other sources of variability are also present.

In clinical care, diagnosis often depends on measurements, and disagreement in these measurements may have obvious implications for clinical practice and may also have medico-legal consequences[2]. Before the introduction, in clinical practice, of a new diagnostic method, it is essential to assess the agreement between the new method and the traditional one and it is also fundamental to know whether the new method can be reproduced by a second observer. In research, disagreement in measurements may lead to discrepant results in validity or randomized controlled studies with the same objectives and with the same methods, consequently misleading and heterogeneous results in meta-analysis will be found[2]. As it is impossible to control the various sources of variation of a measurement, agreement studies have a very important role. Despite the importance of agreement studies, misleading and sometimes inappropriate measures of agreement have been used in leading medical literature[2,3]. Considering the limitations of current strategies to assess agreement and as no strategy seems to be fail-safe to compare the degree of agreement among different populations, agreement studies should be interpreted with caution and possibly combined with the use of different strategies to assess agreement always with the limitations of these strategies in mind[2,4].

In this article, we propose a new approach to assess agreement based on information theory.

Methods and Results

Most used strategies to assess agreement

The Intraclass Correlation Coefficient (ICC) is claimed to be suitable for observer agreement assessment[5]. The ICC is defined as the ratio of the variance between subjects, to the total variance[6,7]. These variances are derived from analyses of variance (ANOVA). Fleiss and Shrout present different kinds of ICC derive from different ANOVA models, and the ANOVA model depends on the study design[6]. One-way random effects model should be used when the subjects are deemed a random sample of subjects to be evaluated by the observers. The focus of interest is the difference of the individual observer's rating from the average rating of the observers for the i^{th} subject[6].

In two-way models the observers are deemed an important factor in the ICC computation. In two-way random effects model, not only the subjects are deemed random, but the observers are deemed a random effect as well. In two-way mixed model each target is evaluated by k observers, who are the only observers of interest, in this case, the observers are a fixed effect while the subject ratings are a random effect[6].

The ICC (from two-way models) that should be used for assessing agreement was defined by McGraw and Wong as the 'ICC for agreement'. We obtain the 'ICC for consistency' or the 'ICC for agreement' excluding or not the observer variance from the denominator mean square, respectively[8]. The systematic variability due to observers is irrelevant for 'ICC for consistency' but it is not for 'ICC for agreement'.

The ICC ranges from 0 (no agreement) to 1 (perfect agreement), however it can be negative, how such negative ICC values should be interpreted is quite unclear. The ICC assumptions, multivariate normal distributions and equality of variances, should be checked.

An important limitation of ICC is that it is strongly influenced by the variance of the trait in the population in which it is assessed[9]. This limitation can be illustrated with the following example:

suppose that we aim to assess whether a depression scale can be reproduced by a second observer when applied to a random sample of the adult population (an heterogeneous population, with high variance) the scale's ICC may be higher than when the scale is applied to a very homogeneous population (with low variance), such as patients hospitalized for acute depression. This ICC characteristic has also been considered by some authors as an advantage, for it would make the discordance relative to the magnitude of measurements[10], however comparability across populations is not possible with ICC. Consequently, ICC values have no absolute meaning, and the cut-off value of 0.75 proposed by Burdock[7] and Lee[11], and often reproduced to signify a good agreement, has limited justification.

Lin's concordance correlation coefficient (CCC) is the Pearson Correlation of Coefficient, which assesses the closeness of the data to the line of best fit, modified by taking into account how far the line of best fit is from the 45º line crossing the origin[12]. Lin objected to the use of the ICC as a way to assess agreement between methods of measurement and developed the concordance correlation coefficient (CCC). However, there are similarities between certain specifications of the ICC and the CCC [13]. Some limitations of ICC, like the limitation of comparability of population described above, are also present in CCC[14].

Bland and Altman propose the limits of agreement to assess agreement between methods of measurement[15]. Limits of agreement can be calculated based on the mean difference between the measurements of two methods in the same subjects and the standard deviation of these differences. Approximately 95% of these differences will lay between the mean differences ± 1.96 standard deviation of these differences. The limits of agreement approach depends on some assumptions about the data: the mean and standard deviation of the differences are constant throughout the range of measurement and these differences are approximately Normally distributed[15]. Limits of agreement are expressed in terms of the scale of measurement and the decision whether a degree of agreement is acceptable or not is always a clinical, not statistical, judgment.

Other approaches[16, 17], have been proposed for assessing agreement; however all of them are also limited when the aim is to compare the agreement in different populations with different trait characteristics.

The New Approach: Information-Based Measure of Disagreement

Entropy, introduced by Shannon, can be described as the average amount of information contained in a variable[18]. A high value of entropy means that a large amount of information is needed to describe an outcome variable about which we have high uncertainty. The sum over all logarithms of possible outcomes of the variable is a valid measure of the amount of information, or uncertainty, contained in a variable. Consider that we aim to measure disagreement between measurements obtained by observer A (variable A) and observer B (variable B). Also, consider for variable A, a vector A that can take the range of non-negative values $(a_1,\ldots,a_n)$ and for variable B, a vector B that can take the range of non-negative values $(b_{1,\ldots,}b_n)$. The intuition behind our definition is that the disagreement between A and B is related to the differences between them, with the minimum reached when A and B are identical. So, it is then natural to consider $\sum_{i=1}^{n}\log_2|a_i - b_i|$ the amount of information contained in the differences between observers A and B. By adding 1 to the differences, we avoid the behavior of the logarithmic function between 0 and 1, while keeping a quite natural close relation to the notion of entropy where Shannon considered the log of the inverse of a probability, i.e., the log of a value always greater or equal to 1. Now, in order to get a value between 0 and 1 we normalize the amount of information contained in the differences to obtain the following measure of disagreement.

Considering two vectors $A = (a_1,\ldots,a_n)$, and $B = (b_1,\ldots,b_n)$ with non-negative components. The information-based measure of disagreement is defined as:

$$d(A,B)=\frac{1}{n}\sum_{i=1}^{n}\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)$$

with the convention $\frac{|0-0|}{\max\{0,0\}}=0$.

This coefficient equals 0 when the observers agree, or when there is no disagreement, i.e., when $a_i = b_i$. In this case we say that there is no information in the differences between observers A and B. As observer A and B's measurements disagree, the amount of information in the differences between observers A and B increases and the information-based measure of disagreement increases, tending to 1, or, in other words, the distance between the observers increases.

The convention $\frac{|0-0|}{\max\{0,0\}}=0$ has a natural interpretation since if both observers rate 0, they agree, so the contribution of this observation to the final sum of disagreement is $\log_2(1+0) = 0$.

Properties of Information-Based Measure of Disagreement

The information-based measure of disagreement, d(A,B), is a metric, i.e., the following properties hold:

- is non-negative, d(A,B) is always greater or equal to than zero;
- is zero if and only if the observers rate exactly the same measurements, d(A,B)=0 if and only if A=B;
- is symmetric, d(A,B)=d(B,A) and
- the triangular inequality is verified, d(A,B)≤d(A,C)+d(C,B).

The information-based measure of disagreement is scaled invariant, d(A,B)=d(kA,kB) with k a non zero constant. It has also differential weighting, i.e., a difference found between high values contribute less than the same difference found between low values. The appendix contains the proof of each property.

Inference of the information-based measure of disagreement

In the absence of any distributional assumption, the most obvious inference method is the non parametric bootstrap. Each of the M bootstrap samples is taken from a_i (i = 1 to n), and b_i (i = 1 to n), where a_i and b_i are the measurements rated by the observers A and B respectively, on the subject i, and provides an estimate of the proposed information-based measure of disagreement using $d(A,B)=\frac{1}{n}\sum_{i=1}^{n}\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)$. A confidence interval for the proposed measure of disagreement can be obtained from the percentiles of the empirical distribution of the M estimates.

Example - systolic blood pressure

Bland and Altman present the example of measurements of systolic blood pressure of 85 individuals, by two observers (J and R) with sphygmomanometer, and one other measurement, by a semiautomatic device (S)[19]. Luiz et al re-analyze the data and also observe, with a graphical approach, a greater agreement between the two observers than between the observers and the semiautomatic device[16]. Using our information-based measure of disagreement we also obtained significantly more disagreement between each observer and the semiautomatic device than between the two observers (Table 1).

Example - Apgar score at first minute

The Apgar score is widely used in developed countries for assessing the clinical status of the newborn, determining whether or not there is a need for immediate medical care. It is usually assessed at the first and fifth minutes after birth and ranges between zero and ten. It is based on the analysis of the newborn's skin color, response to stimulus, breathing, muscle tonus and heart rate.

Cardiotocography is the most common method used for fetal monitoring during labor. It registers the heart rate of the fetus and contractions of the uterus. The value of fetal monitoring procedure is in its ability to predict newborn outcome, as the prediction of newborn outcome during labor can decrease perinatal mortality and morbidity.

Three obstetricians were asked to evaluate 72 intrapartum cardiotocographic tracings independently and through them predict the Apgar score at the first minute. After birth, the true values of the Apgar scores were evaluated by the caregivers responsible for immediate neonatal support. To simulate an observer who estimated Apgar by chance, we randomly generated Apgar score values between zero and ten.

Table 2 presents the true Apgar scores evaluated after birth for 72 newborns and the Apgar scores predicted by the three obstetricians based on the intrapartum cardiotocographs tracings. The obstetricians were blinded for the neonatal outcome and evaluate tracings independently. Table 2 also presents a random estimation of the Apgar scores created by a generation of random values between zero and ten.

Table 3 presents the agreement, assessed by the ICC, and the disagreement, assessed with the information-based measure of disagreement, between each obstetrician´s predictions and Apgar evaluated by the caregiver for neonatal support and between randomly generated Apgar and evaluated by the caregiver for neonatal support.

As ICC is a measure of agreement, higher values correspond to better agreement. However, as the proposed information-based measure of disagreement is a measure of disagreement, higher values correspond to a higher disagreement, i.e., a worse agreement. ICC results suggest that one obstetrician predicted Apgar scores no better than the random generation of values. This is an artifact caused by the dependence of the ICC on variance. When Apgar scores were randomly generated, variance is much higher (7.7) than when obstetricians predict them (variances ranging from 0.4 and 0.7).

On the other hand, the information-based measure of disagreement evidenced that the Apgar score predictions by the obstetricians were significantly better, i.e. there was less disagreement with Apgar scores evaluated by the caregivers responsible for neonatal support, than by the random generation of Apgar scores (Table 3).

Simulation study

To assess the performance of the proposed measure we used simulations in different scenarios. From Uniform distributions between 0 and 100 we generated the observer A's ratings for sample sizes of 10, 25, 50, 100, 500 and 5000. In a first scenario, Observer B's ratings were created by adding a systematic difference to observer A's ratings: Observer B's ratings were equal to observer A's ratings plus 10 unities. In a second scenario, Observer B's ratings were created by adding a random difference to Observer A's ratings: Observer B's ratings were equal to observer A's ratings plus a quantity generated by a Normal distribution with mean 0 and standard deviation 5. In a third scenario, Observer B's ratings were created by adding both random and systematic differences to Observer A's ratings: Observer B's ratings were equal to observer A's ratings plus a quantity generated by a Normal distribution with mean 10 and standard deviation 5. We assumed as 0 all generated negative Observer B's ratings.

A thousand bootstrap samples were taken from data for each scenario and each sample size to provide the estimates of the proposed information-based measure of disagreement. Table 4 presents the mean and standard deviation of those 1000 estimates (standard error). Table 4 also presents the bias, i.e., the difference between the original and the mean of 1000 estimates of the information-based measure of disagreement. The point estimates of the proposed information-based measure of disagreement had very small bias and standard error even for small sample sizes.

Discussion and Conclusions

We can look at disagreement between observers as the distance between their ratings, so the metric properties are important. Moreover, the proposed measure of disagreement is scale-invariant, i.e., the degree of disagreement between two observers should be the same if the measurements are analyzed in kilograms or in grammas, for example.

Differential weighting is another property of the proposed information-based measure of disagreement: each comparison between two ratings is divided by a normalizing factor, depending upon each pair of ratings alone, before summing. Therefore, the information-based measure of disagreement is appropriate for ratio scale measurements (with a natural zero) and it is not appropriate for interval scale measurements (without a natural zero). For example, outside air temperature in Celsius or Fahrenheit scale does not have a natural zero. The 0 degrees is arbitrary and it does not make sense to say that 20 degrees is twice as hot as 10 degrees. Temperature in Celsius or Fahrenheit scale is an interval scale. On the other hand, height has a natural 0 meaning: the absence of height. Therefore, it makes sense to say that 80 inches is twice as large as 40 inches. Height is a ratio scale. Suppose the heights of a sample of subjects measured independently by two different observers. A difference between the two observers of one inch in a child subject represents a worse observers' error than a disagreement between observers of one inch in an adult subject. Due to differential weighting property of the information-based measure of disagreement, a difference between the observers of one inch in a child in fact weights less to the estimate of information-based measure of disagreement between observers than a difference between the observers of one inch in an adult.

The usual approaches used to evaluate agreement have the limitation of the comparability of populations. In fact, ICC depends upon the variance of the trait population; although this characteristic can be considered an advantage it does not allow one to compare the degree of agreement across different populations. Also the interpretation of the Limits of Agreement depends on what can be considered clinically relevant or not, which could be subjective and different from reader to reader. The comparison of the degree of agreement in different populations is not straightforward. Other approaches[16, 17] to assess observer agreement have been proposed, however the comparability of populations is still not easy with these approaches.

The proposed information-based measure of disagreement, used as a complement to current approaches for evaluating agreement, can be useful to compare the degree of disagreement among different populations with different characteristics, namely with different variances.

Moreover, we believe that information theory can make an important contribution to the relevant problem of measuring agreement in medical research, providing not only better quantification but also better understanding of the complexity of the underlying problems related to the measurement of disagreement.

Acknoledgements

This research was partially supported byCSI2 (PTDC/EIA-CCO/099951/2008) and the grant SFRH / BD / 28419 / 2006.

References

1. Bland M. An Introduction to Medical Statistics. 3th ed. Oxford University Press. 2000.
2. Costa Santos C, Costa Pereira A, Bernardes J. Agreement studies in obstetrics and gynaecology: inappropriateness, controversies and consequences. Br J Obstet Gynaecol 2005; 112 (5): 667-9.
3. Gow RM, Barrowman NJ, Lai L, Moher D. A review of five cardiology journals found that observer variability of measured variables was infrequently reported. J Clin Epidemiol. 2008;61:394-401.
4. Luiz RR, Szklo M. More than one statistical strategy to assess agreement of quantitative measurements may usefully be reported. J Clin Epidemiol. 2005; 58:215-6.
5. de Vet H. Observer reliability and agreement. In: P. Armitage and T. Colton, Editors, Encyclopedia of biostatistics. 1st ed. Wiley, Chichester. 1999, pp. 3123–3127.
6. Shrout PE, Fleiss JL. Intraclass Correlation: uses in assessing rater reliability. Psycological Bulletin.1979; 88: 420-8.
7. Burdock EI, Fleiss JL and Hardesty AS. A new view of interobserver agreement. Psycological Bulletin. 1963;16:373-84.
8. McGraw KO, Wong, SP. Forming inferences about some intraclass correlation coefficients. Psychological Methods. 1996;1(1):30-46.
9. Muller R, Buttner P. A critical discussion of intraclass correlation coefficients. Stat Med 1994;13:2465-2476.
10. Steiner DL, Norman GR. Health Measurements Scales. A practical guide to their development and use. 3th ed. Oxford University Press. 2003.
11. Lee D, Koh D, Ong CN. Statistical evaluation of the agreement between two methods for measuring a quantitative variable. Comput Biol Med 1989;19:61-70.
12. Lin LK. A concordance correlation coefficient to evaluate reproducibility. Biometrics. 1989;45:255–68.
13. Nickerson CAE. A Note on "A Concordance Correlation Coefficient to Evaluate Reproducibility". Biometrics. 1997;53:1503–7.
14. Atkinson G, Nevill A. Comment on the Use of Concordance Correlation to Assess the Agreement between Two Variables. Biometrics. 1997;53:775-7.
15. Bland JM, Altman DG. Statistical methods for assessing two methods of clinical measurement. Lancet 1986;1:307-310.
16. Luiz RR, Costa AJ, Kale PL, Werneck GL. Assessment of agreement of a quantitative variable: a new graphical approach. J Clin Epidemiol. 2003;56:963-7.
17. Monti KL. Folded empirical distribution function curves-mountain plots. Am Stat. 1995;33:525-7.
18. Shannon ^I. A Mathematical Theory of Communication. Bell System Technical Journal. 1948;27:379-423,623-656.
19. Bland JM, Altaman DG. Measuring agreement in method comparison studies. Stat Methods Med Res. 1999;8:135-60.

APPENDIX

Properties of the proposed disagreement measure

The proposed disagreement measure, d(A,B), has the flowing properties:

Property 1 (Non-negativity). *Let A and B be two non-negative, random variables of size n, d(A, B) ≥ 0.*

Proof. If the two observations are zero, then by convention its contribution is zero. On the other observations, the property follows from the fact that the factors in the argument of log are at least 1.

Property 2 (Identity of indiscernibles). *Let A and B be two non-negative, random variables of size n. d(A, B) = 0 if and only if A = B.*

Proof. d(A, B) = 0 if and only if $\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)=0$ for all $1\le i\le n$. So, $a_i=b_i=0$ or $\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1=1$ for all $1\le i\le n$, i.e., if and only if A=B.

Property 3 (Symmetry). *Let A and B be two non-negative, random variables of size n. d(A, B) = d(B, A).*

Proof. *d*(*A*, *B*) = *d*(*B*, *A*) because of maximum and the absolute value symmetry.

Property 4 (Triangular inequality). *Let A, B and C be three non-negative, random variables of size n.* d(A, B) ≤ d(A, C) + d(C, B)

Proof. In order to prove d(A, B) ≤ d(A, C) + d(C, B) we have to prove that for all $1\le i\le n$:

$$\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)\le\log_2\left(\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+1\right)+\log_2\left(\frac{|c_i-b_i|}{\max\{c_i,b_i\}}+1\right) \qquad (I)$$

First we deal with the case where at least two of the elements a_i, b_i and c_i are 0.

If $a_i=b_i=c_i=0$ then the inequality I is trivially true with the convention made.

If $a_i=b_i=0$ and $c_i\neq 0$ then $\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)=0$, $\log_2\left(\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+1\right)\ge 0$ and $\log_2\left(\frac{|c_i-b_i|}{\max\{c_i,b_i\}}+1\right)\ge 0$, and then, again, the inequality I is true.

If $a_i=c_i=0$ and $b_i\neq 0$. In this case $\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)=\log_2\left(\frac{|b_i|}{b_i}+1\right)$, $\log_2\left(\frac{|c_i-b_i|}{\max\{c_i,b_i\}}+1\right)=\log_2\left(\frac{|b_i|}{b_i}+1\right)$ and $\log_2\left(\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+1\right)=0$ which verifies the inequality I.

Now assume that at least two of the values a_i, b_i and c_i are positive. Proving the inequality I is equivalent, to prove:

$$\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\le\left(\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+1\right)\left(\frac{|c_i-b_i|}{\max\{c_i,b_i\}}+1\right)\Leftrightarrow$$

$$\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\le\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+\frac{|c_i-b_i|}{\max\{c_i,b_i\}}+\frac{|a_i-c_i||c_i-b_i|}{\max\{a_i,c_i\}\max\{c_i,b_i\}}+1$$

So, is sufficient to prove that:

$$\frac{|a_i-b_i|}{\max\{a_i,b_i\}}\le\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+\frac{|c_i-b_i|}{\max\{c_i,b_i\}}$$

Suppose that $\max\{a_i,b_i\}\ge c_i$.

Then $\max\{a_i,b_i\}\ge\max\{b_i,c_i\}$ and $\max\{a_i,b_i\}\ge\max\{a_i,c_i\}$.

So, $\frac{1}{\max\{a_i,b_i\}}\le\frac{1}{\max\{c_i,b_i\}}$ and $\frac{1}{\max\{a_i,b_i\}}\le\frac{1}{\max\{c_i,a_i\}}$.

Thus $\frac{|a_i-b_i|}{\max\{a_i,b_i\}}\le\frac{|a_i-c_i|+|c_i-b_i|}{\max\{a_i,b_i\}}\le\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+\frac{|b_i-c_i|}{\max\{c_i,b_i\}}$

Now, suppose that $\max\{a_i,b_i\}<c_i$ then $a_i,b_i\le c_i$. Without loss of generality we can suppose that $b<a<c$. So, prove that:

$$\frac{|a_i-b_i|}{\max\{a_i,b_i\}}\le\frac{|a_i-c_i|}{\max\{a_i,c_i\}}+\frac{|c_i-b_i|}{\max\{c_i,b_i\}}$$

is the same to prove:

$$\frac{c_i-a_i}{c_i}+\frac{c_i-b_i}{c_i}-\frac{a_i-b_i}{a_i}\ge 0$$

and

$$\frac{c_i-a_i}{c_i}+\frac{c_i-b_i}{c_i}-\frac{a_i-b_i}{a_i}=2-\frac{a_i}{c_i}-\frac{b_i}{c_i}-\left(1-\frac{b_i}{a_i}\right)=1-\frac{a_i}{c_i}-\frac{b_i}{c_i}+\frac{b_i}{a_i}$$

as $a<c$ then $\frac{b}{c}<\frac{b}{a}$ and $\frac{a}{c}>0$. Thus $1-\frac{a_i}{c_i}-\frac{b_i}{c_i}+\frac{b_i}{a_i}>0$

Property 5. *Let A and B be two non-negative, random variable of size n.* d(A, B) ≤ 1.

Proof. The property follows the fact that $\frac{|a_i-b_i|}{\max\{a_i,b_i\}}\le 1$, so $\log_2\left(\frac{|a_i-b_i|}{\max\{a_i,b_i\}}+1\right)\le 1$

Property 6. (Scale invariance)**.** *Let A and B be two non-negative, random variable of size n and k a positive, non zero constant. d*(*A*, *B*) = *d*(*kA*, *kB*).

Proof.

$$d(kA,kB)=\frac{\sum_{i=1}^{n}\log_2\left(\frac{|ka_i-kb_i|}{\max(ka_i,kb_i)}+1\right)}{n}$$

$$=\frac{\sum_{i=1}^{n}\log_2\left(\frac{k|a_i-b_i|}{k\max(a_i,b_i)}+1\right)}{n}=d(A,B)$$

Commitment and Authentication Systems*

Alexandre Pinto** André Souto Armando Matos Luís Antunes

DCC-FC & LIACC
R. Campo Alegre 1021/1055
4169-007 Porto

Abstract. In the present paper, we answer a question raised in the paper *Constructions and Bounds for Unconditionally Secure Non-Interactive Commitment Schemes*, by Blundo et al, 2002, showing that there is a close relation between unconditionally secure commitment schemes and unconditionally secure authentication schemes, and that an unconditionally secure commitment scheme can be built from such an authentication scheme and an unconditionally secure cipher system.
To investigate the opposite direction, we define optimal commitment systems and show that these must be resolvable design commitment schemes. Then, a proof is given that the resolvable design commitment schemes are a composition of an authentication system and a cipher system and the conclusion follows that this is the case for all optimal commitment systems.
We also show how to build optimal schemes from transversal designs that are easy to build and can be more efficiently implemented than the proposal in the previously cited paper.

Keywords: Commitment, Authentication, Unconditional Security, Galois Field.

1 Introduction

Commitment schemes were introduced by Blum ([1]). It is not possible to build unconditionally secure commitment schemes with only two parties, but Rivest proposed the first unconditionally secure non interactive commitment scheme with a trusted initializer, in a non-published manuscript called "Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer" and available at
http://citeseer.ifi.unizh.ch/rivest99unconditionally.html.

In [2], the authors begin a mathematical formalization of such commitment schemes. They also prove some lower bounds on the binding probabilities and propose and analyse implementations of optimally secure systems. They give a general description of a commitment scheme in Rivest's model that uses encoding and authentication keys and also a simplified scheme where the authentication key is not necessary. Then they offer a construction of commitment schemes based on resolvable designs and analyse its binding probabilities.

They list two open problems: finding a lower bound on the amount of information that has to be pre-distributed to the users and sent by the sender to the receiver; and the existence of some relation between these schemes and authentication codes.

The first of these questions was answered in [7], while the present paper proposes to answer the second one. We show that an unconditionally secure commitment scheme with trusted initializer can be built from

- a composition of an unconditionally secure authentication code without secrecy, without splitting and with no arbitration
- and an unconditionally secure cipher system.

This relation suggests an attack already referred in [7] that is the counterpart of the impersonation attack of an authentication system. We give a combinatorial and an information-theoretic lower bound for its probability of success. The second of these bounds is already present in [7] but while their proof used techniques from hypothesis testing, ours uses only the definition of mutual information and the log sum inequality.

We begin by giving some definitions and notation in Section 2, as well as formal definitions of unconditionally secure cipher, authentication and commitment systems. We then analyse the possible attacks against commitment schemes in Section 3, and show how these can be built from a cipher system and an authentication code.

In Section 5, we define the notion of optimal commitment scheme and show that such a scheme is a resolvable design commitment scheme as proposed in [2]. We follow with the main results of this paper, showing that optimal systems must be resolvable design commitment schemes and that all of these can be decomposed into a cipher system and an authentication code.

We then propose a generalization of the affine plane commitment scheme in [2] that is efficiently implementable in both hardware and software by allowing an alphabet of source states with size $|S| = 2^n$ rather than having $|S| = p$ for prime p. In the former case, the needed arithmetic operations reduce to bit shifts and bitwise logical operations, which have very fast hardware and machine-code implementations.

We show that this is possible for every n, by building an appropriate Transversal Design and using a result due to Stinson ([12]) to turn it into an unconditionally secure authentication system. Our commitment scheme follows from composition with the One-Time Pad cipher system. By the previous results, this scheme is optimal.

Finally, Section 7 contains some concluding remarks and possible directions for future work.

2 Preliminaries

We denote alphabets by calligraphic type, e.g. $\mathcal{P}, \mathcal{C}$. Depending on context, these alphabets can be seen as subsets of $\mathbb{N}$ or of $\{0,1\}^*$. Elements of these alphabets are usually represented by lowercase letters. The size of a set is denoted by $|\cdot|$. Random variables over these sets are represented by uppercase versions of the name of the set, like P, C and so on. Greek letters are reserved for some probabilities and real parameters not greater than 1. Probability expressions of the type $\Pr[X = x]$ and $\Pr[X = x|Y = y]$ are sometimes simplified to $p(x)$ and $p(x|y)$ respectively. The function $E(\cdot)$ denotes the expected value of some distribution.

Sometimes, functions of two arguments are written as parameterized functions of one argument. For example, $f(k, s)$ is the same as $f_k(s)$. The function $H(\cdot)$ and its variants denote Shannon's entropy function.

The users of the protocols bear the standard names of the literature: Alice and Bob are the legitimate participants, Eve is a passive eavesdropper, Oscar is a malicious opponent with complete power over the channel between Alice and Bob, and Ted is a trusted initializer. Alice

* A preliminary version has appeared in the proceedings of ICITS '07.
** Corresponding author contacts – email: alex.miranda.pinto@gmail.com, phone: +351 919528890, fax: N/A.

is always the sender and Bob the receiver. In the commitment scheme, both Alice and Bob can be malicious and try to break the protocol.

We now give formal definitions of the cryptographic constructions used in this paper.

2.1 Cryptographic Systems

Cipher Systems

Definition 1. *A cipher system is a tuple denoted* $CP(\mathcal{P}, \mathcal{C}, \mathcal{K}, f(k,p))$ *where* $\mathcal{P}$ *is the alphabet of plaintext messages,* $\mathcal{C}$ *is the alphabet of ciphertext messages and* $\mathcal{K}$ *is the alphabet of secret keys. For each* $k \in \mathcal{K}$*, there is a function* $f_k : \mathcal{P} \mapsto \mathcal{C}$ *with* $f_k(p) = f(k,p)$ *that is injective and defined for all* $p \in \mathcal{P}$.$\diamond$

A cipher system is unconditionally secure if the random variables $P, K, C = f(K,P)$ satisfy:

$$H(P) = H(P|C).$$

Authentication Systems We give only the details needed for authentication codes and refer the reader to [8], [9] and [10] for more information.

Authentication codes without secrecy allow a party to send a message composed of a source value s and an authenticator a such that an attacker has at most a probability α of forging a new message or a probability β of altering a known valid message such that the receiver will accept these forgeries as valid.

If the attacker sees i valid messages before sending his forgery, this is called a deception attack of level i. The most basic attacks are the impersonation attack ($i = 0$) and the substitution attack ($i = 1$) and are the only ones considered in this paper. The probability of success for an attack of level i is denoted P_{d_i}.

The participants of the scheme share a secret key that allows the sending party to compute the right authenticator for a source value, by computing $a = f(k,s)$, and the receiving party to decide if a message is a forgery or not, by evaluating $g(k, \langle s, a \rangle)$.

There is always some positive probability of success for any attack. We list some bounds from the literature: $\log P_{d0} \geq H(K|M) - H(K)$ ([8]), $\log P_{d1} \geq -H(K|M)$ ([5]), $P_{d_0} \geq \frac{|\mathcal{S}|}{|\mathcal{M}|}$ and $P_{d_1} \geq \frac{|\mathcal{S}|-1}{|\mathcal{M}|-1}$ ([11], [12]). An authentication code is unconditionally secure if the maximum probabilities of success meet these bounds. We only consider systems where any pair (s,a) is valid for at least one key.

Authentication Attacks: In an impersonation attack, the attacker sends a forgery $(s,a) \in \mathcal{S} \times \mathcal{A}$ without seeing any valid message. The probability of the receiver accepting this message as valid is

$$\text{payoff}(s,a) = \sum_{k \in \mathcal{K},\, g_k(s,a)=1} \Pr[K = k]$$

In a substitution attack, the attacker knows that (s_1, a_1) is a valid message before sending a forgery $(s,a) \in \mathcal{S} \times \mathcal{A}$. The probability of the receiver accepting this message as valid is

$$\text{payoff}(s,a,s_1,a_1) = \frac{\sum_{k \in \mathcal{K},\, g_k(s,a)=1,\, g_k(s_1,a_1)=1} \Pr[K=k]}{\sum_{k \in \mathcal{K},\, g_k(s_1,a_1)=1} \Pr[K=k]}$$

Definition 2. *An authentication code without arbitration, without splitting and without secrecy is a tuple denoted* $AC(\mathcal{S}, \mathcal{A}, \mathcal{K}, f(k,s), g(k, \langle s, a \rangle), \alpha, \beta)$ *where* $\mathcal{S}$ *is the set of source states,* $\mathcal{A}$ *is the set of authenticators and* $\mathcal{K}$ *is the set of the secret keys. For each* $k \in \mathcal{K}$*, there is an injective encoding rule* $f_k : \mathcal{S} \mapsto \mathcal{A}$ *with* $f_k(s) = f(k,s)$ *that computes the message authentication code (mac) for each source value* $s \in \mathcal{S}$*. For each* $k \in \mathcal{K}$*, a verification function* $g_k : \mathcal{S} \times \mathcal{A} \mapsto \{0,1\}$ *with* $g_k(s,a) = g(k, \langle s, a \rangle)$ *can be defined as* $g_k(s,a) = 1$ *iff* $f(k,s) = a$*.*

The value α *is the maximum chance of success for an impersonation attack and* β *is the maximum chance of success for a substitution attack. Formally, for any fixed* $k \in \mathcal{K}$*,*

$$\max_{(s,a) \in \mathcal{S} \times \mathcal{A}} \text{payoff}(s,a) \leq \alpha$$

and

$$\max_{(s,a),(s_1,a_1) \in \mathcal{S} \times \mathcal{A}} \text{payoff}(s,a,s_1,a_1) \leq \beta$$

$\diamond$

Commitment Systems Commitment schemes with a trusted initializer allow a sender to commit to a value and send that commitment to a receiver such that the value she committed to remains hidden from this. In a second step, the sender reveals her commitment and the receiver may verify that the sender is not fooling him. The third participant is required only to give the other two some information that enables them to carry out the protocol. This third participant is completely honest and trusted by the other two.

A commitment scheme must satisfy a Concealing Property, i.e., the receiver can guess the value committed to only with a probability equal to a uniform random guess. On the other hand, the sender's commitment must effectively bind her, which means she can not open to the receiver a value different from her commitment. As shown in [2], a commitment system can not be completely binding, and so we say a system is $(1-\epsilon)$-binding if the probability of the sender deceiving the receiver is at most ϵ.

Definition 3. *A commitment scheme is a tuple denoted* $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ *where* $\mathcal{X}$ *is the source states alphabet,* $\mathcal{Y}$ *is the coded states alphabet,* $\mathcal{K}$ *is the alphabet of the committer's keys,* $\mathcal{V}$ *is the alphabet of the verifier's tags. For each* $k \in \mathcal{K}$*, there is an injective encoding rule* $f_k : \mathcal{X} \mapsto \mathcal{Y}$ *with* $f_k(x) = f(k,x)$ *that computes the encoding of each possible commitment* $x \in \mathcal{X}$*. For each* $v \in \mathcal{V}$*, there is a verification function* $g_v : \mathcal{K} \mapsto \{0,1\}$ *with* $g_v(k) = g(v,k)$*.*

The values α *and* β *are the maximum chances of success for the two kinds of attack described in Section 3.1. Formally,*

$$\max_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}_k} \Pr[V = v] \leq \alpha$$

and

$$\max_{k,k' \in \mathcal{K}} \frac{\sum_{v \in \mathcal{V}_{k'} \cap \mathcal{V}_k} \Pr[V=v]}{\sum_{v \in V_{k'}} \Pr[V=v]} \leq \beta$$

$\diamond$

2.2 Combinatorial Designs

Design theory is a large body of research dedicated to statistical constructions known as designs. We give only the results and definitions we need in this paper and refer the reader to some textbook in design theory.

Definition 4. *A $t-(v,k,\lambda,b,r)$ design is a pair $(\mathcal{D},\mathcal{S})$ where $b=|\mathcal{D}|$, $t\leq k<v$, $\lambda>0$, $\mathcal{S}$ is a set of v distinct elements, called points, and $\mathcal{D}$ is a collection of subsets of $\mathcal{S}$ each with exactly k elements, called blocks. Besides, every point occurs in exactly r blocks and every subset of $\mathcal{S}$ with exactly t points occurs in exactly λ blocks.*

These constructions are called t-designs. When $t=2$, they are usually called Balanced Incomplete Block Designs (BIBD). $\diamond$

Definition 5. *A design is said to be resolvable if its blocks can be partitioned into sets $\mathcal{P}_i$ called parallel classes, each with exactly v/k elements, such that the blocks in each parallel class form a partition of $\mathcal{S}$.*

A resolvable $1-(v,k,\lambda,b,r)$ design is called affine if for any two blocks B_1, B_2 belonging to different parallel classes, it happens that $|B_1\cap B_2|$ is equal to k^2/v. $\diamond$

Theorem 1. *If $(\mathcal{D},\mathcal{S})$ is a $t-(v,k,\lambda,b,r)$ design, then $b=\frac{\lambda\binom{v}{t}}{\binom{k}{t}}$. Furthermore, each point occurs in exactly $r=\frac{\lambda\binom{v-1}{t-1}}{\binom{k-1}{t-1}}$.*

Definition 6. *A $t-(v,k,\lambda,b,r)$ design is called symmetric if $b=v$, and equivalently $k=r$.*

Theorem 2. *In a symmetric $t-(v,k,\lambda,b,r)$ design, the intersection of any two blocks has cardinality λ.*

Definition 7. *A transversal design $TD(k,n,\lambda)$ is a pair $(\mathcal{D},\mathcal{S})$ such that $|\mathcal{S}|=k\cdot n$, the points in $\mathcal{S}$ can be divided into exactly k groups of n elements each, there are $\lambda\cdot n^2$ blocks, each of them containing at most one point from each group, and any pair of points from distinct groups occurs in exactly λ blocks.* $\diamond$

It is easy to see from the definitions that a transversal design is not a 2-design because two points from the same group are never contained in any block.

3 Analysis of Commitment Schemes

This section presents an analysis of the possible attacks against a commitment scheme and shows how to build such schemes from a cipher and an authentication scheme.

3.1 Security

In a commitment scheme, both participants can launch attacks.

Bob's Attack The security of a commitment scheme can be measured by the probability that Alice has of cheating Bob while Bob can not cheat Alice with more than a priori probability. Bob's chances at guessing each x should not be altered by his knowledge of v and y, i.e., for all triples (x,y,v), $p(x|y,v)=p(x)$. This can be summarized using Shannon's entropy with $H(X)=H(X|Y,V)$.

Alice's Attack Let Alice commit to a value x and send $y=f(k,x)$ to Bob where k is her secret key. Alice cheats Bob if she can reveal a $k'\neq k$ such that $f_{k'}^{-1}(y)=x'$ with $x'\neq x$, and Bob accepts k' as valid, i.e., $g_v(k')=1$. It is proved in [2] that a commitment scheme can not be invulnerable against all of Alice's attacks: Alice can compute the set $\mathcal{V}_k=\{v\in\mathcal{V}:p(v|k)>0\}$ of all the tags that Bob may have. She then picks the tag $v_0\in\mathcal{V}_k$ that maximizes $p(v|k)$. Let $\alpha=p(v_0|k)$. By an averaging argument, $\alpha\geq 1/|\mathcal{V}_k|$. Now, Alice picks two values $x\neq x'$ and computes $y=f(k,x)$. But by the concealing property, there is a key k' such that $f(k',x')=y$ and $g(v_0,k')=1$ which allows Alice to cheat successfully if Bob's tag is v_0. The success probability of this attack is the probability that Bob is holding the tag chosen by Alice, α. It is shown in the same paper that the average probability of this attack is at least $2^{-H(V|K)}$ and therefore there's at least one instance with at least this probability of success.

The attack described above is the counterpart to a substitution attack in an authentication system. There is yet another attack that Alice can perform, which has been pointed in [7]. This is the counterpart of an impersonation attack. These relations are a consequence of the construction of commitment schemes from authentication codes. In the previous attack, Alice makes the best possible use of her private information, but she can also launch an attack ignoring it altogether. To do this, Alice simply computes for each key the probability that Bob accepts it, i.e., for a fixed key k she finds

$$\gamma(k)=\sum_{v\in\mathcal{V}_k}p(v). \tag{1}$$

She then picks the key that maximizes the above sum and reveals it to Bob in the revealing step. We give two combinatorial lower bounds for this attack when the distribution of the keys and tags is uniform.

Theorem 3. *There is some $k\in\mathcal{K}$ with probability of success $\gamma(k)\geq\frac{E(|K_v|)}{|\mathcal{K}|}$, where $E(|K_v|)$ signifies the average number of keys that each tag validates.*

Proof. Consider $\gamma(k)$ as defined above. Its average value is

$$1/|\mathcal{K}|\sum_{k\in\mathcal{K}}\sum_{v\in\mathcal{V}_k}p(v)=$$
$$1/|\mathcal{K}|\sum_{v\in\mathcal{V}}|\mathcal{K}_v|p(v)=$$
$$\frac{E(|K_v|)}{|\mathcal{K}|}.$$

Then, by an averaging argument, there is some k which has $\gamma(k)\geq\frac{E(|K_v|)}{|\mathcal{K}|}$.

Corollary 1. *There is some $k \in \mathcal{K}$ with probability of success $\gamma(k) \geq \frac{E(|\mathcal{V}_k|)}{|\mathcal{V}|}$, where $E(|\mathcal{V}_k|)$ signifies the average number of tags that validate each key.*

Proof. It suffices to note that $\sum_{k\in\mathcal{K}} |\mathcal{V}_k| = \sum_{v\in\mathcal{V}} |\mathcal{K}_v|$.

We can show an analog result with Shannon's entropy:

Theorem 4. *There is some $k \in \mathcal{K}$ with probability of success*

$$\gamma(k) \geq 2^{-I(K;V)}.$$

Proof. By definition of mutual information (see [3])

$$-I(K;V) = \sum_{k\in\mathcal{K}, v\in\mathcal{V}} p(k,v) \log \frac{p(k)p(v)}{p(k,v)}.$$

For each $k \in \mathcal{K}$, $p(k,v) = 0$ for every $v \notin \mathcal{V}_k$. Thus, the above can be written

$$\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v) \log \frac{p(k)p(v)}{p(k,v)}. \tag{2}$$

The log sum inequality (see [3]) states that

$$\sum_{i=1}^{n} a_i \log \frac{b_i}{a_i} \leq \left(\sum_{i=1}^{n} a_i\right) \log \frac{\sum_{i=1}^{n} b_i}{\sum_{i=1}^{n} a_i}. \tag{3}$$

Applying (3) to (2),

$$\begin{aligned}
-I(V;K) &= \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v) \log \frac{p(k)p(v)}{p(k,v)} \\
&\leq \left(\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v)\right) \log \frac{\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k)p(v)}{\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v)} \\
&= 1 \cdot \log \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k)p(v) \\
&= \log \sum_{k\in\mathcal{K}} p(k)\gamma(k) \\
&= \log E(\gamma(K))
\end{aligned}$$

where $E(\gamma(K))$ is the average value of the success probability for each k. By an averaging argument, there is at least one k that has probability greater or equal to the average value. For this k:

$$\gamma(k) \geq E(\gamma(K)) \geq 2^{-I(K;V)}.$$

A commitment scheme is said to be unconditionally secure if it is perfectly concealing and the maximum probabilities of success for these two attacks are equal and meet the lower bounds. This implies $H(K) = H(V|K) + H(K|V)$.

3.2 Construction of Commitment Schemes

This section presents a proof that an unconditionally secure commitment scheme can be built using an unconditionally secure cipher system and an unconditionally secure authentication system without secrecy as building blocks. Each user in these systems has a function to play. We call that function a "role". In composing a commitment scheme with two different systems, the users of the former will have to play the different roles of the latter at different steps, so we refer to these roles by writing the abbreviation of the system followed by the role played, all within square brackets. In the remainder of the paper, CP stands for "cipher system", AC stands for "authentication system" and CM stands for "commitment system". The cipher system consists of three roles: [CP.Alice], [CP.Bob] and [CP.Eve]; the authentication system has roles [AC.Alice], [AC.Bob] and [AC.Oscar].

As previously mentioned, in a commitment scheme there are two kinds of attacks. The first is against secrecy: Bob must not learn the secret value Alice committed to before the right time, so Alice sends it enciphered. The second attack is against authentication: Alice must not send a fake opening key, so she must send it through an authentication scheme. In the first step, Alice uses a cipher system without receiver. She merely sends Bob an encrypted message, but he must not be able to open it. Essentially, Bob takes the role of [CP.Eve]. After Bob receives a key in the revealing step, he takes the role of [CP.Bob] and opens the ciphertext learning Alice's commitment. In the second step, Alice sends Bob the key to open the encrypted message he has, but Bob needs to be sure it is the right key, distributed to her in the initial phase. Essentially, Alice acts as a relay between Ted and Bob. Since she reads what the initializer sends and has a choice of relaying that message or changing it for another one altogether, she has complete control over the channel. In this phase, Ted plays the role [AC.Alice], Alice plays the role [AC.Oscar] and Bob plays the role [AC.Bob]. We summarize the above in Table 1.

User	**Committing Step**	**Revealing Step**
Alice	[CP.Alice]	[AC.Oscar]
Bob	[CP.Eve]	[CP.Bob] / [AC.Bob]
Ted	...	[AC.Alice]

Table 1. Roles Played

Theorem 5. *Given an unconditionally secure cipher system $CP(\mathcal{P},\mathcal{C},\mathcal{K},f(k,p))$ and an authentication system without secrecy $AC(\mathcal{S},\mathcal{A},\mathcal{E},h(e,s),g(e,\langle s,a\rangle),\alpha,\beta)$ with $\mathcal{S}=\mathcal{K}$, there is a commitment scheme with initializer (per Rivest's model) $CM(\mathcal{P},\mathcal{C},\mathcal{S}\times A,\mathcal{E},f(s,p),g(e,\langle s,a\rangle),\alpha,\beta)$.*

Proof. The several components of the commitment scheme are obtained from the cipher and the authentication system as shown in Table 2. Because we have used letters given in the initial definitions, there are two different alphabets labelled $\mathcal{K}$. They are not to be confused. For each $k \in CP.\mathcal{K}$, there are $|\mathcal{A}|$ different $k' \in CM.\mathcal{K}$, all with the same behaviour in the cipher system. Considering the analysis in [2], these $|\mathcal{A}|$ keys form a parallel class of keys in the combinatorial design used as basis for the resolvable design commitment scheme. The protocol is as follows:

Cipher	Commitment	Authentication
$\mathcal{P}$	$\mathcal{X}$	...
$\mathcal{C}$	$\mathcal{Y}$	...
$\mathcal{K}$	...	$\mathcal{S}$
...	$\mathcal{K}$	$\mathcal{S} \times \mathcal{A}$
...	$\mathcal{V}$	$\mathcal{E}$
$f(k,p)$	$f(k,x)$	...
...	$g(k,v)$	$g(k,\langle s,a\rangle)$

Table 2. Equivalences between systems

1. **Initialization:** Ted chooses uniformly at random an encryption key $s \in \mathcal{S}$ and an authentication key $e \in \mathcal{E}$, computes the authenticator $a = h(e,s)$ and sends $\langle s,a\rangle$ to Alice and e to Bob.
2. **Committing Step:** Alice commits to x by sending Bob the encryption $y = f(k,x)$.
3. **Revealing Step:** Alice sends Bob a possibly false key $\langle s',a'\rangle$. Bob checks if $g_e(\langle s',a'\rangle) = 1$ and if so, he decrypts $x' = f_{s'}^{-1}(y)$.

This construction yields a commitment scheme that follows Rivest's model, as is shown next. The crucial point of this construction is that the source value that Ted wants to send Bob in the revelation step is the actual key that the latter must use to open Alice's commitment. The figures in Section 4 can help to understand this. It is easy to verify that the families of functions $f(k,p)$ and $g(k,\langle s,a\rangle)$ satisfy the formal requirements of the commitment scheme. Now we check the concealing and binding properties.

Concealing Let x_0 be the value Alice committed to, k_0 the key she holds and $y_0 = f(k_0,x_0)$ the value Bob received. Let e_0 be Bob's tag. Let $X_{y_0} = \{x \in \mathcal{X} : \exists k \in \mathcal{K} \text{ s.t. } f(k,x) = y_0\}$ be the set of possible plaintexts for the ciphertext Bob holds, $\mathcal{K}_{e_0} = \{k \in \mathcal{K} : \exists a \in \mathcal{A} \text{ s.t. } h(e_0,k) = a\}$ be the set of possible Alice's keys and let $X_{y_0,e_0} = \{x \in \mathcal{X} : \exists k \in \mathcal{K}_{e_0} \text{ s.t. } f(k,x) = y_0\}$ be the set of possible values for Alice's commitment given the information Bob knows. Because the authentication system does not have splitting, for each possible $k \in \mathcal{K}$ and $e \in \mathcal{E}$, there is exactly one $a \in \mathcal{A}$ such that $h(e,k) = a$. Therefore, all the keys can be associated to each particular e, and so $\mathcal{K} = \mathcal{K}_{e_0}$, implying that $X_{y_0} = X_{y_0,e_0}$.

Bob's probability of guessing Alice's commitment without or with knowledge of e_0 is, respectively, $p(x_0|y_0) = \sum_{k\in\mathcal{K}, f(k,x_0)=y_0} p(k)$ and $p(x_0|y_0,e_0) = \sum_{k\in\mathcal{K}_{e_0}, f(k,x_0)=y_0} p(k)$ and by the above reasoning they're equal. Then

$$\begin{aligned} H(X|Y,E) &= E_{y,e}(H(X|Y=y,E=e)) \\ &= E_{y,e}(\sum_{x\in\mathcal{X}} p(x|y,e)\log 1/p(x|y,e)) \\ &= E_y(\sum_{x\in\mathcal{X}} p(x|y)\log 1/p(x|y)) \\ &= E_y(H(X|Y=y)) \\ &= H(X|Y). \end{aligned}$$

If follows, by assumption, that $H(X|Y,E) = H(X)$ and this commitment scheme satisfies the concealing property.

Binding Let x_0 be the value Alice committed to and (k_0,a_0) be the key/authenticator pair that she holds. In order to reveal a value $x' \neq x_0$, Alice needs to make Bob accept a key $k' \neq k_0$. Alice can make two kinds of attack, as described in Section 3.1. Her chances of success are, for the first attack:

$$\max_{(k',a')\in\mathcal{K}\times\mathcal{A}} \sum_{e\in\mathcal{E},\, g(e,\langle k',a'\rangle)=1} p(e).$$

This corresponds to the impersonation attack against the authentication system and so this probability is at most α. Likewise, for the second attack:

$$\max_{(k',a'),(k_0,a_0)\in\mathcal{K}\times\mathcal{A},k'\neq k_0} \frac{\sum_{e\in\mathcal{E},\, g(e,\langle k',a'\rangle)=1,\, g(e,\langle k_0,a_0\rangle)=1} p(e)}{\sum_{e\in\mathcal{E},\, g(e,\langle k',a'\rangle)=1} p(e)},$$

which corresponds to a substitution attack and is at most β. Thus, this commitment is $(1 - \max(\alpha,\beta))$-binding.

Corollary 2. *Given an unconditionally secure cipher system and an unconditionally secure authentication system without secrecy there is an unconditionally secure commitment scheme with initializer (per Rivest's model).*

Next, we show how the different flows of information in the three systems are related.

4 Flow Analysis

In the conclusion to the paper [2], the authors suggest a possible relation between commitment schemes and authentication schemes with arbitration, but point that the information flows between these systems are different. Here, we analyse the different flows of information in a commitment scheme, and how these are realized through the flows present in the cipher and in the authentication systems. We understand by information flows the data that is sent from one user to another user. The following pictures help visualize the flows in the different systems. In these figures, there are blocks representing each participant in the system and arrows representing the messages sent by them, this is, the flows of information between users. Within some blocks is another name within square brackets. This represents the name of the user of the commitment scheme that will be playing the role indicated by the block. For instance, when Alice sends her commitment to Bob, he is playing the role of Eve in the cipher scheme: he receives a ciphertext but can not read it. Next is shown how each flow is used to implement the flows of the final commitment scheme. When necessary, we describe roles with the same notation of Section 3.2. In the following list, we describe the flows in each system and identify them with a name between brackets that indicates the system the flow belongs to and the step when it takes place. When some steps have two similar flows, these are further distinguished with letters 'a' and 'b'.

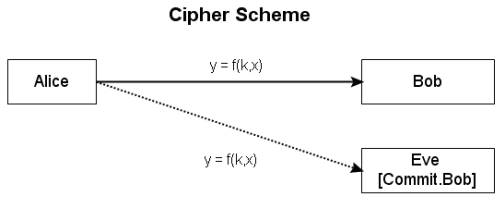

Fig. 1. A Cipher Scheme

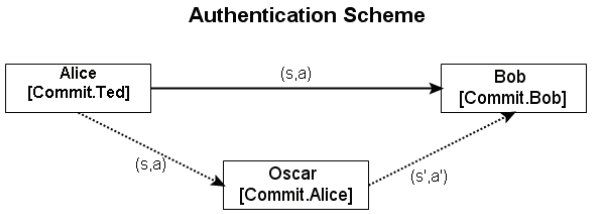

Fig. 2. An Authentication Scheme

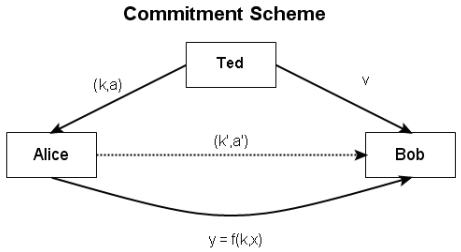

Fig. 3. A Commitment Scheme

(CP1) Alice (a) and Bob (b) receive a secret key by some secure channel. This includes the case where they create a key themselves and exchange it.
(CP2) A message is sent from Alice to Bob (a) and possibly also read by Eve (b).

In an authentication system as described above, there are the following flows:

(AC1) Alice (a) and Bob (b) receive a secret key by some secure channel.
(AC2) A message is sent from Alice to Oscar (a), who may change it before relaying it to Bob (b). Note this is just a simplified model. In reality, Alice sends the message to Bob, but Oscar may intercept and alter it or not.

In a commitment scheme, there are the following information flows:

(CM1) Ted gives a key to Alice (a) and a verification tag to Bob (b).
(CM2) Alice sends her commitment to Bob.
(CM3) Alice sends her key to Bob to open her commitment.

The information flows of the commitment scheme are carried out by the information flows of the other systems like this:

- Flow (CM1.b) is achieved by flow (AC1.b). Flow (AC1.a) is ignored because Ted does not need to remember the key after he creates a valid message to send Alice. Flow (CM1.a) is achieved by flow (AC2.a), that is, Ted takes the role [AC.Alice] and sends a message to Alice ([AC.Oscar]). Due to the nature of the construction, namely because the authentication system does not have secrecy, flow (AC2.a) includes flow (CP1.a), because Alice now has a key for the cipher system.
- Flow (CM2) is achieved by flow (CP2.b).
- Flow (CM3) is achieved by flow (AC2.b). From this message, Bob deduces a key, completing flow (CP1.b), and opens the commitment by flow (CP2.a).

Cipher	Commitment	Authentication
(CP1.a)	(CM1.a)	(AC2.a)
...	(CM1.b)	(AC1.b) [(AC1.a)]
(CP2.b)	(CM2)	...
(CP1.b) (CP2.a)	(CM3)	(AC2.b)

Table 3. Information Flows

5 Optimal Commitment Schemes

In [2], the authors propose a general commitment scheme which they call "resolvable design commitment scheme". In this section, we define optimal commitment schemes and show that they are resolvable design affine commitment schemes. Then, we close the circle showing that

all resolvable design commitment schemes can be viewed as the composition of a cipher system and an authentication system. For simplification, in what follows, consider that the source values, keys and verification tags are distributed uniformly, since this maximizes uncertainty and therefore security.

Definition 8. *A commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ is optimal if it is unconditionally secure, $|\mathcal{X}| = |\mathcal{Y}|$ and has the minimum number of keys for a fixed number of source states and the desired security level. Besides, the probability of Alice's cheating should be equal to the probability of Bob's cheating.* $\diamond$

Lemmas 1 and 2 give some properties that an optimal commitment system must have. Lemma 3 excludes BIBDs as the possible minimal system, and this is necessary because such systems are not resolvable. This means that there can be pairs of blocks with empty intersection and so these are counted in Lemma 4. After these lemmas, we're ready to give the two main theorems: that an optimal commitment system must be affine resolvable and that a resolvable commitment scheme can be decomposed into a cipher system and an authentication system.

Lemma 1. *If a commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ is optimal, then $\alpha = \beta$ and $|\mathcal{V}| = (1/\alpha)^2$.*

Proof. By definition,

$$\alpha = \max_{k \in \mathcal{K}} |\mathcal{V}_k|/|\mathcal{V}| \geq E(|\mathcal{V}_k|)/|\mathcal{V}|.$$

and by optimality of the system the above holds with equality. Then, $|\mathcal{V}_k|$ is constant for all k. Likewise, and by Theorem 3, $|\mathcal{K}_v|$ must be constant. By definition,

$$\begin{aligned}
\beta &= \max_{k \neq k' \in \mathcal{K}} \frac{\Pr[g(v,k) = 1, g(v,k') = 1]}{\Pr[g(v,k') = 1]} \\
&= \max_{k \neq k' \in \mathcal{K}} \frac{|\{v \in \mathcal{V} : g(v,k) = g(v,k') = 1\}|}{|\{v \in \mathcal{V} : g(v,k) = 1\}|} \\
&= \max_{k \neq k' \in \mathcal{K}} \frac{|\mathcal{V}_k \cap \mathcal{V}_{k'}|}{|\mathcal{V}_k|}.
\end{aligned}$$

Let $\mu = \max_{k \neq k' \in \mathcal{K}} |\mathcal{V}_k \cap \mathcal{V}_{k'}|$. Then

$$\beta = \frac{\mu}{|\mathcal{V}_k|}.$$

The value μ can not be 0 because if so each tag would verify exactly one key and Bob would be able to cheat Alice with absolute certainty, so β is minimum if $\mu = 1$ and

$$\begin{aligned}
\alpha &= |\mathcal{V}_k|/|\mathcal{V}| \\
\beta &= 1/|\mathcal{V}_k|.
\end{aligned}$$

We show that γ is minimum when $\alpha = \beta$. Let $|\mathcal{V}_k|$ and $|\mathcal{V}|$ be such that $\alpha = \beta = \gamma$. Then $|\mathcal{V}_k|^2 = |\mathcal{V}|$. Denote by α_0 and γ_0 the values of α and γ respectively in this case.

Assume for contradiction that there is some combination of values $|\mathcal{V}_k|$ and $|\mathcal{V}|$ such that $\gamma = \max(\alpha, \beta) < \gamma_0$. Assume w.l.o.g that $\alpha > \beta$. Then

$$\begin{aligned}
&\alpha = \gamma < \gamma_0 = \alpha_0 \Rightarrow \\
&|\mathcal{V}_k|/|\mathcal{V}| < |\mathcal{V}|^{1/2}/|\mathcal{V}| \Leftrightarrow 1/|\mathcal{V}_k| > 1/|\mathcal{V}|^{1/2} \Rightarrow \\
&\beta > \alpha_0 > \alpha.
\end{aligned}$$

Thus we have a contradiction, and so the minimum γ is achieved when $\alpha = \beta$. This implies that $1/|\mathcal{V}_k| = |\mathcal{V}_k|/|\mathcal{V}|$ and $|\mathcal{V}| = |\mathcal{V}_k|^2 = (1/\alpha)^2$.

Lemma 2. *If a commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ is optimal, then $|\mathcal{X}|^2 = |\mathcal{K}| = |\mathcal{V}|$.*

Proof. Let v be Bob's tag and y the coding of Alice's commitment. Define $\mathcal{X}_{y,v} = \{x : \exists\ k \in \mathcal{K}_v \ s.t.\ f_k^{-1}(y)\ \}$ to be the set of possible commitments for Alice given the information Bob holds. Then, $|X_{y,v}| \leq |\mathcal{K}_v|$. Because $|\mathcal{K}_v|$ is constant, by Lemma 1,

$$\begin{aligned}
H(X|Y,V) &= \sum_{y \in \mathcal{Y}, v \in \mathcal{V}} \Pr(y,v) H(X|Y=y, V=v) \\
&\leq \sum_{y \in \mathcal{Y}, v \in \mathcal{V}} \Pr(y,v) \log |X_{y,v}| \\
&\leq \sum_{y \in \mathcal{Y}, v \in \mathcal{V}} \Pr(y,v) \log |\mathcal{K}_v| \\
&= \log |\mathcal{K}_v|
\end{aligned}$$

and by optimality of the system $\log |\mathcal{K}_v| \geq H(X) = \log |\mathcal{X}|$.

By the proof of Corollary 1, $|\mathcal{K}| \cdot |\mathcal{V}_k| = |\mathcal{K}_v| \cdot |\mathcal{V}|$ and using Lemma 1, this brings $|\mathcal{K}| = |\mathcal{K}_v| \cdot |\mathcal{V}_k| \geq |\mathcal{X}| \cdot |\mathcal{V}_k|$. But since the system is optimal and the number of keys is minimal, then it must be that $|\mathcal{K}_v| = |\mathcal{X}|$. Bob's chance of guessing Alice's commitment is $1/|\mathcal{X}|$, by definition of security. From Lemma 1, Alice's chance is $1/|\mathcal{V}_k|$. Because in an optimal system Alice's chance of success is equal to Bob's, $1/|\mathcal{X}| = 1/|\mathcal{V}_k| \Leftrightarrow |\mathcal{X}| = |\mathcal{V}_k|$ which implies $|\mathcal{K}| = |\mathcal{X}|^2 = |\mathcal{V}|$.

Lemma 3. *If a commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ is optimal, then its incidence matrix can not be a Balanced Incomplete Block Design (BIBD).*

Proof. By Lemma 2, $|\mathcal{V}| = |\mathcal{K}|$. Suppose the incidence matrix of the commitment scheme is a $2 - (v, k, \lambda, b, r)$ Balanced Incomplete Block Design. Then the design is symmetric and by Theorem 2, any two keys have exactly λ tags validating them. By the proof of Lemma 1, the maximum intersection between any two lines should be 1, so $\lambda = 1$. Then, by Theorem 1, each tag validates exactly $r = (v-1)/(k-1)$ keys and again by Theorem 2, $r = k$. This brings $b = v = k^2 - k + 1$ where $b = |\mathcal{K}|$ and $v = |\mathcal{V}|$. By definition, $r = |\mathcal{K}_v|$.

By Lemma 2, $|\mathcal{K}|/|\mathcal{X}| = |\mathcal{K}_v|$. Then, b/k must be an integer, but

$$\begin{aligned}
\frac{b}{k} &= \frac{k^2 - k + 1}{k} \\
&= k - 1 + 1/k
\end{aligned}$$

and this can not be an integer for $k > 1$.

Lemma 4. *Let $p = |\mathcal{X}|$. If a commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ is optimal, then the sum of distinct pairs of keys that don't have any tag in common is $p^2 \cdot (p-1)/2$.*

Proof. Consider a square matrix where cell (i,j) contains the value $|\mathcal{V}_{k_i} \cap \mathcal{V}_{k_j}|$ where $k_i \neq k_j \in \mathcal{K}$.

There are $p^2 \cdot (p^2 - 1)$ filled cells. We count the pairs that have a non-empty intersection. Each key has tag v_1 in common with $p-1$ different keys and since it is validated by p tags, each key contributes with $p(p-1)$ to the total of the sum

$$\sigma' = \sum_{k_i \in \mathcal{K}} \sum_{k_j \neq k_i \in \mathcal{K}} |\mathcal{V}_{k_i} \cap \mathcal{V}_{k_j}|.$$

Therefore, $\sigma' = p^3(p-1)$. But this sum counts each pair twice, so the total number of distinct intersections is $\sigma = p^3(p-1)/2$. We can find the total number of distinct key pairs that don't intersect, recalling that in this particular case all filled cells are either 0 or 1, which means that σ is the sum of all 1s in the table. As before, only a half of the matrix needs to be considered. Then, the number of distinct key pairs without tags in common is

$$(p^4 - p^2)/2 - p^3(p-1)/2 = p^2 \cdot (p-1)/2.$$

Theorem 6. *If a commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ is optimal, then it is an affine resolvable commitment scheme, and all keys in each parallel class encrypt each value $x \in \mathcal{X}$ to the same value $y \in \mathcal{Y}$. That is, the function $y = f(k,x)$ depends only on the index of the parallel class containing k, and not on k itself.*

Proof. Let $p = |\mathcal{X}|$. From the previous results, there are p^2 keys. Fix some $x_0 \in \mathcal{X}$. The concealing property implies that there must be exactly p keys transforming x_0 into each possible value $y \in \mathcal{Y}$. Then, the keys can be grouped in p groups such that all keys $k_{i,j}$ in group i satisfy $f(k_{i,j}, x_0) = y_i$. There are exactly p keys validated by each verifier tag. For each two keys k_i and k_j validated by the same tag, it must happen that $f(k_i, x_0) \neq f(k_j, x_0)$ or else there won't be enough keys to hit all the values in $\mathcal{Y}$. But by Lemma 4 and a counting argument, this implies that all pairs of keys in different groups must have exactly one common tag. Since there are p disjoint keys in each group, each validated by p tags, each group forms a partition of $|\mathcal{V}|$ and is therefore a parallel class. The design is therefore resolvable and since the maximum intersection between two keys is $1 = k^2/v$ it is also affine.

Now consider a value $x_1 \in \mathcal{X}$ different from x_0. Suppose there are two keys k_i, k_j in different groups that code x_1 in the same way. That is:

$$\begin{aligned} &f(k_i, x_0) \neq f(k_j, x_0) \\ &f(k_i, x_1) = f(k_j, x_1) \\ &|\mathcal{V}_{k_i} \cap \mathcal{V}_{k_j}| = 1. \end{aligned}$$

Following the same reasoning as above, if $f(k_i, x_1) = f(k_j, x_1)$ then they can not have any tag in common, contradicting the previous division in groups. Therefore, keys in different groups code x in different ways and by a counting argument all keys in the same group transform x into the same y.

Repeating the argument for any $x_l \in \mathcal{X}$ and for all groups, it must happen that all keys k_i, k_j in the same group satisfy

$$f(k_i, x_l) = f(k_j, x_l)$$

and so the theorem is proved.

Theorem 7. *A resolvable design commitment scheme obtained from a resolvable $1-(v,k,\lambda,b,r)$ design $(\mathcal{D}, \mathcal{S})$ is a composition of a perfectly secure cipher system and an authentication code with $P_{d0} = k/v$ and $P_{d1} = \frac{\max |B_1 \cap B_2|}{k}$ for all distinct $B_1, B_2 \in \mathcal{D}$.*

Proof. Let B be Alice's block and w be Bob's tag in the above system. Then $w \in B$. By Theorem 1, $b = r \cdot v/k$. Then, w can be seen as a number between 0 and $v-1$.

By definition of resolvable design, B belongs to some parallel class. Then, B can be written (i,j), where i is the index of the parallel class and j is the index of the block within the parallel class. The pair (i,j) can be interpreted as a pair source value / authenticator.

When Alice commits to $x_0 \in \mathbb{Z}_r$, she sends Bob $y_0 = (x_0 + i) \bmod r$. Then, x_0 can be seen as a symbol in alphabet $\Sigma = \mathbb{Z}_r$ and y_0 as a displacement of i positions over that alphabet. This corresponds to applying a Ceasar's cipher to the secret message x_0. In general, Ceasar's cipher is not secure, but here the message is composed of only one symbol, which means its size is equal to the size of the key. In this situation, it is equivalent to the one-time pad and is unconditionally secure. Since all blocks in the same parallel class encipher x_0 in exactly the same way, the index of the parallel class represents the cipher key used by Alice.

Now we show that the design used to check the validity of the value revealed after the commitment can also be used to make an authentication scheme with r source states, v/k possible authenticators for each source state and v encoding rules. Any block from the design can be identified by a pair (i,j), where i is the index of the parallel class it belongs to and j its index within that class. Let B be the set represented by a block with indices (i,j). Then, B can be interpreted as belonging to an authentication code: i represents the source code and j the respective authenticator. Each element $w \in B$ contributes for the definition of an encoding rule in the following way: $f(w,i) = j$. Since each w belongs to exactly one block in each parallel class, there is a unique value associated to each pair (w,i). Since there are k elements in each block, there are exactly k encoding rules associating i to j.

The probability of an impersonation attack is the maximum probability of finding the right authenticator for a specific pair key / source state. There are a total of v encoding rules. The number of rules that associate some source state i to a given authenticator j is, by construction, k. Thus, for all such pairs (i,j),

$$\text{payoff}(i,j) = \sum_{w \in \mathcal{S} : f(w,i) = j} 1/v = k/v$$

and because this is constant for all pairs, it is the probability of an impersonation attack. For the substitution attack, suppose the attacker knows that the secret encoding rule associates i_1

to j_1. By construction, there are k such keys. For any pair (i, j), the probability of success is

$$\begin{aligned}\text{payoff}(i, j, i_1, j_1) &= \frac{\sum_{w \in \mathcal{S}: f(w,i)=j, f(w,i_1)=j_1} p(w)}{\sum_{w \in \mathcal{S}: f(w,i_1)=j_1} p(w)} \\ &= \frac{|B_{i,j} \cap B_{i_1,j_1}|}{k}\end{aligned}$$

where $B_{i,j}$ is the block of the design indexed by (i, j). Then, the probability of the substitution attack is $\frac{\max |B_1 \cap B_2|}{k}$ over all $B_1, B_2 \in \mathcal{D}, B_1 \neq B_2$.

6 Generalization to Galois Fields

As noted in [2], affine resolvable 2-designs are optimal among the resolvable designs in terms of binding probabilities, however not many classes of such designs are known to exist. Notwithstanding this, there are other kinds of designs that can achieve the same goals, namely Transversal Designs.

This section addresses this question by showing how to construct a resolvable Transversal Design $TD(2^n, 1, 2^n)$, for any n, that is also a $1-(2^{2n}, 2^n, 1)$ affine resolvable design. From such a design, we then build an unconditionally secure authentication code and an unconditionally secure commitment scheme.

Theorem 8. *For any positive integer n, it is possible to construct a Transversal Design $TD(2^n, 1, 2^n)$.*

Proof. The order of any finite field can be written p^k, where p is a prime and $k \geq 1$ is an integer (see [4]). Let $GF(p^n)$ represent one such field. For any finite field $GF(p^n)$, there is a primitive polynomial of degree n and coefficients modulo p ([4]). For composite orders p^n, the elements of the field are considered to be polynomials and the operations of the field are addition and multiplication of polynomials modulo the prime p. Addition is denoted by $\oplus$ and multiplication by $\odot$.

Fix some n and some primitive polynomial for $GF(2^n)$. We build a table with 2^{2n} rows and 2^n columns. The rows are divided in 2^n groups of 2^n elements each, and uniquely identified by a pair (a, b) where $a, b \in \mathbb{Z}_{2^n}$. Each column is labelled by a number $x \in Z_{2^n}$. Cell $((a, b), x)$ holds the value $a \odot x \oplus b$, which is a number in $\mathbb{Z}_{2^n}$.

We show this table represents a $TD(2^n, 1, 2^n)$ transversal design. Consider the set of points $\mathcal{V} = \{0, 1, \ldots, 2^{2n} - 1\}$ and divide them in 2^n groups of 2^n points. Each row represents a block with 2^n points, one from each group. For row j, the i^{th} value represents the index of the point in the i^{th} group that belongs to the j^{th} block. By construction, each block has one point in each group. Finally, each two points from distinct groups can occur in only one block. To see this, let (x_0, y_0) and (x_1, y_1) be two points from distinct groups, where $x_0 \neq x_1$. For both points to be in the same block, there must be a pair (a, b) such that $a \odot x_0 \oplus b = y_0$ and $a \odot x_1 \oplus b = y_1$. Then,

$$\begin{aligned}&a \odot x_0 - y_0 = a \odot x_1 - y_1 \Leftrightarrow \\ &a \odot (x_0 - x_1) = y_0 - y_1.\end{aligned}$$

Since $(x_0 - x_1)$ and $(y_0 - y_1)$ are defined and $(x_0 - x_1) \neq 0$, then a is completely determined, and so is b. That means there is only one pair satisfying both equations, which means both points can belong to only one block. This concludes the proof.

This design originates an authentication code $AC(2^n, 2^n, 2^{2n}, 1/2^n, 1/2^n)$, as proved in [12].

With such an unconditionally secure authentication scheme, and using the one-time pad as a perfect cipher system, we can build an unconditionally secure commitment scheme as outlined in section 3.2.

Let $\mathcal{S} = \{0, 1\}^n$. The protocol is as follows:

1. **Initialization:** Ted chooses randomly a pair $(a, b) \in \mathcal{S} \times \mathcal{S}$ and a number $x_1 \in \mathcal{S}$, computes $y_1 = a \odot x_1 \oplus b$ and sends the pair (x_1, y_1) to Bob and the pair (a, b) to Alice.
2. **Committing Step:** Alice commits to $x_0 \in \mathcal{S}$ by enciphering $y_0 = x_0 \oplus a$ and sending y_0 to Bob.
3. **Revealing Step:** Alice sends (a', b') to Bob, who checks that $a' \odot x_1 \oplus b' = y_1$. If so, he computes $x'_0 = y_0 \oplus a'$ and accepts x'_0 as Alice's commitment.

The resulting commitment scheme is a generalization of the affine plane commitment scheme. It uses the one-time pad as cipher system, which is very fast to implement both in software and hardware. Besides, it allows the use of a complete alphabet of strings of size n, whereas in the affine plane with order p, the alphabet of allowed values does not coincide with any alphabet of all strings of a given size. In general, the latter systems will be less efficiently implemented in hardware and software because the basic instructions are more oriented to a fixed size of bits than the corresponding arithmetic value.

Finally, we address the matter of calculating addition and multiplication in a Galois Field. Each element of $GF(2^n)$ is a binary string of size n, where bit i corresponds to the coefficient of term x^i in a polynomial of degree strictly less than n. Addition is performed by adding the polynomial coefficients degree by degree, which corresponds easily to an exclusive-or between both strings. Thus, this operation can be performed extremely fast both in hardware and software, especially if n corresponds to the size of the word of the microprocessor used.

Multiplication can be computed by shifting one of the strings to the left an appropriate number of positions for each bit 1 in the other string, XORing the several displaced versions together and computing the remainder of the division by the primitive polynomial. This sounds complicated, but can all be implemented with shift and XOR instructions, and both kinds are quickly implemented in hardware and machine-code. A whole analysis of a possible implementation is given in [6].

7 Conclusion and Further Work

This paper continues the work began in [2] and [7] in the analysis of unconditionally secure commitment schemes. It gives formal characterizations of cipher schemes, authentication schemes without secrecy, splitting or arbitration and perfectly concealing commitment schemes with a trusted initializer, as proposed by Rivest. Then, we showed how to build an unconditionally secure commitment scheme using an unconditionally secure cipher system and an unconditionally secure authentication code. Based on this construction, we showed a new proof for an attack

against commitment schemes analog to the impersonation attack of authentication codes that had been referred in [7] and gave a similar combinatorial lower bound. Then we showed that the resolvable design schemes proposed in [2] can be built according to our composition and that any optimal commitment system must be a system of this kind. We then considered whether it would be possible to build commitment schemes using the One-Time Pad as the cipher system. This requires the source alphabet to have 2^n elements. A positive answer was given to this question by showing how to build an adequate transversal design and then an authentication code with a source alphabet of size 2^n that is unconditionally secure against impersonation and substitution attacks. With the composition given before, this implies the existence of such commitment schemes that are in fact generalizations of the resolvable design commitment scheme to Galois Fields of composite order. The construction used can be applied to any field of order $GF(p^n)$ for any prime p, and not just $p = 2$.

This answers affirmatively the question raised in [2] about the relation of commitment schemes and authentication schemes, although with a solution different from the one suggested in that paper. Also this shows that a cipher system is needed too. Accordingly, the theory of authentication codes without secrecy can be used in the analysis of commitment schemes. Further work can be done to understand if other kinds of authentication codes can also be used to develop different kinds of commitment schemes and if there are viable alternatives to the cipher system.

8 Acknowledgements

This work was partially supported by KCrypt project (POSC/EIA/60819/2004), the grants SFRH/BD/13124/2003 and SFRH/BD/28419/2006 from Fundação para a Ciência e Tecnologia (FCT) and funds granted to Laboratório de Inteligência Artificial e Ciências de Computadores(LIACC) through the Programa de Financiamento Plurianual, FCT and Programa POSI.

References

1. Manuel Blum, *Coin flipping by telephone: a protocol for solving impossible problems* 24th IEEE Spring Computer Conference, pp 133 – 137 IEEE Press, 1982.
2. C. Blundo, B. Masucci, D.R. Stinson, R. Wei, *Constructions and Bounds for Unconditionally Secure Non-Interactive Commitment Schemes* (Designs, Codes and Cryptography), Vol 26, 2002.
3. Thomas M. Cover, Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.
4. Rudolf Lidl & Harald Niederreiter, *FINITE FIELDS* pp 50–51, 89–91 Cambridge University Press, 1983.
5. Ueli Maurer, *Authentication theory and hypothesis testing* IEEE Transactions on Information Theory, vol. 46(4), 1350-1356 2000.
6. D. McGrew and J. Viega, *The Galois/Counter Mode of Operation (GCM)* Submission to NIST Modes of Operation Process, January, 2004.
7. Anderson C. A. Nascimento, Akira Otsuka, Hideki Imai, Jörn Müller-Quade, *Unconditionally Secure Homomorphic Pre-distributed Commitments* Applied Algebra, Algebraic Algorithms and Error-Correcting Codes 2003, pp 87-97, 2003.
8. Gustavus J. Simmons, *Authentication Theory / Coding Theory*, Advances in Cryptology: Proceedings of CRYPTO' 84, Lecture Notes in Computer Science, vol. 196, 411-432, Springer Verlag, Berlin, 1985.
9. G. J. Simmons, *Message authentication: a game on hypergraphs* Congressus Numerantium, vol. 45, 161-192, 1984.
10. Gustavus J. Simmons, *A Natural Taxonomy for Digital Information Authentication Schemes*, Advances in Cryptology - CRYPTO' 87: Proceedings, Lecture Notes in Computer Science, vol. 293, 269-288, Springer Verlag, Berlin, 1988.
11. Douglas R. Stinson, *Combinatorial Characterization of Authentication Codes*, Advances in Cryptology: Proceedings of CRYPTO' 91, Lecture Notes in Computer Science, vol. 576, 62-72, Springer Verlag, Berlin, 1992.
12. Douglas R. Stinson, *Some Constructions and Bounds for Authentication Codes*, Advances in Cryptology - CRYPTO' 86: Proceedings, Lecture Notes in Computer Science, vol. 263, 418-425, Springer Verlag, Berlin, 1987.

One-way functions using Kolmogorov complexity*

Luís Antunes[1,2**]
Armando Matos[1,4***]
Alexandre Pinto[3]
André Souto[1,2†]
Andreia Teixeira[1,2‡]

[1] DCC - Faculdade de Ciências dad Universidade do Porto
[2] Instituto de Telecomunicações
[3] CCTC / DI - Universidade do Minho
[4] LIACC

emails: lfa@dcc.fc.up.pt; acm@dcc.fc.up.pt; ampinto@di.uminho.pt;
andresouto@dcc.fc.up.pt;andreiasofia@dcc.fc.up.pt

Abstract. A one-way function is intuitively defined to be an honest function which is easy to compute, but hard to invert. We propose an individual approach to one-way functions based on Kolmogorov complexity and prove some relationshipsbetween the new proposals and the classical definitions of one-way functions. In particular, we prove that Kolmogorov one-way functions are more restrictive than the classical one-way functions. We also relate Kolmogorov one-way functions with the conjecture of polynomial time symmetry of information.

1 Introduction

Our main objective is to prove the perfect secrecy of assymmetric cryptographic protocols using AlgorithmicInformation Theory instead of Information Theory. The advantage is the possibility of characterizing the security of individual instances of cryptographic protocols. For that purpose, we will apply the time-bounded version of Kolmogorov complexity to the analysis of cryptographic primitives that compose the assymmetric cryptographic protocols, namely one-way functions, pseudo-random generators, extractors, and zero-knowledge interactive proof systems. In this paper, we study one-way functions, defining Kolmogorov one-way functions. Intuitively speaking, a one-way function is a function that is easy to evaluate but hard to invert. The existence of these functions is an open question which implies $\mathbf{P} \neq \mathbf{NP}$. It is also an open question if this is a necessary condition. It is well known that the existence of one-way functions is necessary for the existence of pseudo-random generators, digital signatures, identification schemes, and public-key encryption. On the other hand, it is also known [BM84,GMR88,IL89,ILL89,Rom90] that the one-way functions are sufficient for the creation of a pseudo-random generator and trapdoor one-way functions are sufficient for the construction of public-key encryption and signature schemes. This is a motivation for the study of one-way functions in more depth.

We present a characterization of injective one-way functions based on time-bounded Kolmogorov complexity.

Classically, there are several definitions of one-way functions, such as: strong, weak and deterministic. Informally, f is a strong one-way function if all efficient inverting probabilistic algorithms succeed with negligible probability; f is a weak one-way function if all efficient inverting probabilistic algorithms fail with non-negligible probability; in the case of deterministic one-way functions, the function only needs to be resistant to deterministic adversaries. An interesting fact about these functions is that not every weak one-way function is a strong one-way function; however weak one-way functions exist if and only if strong one-way function exist (see [Gol01] for details).

The Kolmogorov complexity $K(x)$ ([Kol65], [Sol64] and [Cha66]) of a string x is the length of the shortest program producing x in a universal Turing machine. The time-bounded version of Kolmogorov complexity $K^t(x)$, is the length of the shortest program producing the string x in a universal Turing machine within time $t(|x|)$.

* A preliminary version of this work was presented at Computability in Europe 2010
** The authors are supported by CSI^2 (PTDC/EIA- CCO/099951/2008)
*** The author is also partially funded by LIACC through PFP of FCT
† The author is also supported by the grant SFRH / BD / 28419 / 2006 from FCT
‡ The author is also supported by the grant SFRH / BD / 33234 / 2007 from FCT

In this work, we characterize one-way functions using different approaches. In this section let us denote the expression $E(K^t_f(x|f(x),n))$ by E, the expected value of time-bounded Kolmogorov complexity of an object $x \in \Sigma^n$ given $f(x)$, where f is the description of the function, given by an oracle. We first relate E with the classical notions of one-way functions (Theorems 4, 5 and 6). We show that if $E > c$ for any positive constant c, then f is a weak one-way function. In the another direction, if f is a weak one-way function, then there is a polynomial $q(n)$ such that $E > \frac{\log(n)}{q(n)}$; on the other hand, we show that if f is a strong one-way function, then $E > c\left(1 - \frac{1}{q(n)}\right)\log n$ for every constant c and for every polynomial $q(n)$. In a second approach, we define one-way functions based on individual instances (Definition 7) and relate it with the classical notion of weak one-way functions (Theorem 7); then a characterization of one-way functions based on time-bounded Kolmogorov complexity of the individual instances is given (Definition 8) and a relationship with deterministic one-way functions is studied (Proposition 1). For the second approach the intuition is that the time-bounded Kolmogorov complexity is suitable to one-way functions using individual instances. We expect that x and f give all the information needed to compute in polynomial time $f(x)$ and, on the other hand, the value $f(x)$ does not convey in polynomial time useful information about x, in fact, we expect that the length of a shortestprogram computing x given $f(x)$, $|x|$, and f should be approximately equal to the length of a shortest program computing x without any auxiliary input.

We show that Kolmogorov one-way functions are *more restrictive* than classical one-way function in the sense that the existence of Kolmogorov one-way functions with certain parameters implies the existence of classical one-way functions.

In [LM93] and [LW95], the authors relate the existence of one-way functions and the conjecture of polynomial time symmetry of information. For the unbounded version of Kolmogorov complexity, symmetry of information was first proved by Levin (as suggested in [ZL70]), but the proof is not valid when polynomial time-bounds restrictions are imposed. The conjecture of polynomial time symmetry of information has close connections to several complexity theoretic questions, similar to the connections concerning the existence of one-way functions. In this work, we relate this conjecture with the existence of Kolmogorov one-way functions, by proving that the polynomial time symmetry of information fails if Kolmogorov one-way functions exist (Propositions 2 and 3 and Theorem 8).

2 Preliminaries

All the strings we use are elements of $\Sigma^* = \{0,1\}^*$ and we denote them by x, y, z. The function log always denotes the function $\log_2$ and $|.|$ represents the length of a string. The number of elements of a set A, i.e. its cardinality, is denoted by $\#A$. We assume that any time bound $t(n)$ we use is constructible and larger than n. For convenience we give definition of some order notations.

Definition 1. *Let f and g be two positive functions.*

- $f(n) \in O(g(n))$ *if and only if* $\exists k > 0,\ n_0\ \forall n > n_0, f(n) \leq k \cdot g(n)$.
- $f(n) \in \Omega(g(n))$ *if and only if* $\exists k > 0,\ n_0\ \forall n > n_0, f(n) \geq k \cdot g(n)$.
- $f(n) \in \omega(g(n))$ *if and only if* $\forall k > 0,\ \exists n_0\ \forall n > n_0, f(n) \geq k \cdot g(n)$.

2.1 Kolmogorov Complexity

We present the basic definitions and the results necessary for the rest of this paper; further details on Kolmogorov complexity can be found, for instance, on the comprehensive textbook [LV08]. We will use the prefix-free definition of Kolmogorov complexity. A set of strings A is prefix-free if there are not two strings x and y in A such that x is a proper prefix of y.

Definition 2. *Let U be a fixed universal Turing machine with a prefix-free domain. For any strings $x, y \in \{0,1\}^*$, the* Kolmogorov complexity of x given y with oracle access to f *is*

$$K_f(x|y) = \min_p\{|p| : U_f(p,y) = x\}.$$

For any time constructible t, the t time-bounded Kolmogorov complexity of x given y with oracle access to f *is defined by*

$$K_f^t(x|y) = \min_p\{|p| : U_f(p, y) = x \text{ in at most } t(|x| + |y|) \text{ steps}\}.$$

The default value for y, the auxiliary input for the program p, is the empty string ε and for oracle f is the null function. In order to avoid overloaded notation, in those cases we typically drop these arguments in the notation. Notice that Kolmogorov complexity is machine independent in the sense that we can fix a universal Turing machine U whose program size is at most a constant additive term worse than in any other machine, and the running time is, at most, a logarithmic multiplicative factor slower than in any other machine.

One important result in Kolmogorov complexity is the following theorem on the existence of incompressible strings (see [LV08]).

Theorem 1. (Incompressibility Theorem)

1. *For each n,* $\max\{K(x) : |x| = n\} = n + K(n) + O(1)$.
2. *For each fixed constant r, the set $\{x : (|x| = n) \wedge (K(x) \leq n + K(n) - r)\}$ has at most $2^{n-r+O(1)}$ elements, i.e, almost all x have nearly maximum complexity.*

In Information Theory, one useful result is the symmetry of information which states that, given two distributions X and Y, $I(X|Y) = I(Y|X)$, where $I(\cdot)$ is the mutual information (see). In [ZL70], it is shown that in the resource unbounded case, the symmetry of information concerning the Kolmogorov complexity also holds:

Theorem 2. (Symmetry of Information) *For all strings x and y in $\{0,1\}^n$,*

$$K(x, y) = K(x) + K(y|x) + O(\log n).$$

We will be interested in relating the existence of Kolmogorov one-way functions with polynomial time-bounded symmetry of information, which can be described by:

Hypothesis 3 (Polynomial time-bounded symmetry of information) *Let t be a polynomial. For all strings x and y in $\{0,1\}^n$,*

$$K^t(x, y) = K^t(x) + K^t(y|x) + O(\log n).$$

This conjecture is unknown to hold unconditionally, but in [LM93] and in [LW95], it is shown that:

- If **P** = **NP** then polynomial time symmetry of information holds ([LW95]);
- If deterministic one-way functions exist, then the polynomial time symmetry of information conjecture is false ([LM93,LW95]).

In [LR05], the authors explore this conjecture for other types of time-bounded complexity measures.

2.2 One-way functions

Now we present the notion of one-way function.

Definition 3. *A function f is* honest *if for some $k > 0$ and for all $x \in \{0,1\}^\star$,*

$$(|f(x)| \leq |x|^k + k) \wedge (|x| \leq |f(x)|^k + k)$$

In all definitions presented in this paper we assume that f is honest.

Definition 4 (Deterministic one-way function, [LM93]). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is a* deterministic one-way function *if the following two conditions hold:*

1. *Easy to compute: there is a (deterministic) polynomial-time algorithm A such that on every input x, the algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).*
2. *Slightly hard to invert: for any polynomial time algorithm B, for some polynomial q, for all sufficiently large n,*

$$\text{prob}_{x\in\Sigma^n}[f(B(f(x), n)) \neq f(x)] > \frac{1}{q(n)}.$$

Definition 5 (Weak one-way function). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is a* weak one-way function *if the following two conditions hold:*

1. *Easy to compute: there is a (deterministic) polynomial-time algorithm A such that on every input x, the algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).*
2. *Slightly hard to invert: there is a polynomial q such that for every probabilistic polynomial-time algorithm B and all sufficiently large n's,*

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}[f(B(f(x), r, n)) \neq f(x)] > \frac{1}{q(n)}.$$

Definition 6 (Strong one-way function). *A function $f : \{0,1\}^* \to \{0,1\}^*$ is a* strong one-way function *if the following two conditions hold:*

1. *Easy to compute: there is a (deterministic) polynomial-time algorithm A such that on every input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).*
2. *Hard to invert: for every probabilistic polynomial time algorithm B, every positive polynomial q, and every sufficiently large n,*

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}[f(B(f(x), r, n)) = f(x)] < \frac{1}{q(n)}.$$

It is easy to see that any strong one-way function is a deterministic and it is also a weak one-way function.

3 One-way functions and Kolmogorov complexity

We present some approaches to define one-way functions using Kolmogorov complexity.

3.1 An expected value approach

We first show how one-way functions are related with expected value of polynomial time-bounded Kolmogorov complexity over Σ^n. In particular, we show that if the expectation is at least larger than any constant, we have a weak one-way function but if this value is nearly maximum, then f is a strong one-way function. On the other hand, we show that if f is a strong one-way function, then the expectation must be larger than logarithmic.

Theorem 4. *Let f be an injective and polynomial time computable function. If for all polynomial t and for every constant c, the expected value of $K_f^t(x|f(x), r, n)$ over pairs $(x, r) \in \Sigma^n \times \Sigma^{\leq t(n)}$ is larger than c for sufficiently large $n's$ then f is a weak one-way function.*

Proof. Assume that f is not a weak one-way function. Then, in particular, if $q(n) = n^2$, there is an algorithm B such that $\text{prob}_{x,r}(B(f(x), r, n) \neq x) \leq 1/n^2$ for infinitely many integers n. Let t be any polynomial such that $t(n) \geq \text{time}_B(n) \log(\text{time}_B(n))$. Consider the set:

$$I = \{(x, r) \in \Sigma^n \times \Sigma^{\leq t(n)} : B(f(x), r, n) = x\}$$

Notice that if $(x,r) \in I$ then $K^t_f(x|f(x),r,n) \leq |B|$. Thus, for infinitely many n:

$$\begin{aligned}E(K^t_f(x|f(x),r,n)) &= \sum_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}} \text{prob}(x,r)\cdot K^t_f(x|f(x),r,n) = \\ &= \sum_{(x,r)\in I} \text{prob}(x,r)\cdot K^t_f(x|f(x),r,n) + \sum_{(x,r)\notin I} \text{prob}(x,r)\cdot K^t_f(x|f(x),r,n) \leq \\ &\leq \text{prob}[(x,r)\in I]\cdot|B| + \sum_{(x,r)\notin I} \text{prob}(x,r)\cdot(n+O(1)) < \\ &< |B| + \frac{1}{n^2}(n+O(1)) \in O(1)\end{aligned}$$

Thus, if $E(K^t_f(x|f(x),r,n)) > c$ is satisfied for all constants c and for almost all n, then f is a weak one-way function. □

We now present two results that give some intuition about the expectation of the Kolmogorov complexity of a strong one-way function and a weak one-way function. Notice that the result for weak one-way functions is not tight since the gap for the bound proved in the last theorem is large.

Theorem 5. *Let t be any polynomial. If f is a strong one-way function then, for every constant c and every polynomial q, $E_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}(K^t_f(x|f(x),r,n) > c\left(1-\frac{1}{q(n)}\right)\log n$ for almost all n.*

Proof. Assume, by contradiction that for some constant c and some polynomials q and t we have $E_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}(K^t_f(x|f(x)$ $c\left(1-\frac{1}{q(n)}\right)\log n$ infinitely often. By the Markov bound we have:

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}[K^t_f(x|f(x),r,n) \leq c\log n] > 1 - \frac{c\left(1-\frac{1}{q(n)}\right)\log n}{c\log(n)} = \frac{1}{q(n)}$$

We build an algorithm Q that on input $(f(x),r)$ tries to invert $f(x)$, and succeeds for the cases where $K^t_f(x|f(x),r,n) \leq c\log n$. This algorithm simply runs all programs of size up to $c\log n$ for at most t steps and with randomness r for input $f(x)$. For each such program, Q tests if the output is an inverse of $f(x)$, and if it is , it outputs that inverse. If for the pair (x,r) it happens that $K^t_f(x|f(x),r,n) \leq c\log n$, then Q will find a suitable shortest program and output the correct x. Therefore, its success probability is at least that of (x,r) satisfying this condition, that is, $\Pr\left[K^t_f(x|f(x),r,n) \leq c\log n\right] > \frac{1}{q(n)}$.

Since there are at most $2^{c\log n+1} = 2\cdot 2^{c\log n}$ programs of length at most $c\log n$ and each of them is run for a polynomial number of steps, then clearly Q runs in polynomial time.

By assumption, we know that for infinitely many $n's$,

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}[Q(f(x),r,n) = x] > \frac{1}{q(n)}$$

Thus, f is not a strong one-way function. □

Theorem 6. *Let t be any polynomial. If f is a weak one-way function then, there is a polynomial q such that, $E_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}(K^t(x|f(x),r,n) > \frac{log(n)}{q(n)}$.*

Proof. Assume, by contradiction that for every polynomial p and for some polynomial t, we have $E_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}(K^t(x|f(x)$ $\frac{log(n)}{p(n)}$ infinitely often. By the Markov bound we have:

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}[K^t(x|f(x),r,n) \leq c\log n] > 1 - \frac{\frac{log(n)}{p(n)}}{c\log n} = 1 - \frac{1}{cp(n)}$$

We consider a polynomial time algorithm Q similar with the algorithm that was used in the proof of the last theorem, that on the input $(f(x),r)$ runs all programs of length at most $c\log n$ for t steps each.

Since there are at most $2^{c\log n+1} = 2n^c$ programs of length at most $c\log n$ and each of them is run for a polynomial number of steps, then clearly Q runs in polynomial time.

By assumption, we know that for infinitely many $n's$,

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}[Q(f(x),r,n) = x] > 1 - \frac{1}{cp(n)} = 1 - \frac{1}{q(n)}$$

Thus, f is not a weak one-way function. □

3.2 An individual approach

The characterization of one-way functions proposed in the last section does not give a satisfactory insight about the security of particular individual instances of one-way functions. This issue is important in our approach to cryptography. In fact, to have an individual instance analysis of security we must have a precise control on the quantity of information that each particular instance may leak. In this section we give a notion of one-way function based on Kolmogorov complexity of particular instances.

Definition 7. *Let t be any polynomial, $f:\Sigma^*\to\Sigma^*$ be a polynomial time computable and injective function and $\delta(\cdot)$ a positive function. We say that an instance x of length n is* δ-secure *relatively to a random string $r\in\Sigma^{\leq t(n)}$ if:*

$$K^t_f(x|r,n) - K^t_f(x|f(x),r,n) \leq \delta(n).$$

Let $\varepsilon(\cdot)$ be a function. We say that f is an (ε,δ)-secure Kolmogorov one-way function *if for sufficiently large $n's$ we have:*

$$\text{prob}_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}}(x \text{ is } \delta\text{-secure for } r) \geq \varepsilon(n)$$

Intuitively, an (ε,δ)-secure Kolmogorov one-way function is a function whose probability of having an advantage of more than δ bits in the description of x by seeing $f(x)$ is not larger than $1-\varepsilon$.

Theorem 7. *Let t be any polynomial. If f is an injective and polynomial time computable function which is an $(\omega(1/n),c)$-secure Kolmogorov one-way function for every constant c, then f is a weak one-way function.*

Proof. Let t be any polynomial and consider the following sets:

$$R^- = \left\{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)} : \left(K^t_f(x|r,n) \leq n-\log n\right) \vee \left(K^t_f(x|r,f(x),n) \leq K^t_f(x|r,n)-c\right)\right\}$$

$$R^+ = \Sigma^n\times\Sigma^{\leq t(n)} \setminus R^-$$

Using a counting argument, given r, there are less than $2^{n-\log n+1}$ strings x such that $K^t_f(x|r,n) \leq n-\log n$. Thus, there are at most $2^{n-\log n+1}\times 2^{t(n)}$ pairs $(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}$ such that $K^t_f(x|r,n) \leq n-\log n$.

By the assumption of f being secure Kolmogorov one-way function, there are at most $(1-g(n))\times 2^{n+t(n)}$ pairs $(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}$ such that $K^t_f(x|r,f(x),n) \leq K^t_f(x|r,n)-c$ and where $g(n)\in\omega(1/n)$. Thus,

$$\begin{aligned}\#R^- &\leq 2^{t(n)}\cdot(2^{n-\log n+1}) + (1-g(n))\cdot 2^{n+t(n)} \\ &= 2^{n+t(n)}\left(2^{-\log n+1} + 1 - g(n)\right) \\ &= 2^{n+t(n)}\left(1-g(n)+\frac{2}{n}\right).\end{aligned}$$

The number of pairs in R^+ is at least $\left(g(n)-\frac{2}{n}\right)2^{n+t(n)}$.

Thus,

$$\begin{aligned}E(K^t_f(x|f(x),r,n)) &= \sum_{(x,r)\in\Sigma^n\times\Sigma^{\leq t(n)}} \text{prob}(x,r)\cdot K^t_f(x|f(x),r,n) \\ &\geq \sum_{(x,r)\in R^+} \text{prob}(x,r)\cdot K^t_f(x|f(x),r,n) \\ &\geq \left(g(n)-\frac{2}{n}\right)\cdot(n-\log n-c) \geq \\ &\geq g(n)\cdot(n-\log n-c) - 2\end{aligned}$$

Notice that $(n - \log n - c) \cdot g(n) - 2$ converges to infinity. Hence f, by the expected value characterization given in Theorem 4, is a weak one-way function. □

In order to avoid dealing with probabilities we can think of a different approach based on Definition 7:

Definition 8. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be an injective and polynomial time computable function. We say that f is Kolmogorov one-way function if for all polynomial t, all constant c and for all sufficiently large n and for all x of length n, $K^t_f(x|n) - K^t_f(x|f(x), n) \leq c \log n$.*

Notice that this definition is a particular case of Definition 7, where the value of ε is taken to be 1.

Proposition 1. *If f is a Kolmogorov one-way function then f is a deterministic one-way function.*

Proof. We prove this proposition by contraposition. Assume that f is not a deterministic one-way function. Thus, there is a deterministic polynomial time algorithm B such that for all polynomial q and all n_0, there is an $n \geq n_0$, for which:

$$\#\{x \in \Sigma^n : B(f(x), n) = x\} \geq 2^n - \frac{2^n}{q(n)}.$$

Thus, for an infinity of $n's$, B inverts at least one x such that $|x| = n$, $K^t_f(x|n) > \sqrt{n}$. For these x, we have that $K^t_f(x|n) > \sqrt{n}$ and $K^t_f(x|f(x), n) \leq c'$, where c' is a constant that includes the description of B. Taking those x such that for all c, $\sqrt{n} > c \log n + c'$. So, we have:

$$\begin{aligned} K^t_f(x|n) - K^t_f(x|f(x), n) &\geq \sqrt{n} - c' \\ &> c \log n + c' - c' \\ &= c \log n \end{aligned}$$

□

It is unknown if the existence of Kolmogorov one-way functions defined as in Definition 8 implies or not the existence of strong or even weak one-way functions.

4 On the Kolmogorov complexity one-way functions and the polynomial time symmetry of information conjecture

Longpré and Mocas in [LM93] and Longpré and Watanabe in [LW95] have studied the relationship between classical one-way functions and polynomial time-bounded symmetry of information conjecture.

Similarly, in this section, we explore the connection between the existence of Kolmogorov one-way functions and the polynomial time-bounded symmetry of information.

We begin by observing the following:

Proposition 2. *If there is a Kolmogorov one-way function with respect to Definition 8, then the polynomial time-bounded symmetry of information conjecture is false.*

Proof. In Proposition 1, we proved that if a Kolmogorov one-way function with respect to Definition 8 exists, then a deterministic one-way function also exists. But from [LM93] and [LW95], it is known that if the deterministic one-way functions exist then the conjecture of polynomial time-bounded symmetry of information conjecture does not hold. □

Proposition 3. *If f is an injective total function and computable in polynomial time, then for all, except a polynomial fraction of strings x, if $y = f(x)$ then:*

$$K^t_f(x||x|) \leq K^t_f(y||y|) + O(\log K^t_f(yx, |yx|))$$

Proof. Since the Kolmogorov complexity of a string can not increase by more than a constant when a polynomial time function is applied, we only need to consider the case where $K^t_f(x||x|) \leq K^t_f(y||y|)$.

Consider the strings x of length n such that $K_f(x) > n - \log n$.

Now, considering the image of those strings through f, since only a polynomial fraction of the strings of length n has Kolmogorov complexity less than $n - \log n$, and since f is injective, all except a polynomial fraction of the strings x map to strings y of high Kolmogorov complexity. For those strings, $K^t_f(x|n) - K^t_f(y||y|) \in O(\log K^t_f(yx||yx|))$. □

Theorem 8. *If polynomial time-bounded symmetry of information conjecture holds then $(1/poly, O(\log n))$-Kolmogorov one-way functions do not exist.*

Proof. If $f(x) = y$ and $|x| = n$:

- $K^t_f(yx|n, r) \leq K^t_f(x|n, r) + O(1)$ for all random string $r \in \Sigma^{poly(n)}$;
- $K^t_f(x|n, r) - K^t_f(y|n) \leq O(\log n)$ for all except a polynomial number of x and r.

Thus, by polynomial time-bounded symmetry of information, for all except a polynomial number of $x's$ and $r's$, we get:

$$\begin{aligned} K^t_f(y|n, r) + K^t_f(x|y, n, r) &\leq K^t_f(yx|n, r) + O(\log n) \\ \Leftrightarrow K^t_f(x|y, n, r) &\leq K^t_f(yx|n, r) - K^t_f(y|n, r) + O(\log n) \\ \Rightarrow K^t_f(x|y, n, r) &\leq K^t_f(x|n, r) - K^t_f(y|n, r) + O(\log n) \\ \Rightarrow K^t_f(x|y, n, r) &\leq O(\log n) + O(\log n) = O(\log n) \end{aligned}$$

Then, for all except a polynomial number of $x's$ and $r's$

$$K^t_f(x|r, n) - K^t_f(x|y, r, n) \geq K^t_f(x|r, n) - O(\log n) \geq O(\log n)$$

which implies that f is not an $(1/poly, O(\log n))$-Kolmogorov one-way function. □

References

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13(4):547–569, 1966.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[Gol01] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *SFCS '89*, pages 230–235. IEEE Computer Society, 1989.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *STOC '89*, pages 12–24. ACM, 1989.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.

[LM93] L. Longpré and S. Mocas. Symmetry of information and one-way functions. *Information processing Letters*, 46(2):95–100, 1993.

[LR05] T. Lee and A. Romashchenko. Resource bounded symmetry of information revisited. *Theor. Comput. Sci.*, 345(2-3):386–405, 2005.

[LV08] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 2008.

[LW95] L. Longpré and O. Watanabe. On symmetry of information and polynomial time invertibility. *Information and Computation*, 121(1):14–22, 1995.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC '90*, pages 387–394. ACM, 1990.

[Sol64] R. Solomonoff. A formal theory of inductive inference, part I. *Information and Control*, 7(1):1–22, 1964.

[ZL70] A. Zvonkin and L. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematics Surveys*, 256:83–124, 1970.

Depth as Randomness Deficiency

Luís Antunes*
Computer Science Department
University of Porto

Armando Matos †
Computer Science Department
University of Porto

André Souto ‡
Computer Science Department
University of Porto

Paul Vitányi §
CWI
University of Amsterdam

Abstract

Depth of an object concerns a tradeoff between computation time and excess of program length over the shortest program length required to obtain the object. It gives an unconditional lower bound on the computation time from a given program in absence of auxiliary information. Variants known as "logical depth" and "computational depth", are expressed in Kolmogorov complexity theory.

In this article we derive quantitative relation between logical depth and computational depth and unify the different "depth" notions by relating them to A. Kolmogorov and L. Levin's fruitful notion of "randomness deficiency". Subsequently, we revisit the computational depth of infinite strings, introducing the notion of *super deep* sequences and relate it with other approaches.

1 Introduction

The information contained in an individual finite object (a finite binary string) can be measured by its Kolmogorov complexity—the length of the shortest binary program that computes the object. Such a shortest program contains no redundancy: every bit is information; but is it meaningful information? If we flip a fair coin to obtain a finite binary string, then with overwhelming probability that string constitutes its own shortest description. However, with overwhelming probability, all the bits in the string are apparently meaningless information, just random noise.

The opposite of randomness is regularity; and the effective regularities in an object can be used to compress it and cause it to have lower Kolmogorov complexity. Regular objects contain laws that govern its existence and have meaning. For example, let the object in question be a book on number theory. The book will list a number of difficult theorems. However, it has very low Kolmogorov complexity since all theorems are derivable from the initial few definitions. Our estimative of the difficulty of the book is based on the fact that it takes a long time to reproduce the book from part of the information in it. We can transmit all the information in the book by just transmitting the theorems. The receiver will have to spend a long time to reconstruct the proofs and the full book. On the other hand, we can send all the book. Now the receiver has all the useful information without having to spend much time. Hence, there is a tradeoff: in both cases we send the same information in terms of Kolmogorov complexity, but in the former case it takes a long time to reconstruct it from a short message, and in the latter case it takes a short time to reconstruct it from a long message. The existence of such book is itself evidence of some long evolution preceding it.

Previous work: Bennett [Ben88] introduced the notion of logical depth of an object as the amount of time required for an algorithm to derive the object from a shorter description.

Antunes *et al.* [AFMV06] consider logical depth as one instantiation of a more general theme, computational depth, and propose several other variants based on the difference between a resource bound Kolmogorov complexity measure and the unbounded Kolmogorov complexity.

For infinite sequences, Bennett identified the classes of *weakly* and *strongly deep* sequences, and showed that the halting problem is strongly deep. Subsequently Judes, Lathrop, and Lutz [JLL94] extended Bennett's work defining the classes of *weakly useful* sequences and proved that every weakly useful sequence is strongly deep in the sense of Bennett. Fenner *et al.* [FLMR05] proved that there exist sequences that are weakly useful but not strongly useful. Lathrop and Lutz [LL99] introduced refinements (named *recursive weak depth* and *recursive strong depth*) of Bennett's notion of weak and strong depth, and studied its fundamental properties, showing that recursively weakly (resp. strongly) deep sequences form a proper subclass of the class of weakly (resp. strongly) deep sequences, and also that every weakly useful sequences is recursive strongly deep.

Randomness deficiency of a string x with respect to a probability distribution was defined and investigated by L.A. Levin [Lev74, Lev84] and by A.N. Kolmogorov. Levin [Lev74] also showed that the Kolmogorov complexity of a string x coincides up to an additive constant term with the negative logarithm of the Universal probability of x.

Results: We derive quantitative relations between the different notions of depth: logical depth and computation depth (Section 3). Subsequently, we relate the different "depth" notions to A. Kolmogorov and L. Levin's fruitful notion of "randomness deficiency" (Section 4). We study the common information for infinite strings (Section 5) and finally, based on the connection between randomness deficiency and computational depth, we introduce and study the notion of super deep for infinite sequences (Section 6).

2 Preliminaries

We briefly introduce some notions from Kolmogorov complexity, mainly the standardize notation. We refer to the textbook by Li and Vitányi [LV97] for more details. Let U be a fixed universal Turing machine. For technical reasons we choose one with a separated read-only input tape, that is scanned without backing up, a separate work tape on which the computation takes place, and a separate output tape. Upon halting, the initial segment p of the input that has been scanned is called "program" and the contents of the output tape is called "output". By construction, the set of halting programs is prefix free. We call U the reference universal

*Email: lfa@ncc.up.pt. Web: http://www.ncc.up.pt/~lfa. Adress: Departamento de Ciência de Computadores. Rua Campo Alegre, 1021/1055, 4169 - 007 PORTO, PORTUGAL

†Email: acm@ncc.up.pt. Web: http://www.ncc.up.pt/~acm.

‡Email: andresouto@dcc.fc.up.pt. Web: http://www.ncc.up.pt/~andresouto.The author is supported by the grant SFRH / BD / 28419 / 2006 from FCT

§Email: Paul.Vitanyi@cwi.nl. Web: http://homepages.cwi.nl/~paulv/.

prefix machine. In the rest of this paper we denote the n-prefix of an infinite sequence α by α_n and the i^{th}-bit by α^i.

Definition 2.1 (i) The *Kolmogorov complexity* of a finite binary string x is defined as

$$K(x) = \min_p\{|p| : U(p) = x\},$$

where p is a program, and the *Universal a priori probability* of x is

$$Q_U(x) = \sum_{U(p)=x} 2^{-|p|}.$$

(ii) A time-constructible function t from natural numbers to natural numbers is a function with the property that $t(n)$ can be constructed from n by a Turing machine in time of order $O(t(n))$. For every time-constructible t, the *t-time-bounded Kolmogorov complexity* of x is defined as

$$K^t(x) = \min_p\{|p| : U(p) = x \text{ in at most } t(|x|) \text{ steps}\},$$

and the *t-time bounded Universal a priori probability* is defined as

$$Q^t_U(x) = \sum_{U^t(p)=x} 2^{-|p|},$$

and $U^t(p) = x$ means that U computes x in at most $t(|x|)$ steps and halts.

A different universal Turing machine may affect the program size $|p|$ by at most a constant additive term, and the running time t by at most a logarithmic multiplicative factor. The same will hold for all other measures we will introduce.

Levin [Lev74] showed that the Kolmogorov complexity of a string x coincides up to an additive constant term with the logarithm of $1/Q_U(x)$. This result is called the "Coding Theorem" since it shows that the shortest upper semi-computable code is a Shannon-Fano code of the greatest lower semi-computable probability mass function. In order to state formally the Coding theorem we need to introduce the notion of universal enumerable discrete semimeasure.

Definition 2.2 *There exists a universal enumerable discrete semimeasure, which we denote by* $\mathbf{m}$. *It is a discrete semimeasure that multiplicatively dominates every other enumerable discrete semimeasure.*

Theorem 2.3 (Coding Theorem) *For every* $x \in \{0,1\}^n$,

$$K(x) = -\log Q_U(x) + O(1) = -\log \mathbf{m}(x).$$

Hence, if x has high probability because it has many long descriptions then it must have a short description too.

We refer to *mutual information* of two finite strings as

$$I(x : y) = K(x) + K(y) - K(x, y).$$

Notice that the mutual information is symmetric, i.e., $I(x : y) = I(y : x)$.

3 Depth

Bennett [Ben88] defines the b-significant logical depth of an object x as the time required by the reference universal Turing machine to generate x by a program that is no more than b bits longer than the shortest descriptions of x. Bennett uses time as the number of steps; instead, we generalize the definition to consider the number of steps $t(|x|)$, where t is a time-constructible function. We will use a more general definition in the later theorems.

Definition 3.1 (Logical Depth) The logical depth of a string x at a significance level b is

$$\text{ldepth}_b(x) = \min\left\{t(|x|) : \frac{Q^t_U(x)}{Q_U(x)} \geq 2^{-b}\right\},$$

where the minimum is taken over all time constructible t.

Given a significance level b, the logical depth of a string x is the minimal running time $t(|x|)$, such that programs running in at most $t(|x|)$ steps account for approximately a $1/2^b$ fraction of x's universal probability.

In fact, with some probability we can derive the string by simply flipping a coin. But for long strings this probability is exceedingly small. If the string has a short description then we can flip that with higher probability. Bennett proposal tries to express the tradeoff between the probability of flipping a short program and the shortest computation time from program to object.

Antunes *et al.* [AFMV06] developed the notion of computational depth in order to capture the tradeoff between the amount of help bits required and the reduced computation time to compute a string. The concept is simple: they consider the difference of two versions of Kolmogorov complexity measures.

Definition 3.2 (Basic Computational Depth) Let t be a time constructible function. For any finite binary string x we define

$$\text{depth}^t(x) = K^t(x) - K(x).$$

Bennett is not very precise, but from his original paper [Ben88] we can conclude that he intends that a string x to be $(t(|x|), b)$-deep iff $t(|x|)$ is the least number of steps to compute x from a program of length at most $K(x) + b$. Then, it is straightforward that $\text{depth}^t(x) = K^t(x) - K(x)$ iff x is $(t(|x|), K^t(x) - K(x))$-deep. The relations between Bennett's logical depth and the computational depth follow.

The proof of Item (ii) below uses an idea contained in the proof of Theorem 7.7.1 of [LV97]. Define $K(t)$ where t is a time-constructible function, as $\min_i\{i : T_i \text{ computes } t(\cdot)\}$, where $T_1, T_2, \ldots$ is the standard enumeration of all Turing machines.

Theorem 3.3 *Let t be a time-constructible function.*
(i) If b is the minimum value such that $\text{ldepth}_b(x) = t(|x|)$, *then* $\text{depth}^t(x) \geq b + O(1)$.
(ii) If $\text{depth}^t(x) = b$, *then* $\text{ldepth}_{b+\min\{K(b),K(t)\}+O(1)}(x) \geq t(|x|)$.

Proof. (i) Assume, $\text{ldepth}_b(x) = t(|x|)$. So

$$\frac{Q_U^t(x)}{Q_U(x)} \geq 2^{-b},$$

with $t(|x|)$ least. Assume furthermore that b is the least integer so that the inequality holds for this $t(|x|)$. We also have

$$\frac{Q_U^t(x)}{Q_U(x)} \geq \frac{2^{-K^t(x)}}{Q_U(x)} = 2^{-(K^t(x)-K(x)-O(1))} = 2^{-b-\Delta},$$

where $b+\Delta = K^t(x)-K(x)-O(1)$. The first inequality holds since the sum $Q_U^t(x)$ comprises a term $2^{-K^t(x)}$ based on a shortest program of length $K^t(x)$ computing x in at most $t(|x|)$ steps. Since b is the least integer, it follows that $\Delta \geq 0$. Since $\text{depth}^t(x) = K^t(x) - K(x)$, we find that $\text{depth}^t(x) \geq b + O(1)$.

(ii) Assume that $\text{depth}^t(x) = b$, that is, x is $(t(|x|), b)$-deep. We can enumerate the set S of all programs computing x in time at most $t(|x|)$ by simulating all programs of length $l \leq |x| + 2\log|x|$ for $t(|x|)$ steps. Hence, the shortest such program q enumerating S has length $|q| \leq K(x,t) + O(1)$. But we achieve the same effect if, given x and b we enumerate all programs of length l as above in order of increasing running time and stop when the accumulated algorithmic probability exceeds $2^{-K(x)+b}$. The running time of the last program is $t(|x|)$. (This shows that $K(t,x) \leq K(b,x) + O(1)$, not $K(t) \leq K(b) + O(1)$). The shortest program r doing this has length $|r| \leq K(x,b) + O(1)$. Hence, $K(S) \leq \min\{K(x,t), K(x,b)\}$. By definition, $Q_U^t(x) = \sum_{p \in S} 2^{-|p|}$. Assume, by way of contradiction, that

$$\frac{Q_U^t(x)}{Q_U(x)} < 2^{-b-\min\{K(b),K(t)\}-O(1)}$$

Since $Q_U(x) = 2^{-K(x)-O(1)}$, we have

$$Q_U^t(x) < 2^{-K(x)-b-\min\{K(b),K(t)\}-O(1)}$$

Denote $m = K(x) + b + \min\{K(b), K(t)\} + O(1)$. Since S is a prefix-free $\sum_{p \in S} 2^{-|p|} < 2^{-m}$. Now every string in S can be effectively compressed by at least $m - K(S) - O(1)$ bits. Namely,

$$\sum_{p \in S} 2^{-|p|+m} < 1$$

Therefore, the elements of S can be coded by the Shannon-Fano code with the code word length for p at most $|p| - m + 2$. In order to make this coding effective, we use a program of length $K(S)$ to enumerate exactly the strings of S. This takes an additional $K(S)+O(1)$ bits in the code for each $p \in S$. This way, each $p \in S$ is effectively compressed by $m - K(S) - O(1)$ bits. Therefore, each $p \in S$ can be compressed by at least $K(x) + b + \min\{K(b), K(t)\} - \min\{K(x,t), K(x,b\}$ bits, up to an additive constant we can set freely, and hence by more than b bits which is a contradiction. Hence,

$$\frac{Q_U^t(x)}{Q_U(x)} \geq 2^{-b-\min\{K(t),K(b)\}-O(1)}$$

which proves (ii). ◇

4 A Unifying Approach

Logical depth and computational depth are all instances of a more general measure, namely the randomness deficiency of a string x with respect to a probability distribution, Levin [Lev74, Lev84]. For us, with some abuse of notation, see [LV97], a function $\mu : \{0,1\}^* \to \mathcal{R}$ defines a *probability measure* or *measure* for short, if

$$\begin{aligned} \mu(\epsilon) &= 1, \\ \mu(x) &= \sum_{a \in \{0,1\}} \mu(xa). \end{aligned}$$

Definition 4.1 Let μ be a computable measure. Then,

$$\delta(x \mid \mu) = \left\lfloor \log \frac{Q_U(x)}{\mu(x)} \right\rfloor$$

is the *randomness deficiency*[1] of x with respect to μ. Here Q_U is the universal a priori probability of Definition 2.1.

Note that $Q_U(x)$ is of exact order of $2^{-K(x)}$ by the Coding Theorem 2.3, i.e., up to multiplicative terms $Q_U(x)$ and $2^{-K(x)}$ are equal. (In the literature, see for example [LV97], $\mathbf{m}(x) = 2^{-K(x)}$ is used instead of $Q_U(x)$, and it is straightforward that this is equivalent up to a multiplicative independent constant by the Coding Theorem.)

Levin [Lev74, Lev84] showed that the randomness deficiency of x with respect to μ is the largest, within an additive constant, randomness μ-test for x. So $\delta(x \mid \mu)$ is, in a sense, a universal characterization of "non-randomness", "useful" or "meaningful" information in a string x with respect to a probability distribution μ.

- Randomness deficiency is "almost non-negative" in the sense that $\delta(x \mid \mu) \geq C$ for some constant C (possibly negative) and all x with $\mu(x) > 0$. Indeed, every such element x can be described by its $\log 1/\mu(x)$-bit Shannon-Fano code word conditional to μ. Thus $K(x \mid \mu) \leq \log 1/\mu(x) + O(1)$.
- For all computable measure μ, the randomness deficiency of almost all elements of positive μ-measure is small: the μ-probability concentrated on x's with $\mu(x) > 0$ and $\delta(x \mid \mu) > \beta$ is less than $2^{-\beta}$. Indeed, $\delta(x \mid \mu) > \beta$ implies that $K(x \mid \mu) < \log 1/\mu(x) - \beta$. Since there are at most $2^{\log 1/\mu(x)-\beta}$ programs with less than $\log 1/\mu(x) - \beta$ bits, the number of x's satisfying the inequality cannot be larger.
- Every element with small randomness deficiency with respect to μ possesses every simply describable property of a set of elements of positive μ-probability, on which the majority of μ-probability is concentrated. We identify a property of elements of positive μ-probability with a subset consisting of all elements having that property. More specifically, assume that P is a subset with $\mu(P) \geq 1 - 2^{-\beta}$ (majority of μ-positive elements possess the property P) and $K(P \mid \mu) \leq \gamma$ (the property P has a short description). Then the randomness deficiency of all $x \notin P$ of positive μ-probability satisfies

[1] $\lfloor r \rfloor$ denotes the integer part of r and $\lceil \alpha \rceil$ denotes the smallest integer bigger than α.

$\delta(x \mid \mu) > \beta - \gamma - O(\log|x|)$, which is large if β is large and γ is small. Indeed, ignoring the logarithmic terms, we have $K(x \mid \mu) \leq \log|P| + K(P \mid \mu) \leq \log 1/\mu(x) - \beta + \gamma$.

The randomness deficiency measures our disbelief that x can be obtained by random sampling from μ. By the second property, with high probability, the randomness deficiency of an element randomly drawn from μ is small. On the other hand, if $\delta(x \mid \mu)$ is small, there is no way to refute the hypothesis that x was obtained by μ-random sampling: every such refutation is based on a simply described property possessed by a μ-majority but not by x. At this point, it is important to consider only simply describable properties, as otherwise we can refute the hypothesis by exhibiting the property $P = \{y : \mu(y) > 0\} \setminus \{x\}$.

We now observe that logical depth and computational depth of a string x equals the randomness deficiency of x with respect to the measures $Q^t(x) = \sum_{U^t(p)=x} 2^{-|p|}$ and $2^{-K^t(x)}$ respectively. The proofs follow directly from the definitions.

Lemma 4.2 *Let x be a finite binary string and let t be a time-constructible function.*
(i) $\mathrm{ldepth}_b(x) = \min\{t : \delta(x \mid Q^t) \leq b\}$.
(ii) $\mathrm{depth}^t(x) = \delta(x \mid \mathbf{m}^t)$ *where* $\mathbf{m}^t(z) = 2^{-K^t(z)}$.

5 On the information of infinite strings

Based on the unification of depth concepts for finite strings, we extend these ideas for infinite sequences. In order to motivate our approach we start by introducing Levin's notion of randomness deficiency for infinite sequences.

Definition 5.1 (Levin) *The value $D(\alpha/\mu) = \lfloor \log \sup(\mathbf{m}(\alpha_n)^*/\mu(\alpha_n)) \rfloor$ is called the randomness deficiency of α with respect to the semi-measure μ. Here $\mathbf{m}(\alpha_n)^*$ is the density probability function of $\mathbf{m}(\alpha_n)$.*

Let α and β be two sequences and $\mathbf{m} \otimes \mathbf{m}$ be defined by $\mathbf{m} \otimes \mathbf{m}(\alpha, \beta) = \mathbf{m}(\alpha)\mathbf{m}(\beta)$.

Definition 5.2 (Levin) *The value $I(\alpha : \beta) = D((\alpha, \beta)/\mathbf{m} \otimes \mathbf{m})$ is called the amount of information in α about β or the deficiency of their independence.*

This definition is equivalent to the mutual information $I(\alpha : \beta) = \sup_n I(\alpha_n : \beta_n)$.

Let α and γ be two random infinite and independent strings (in the sense that their prefixes are independent). Consider the following sequence β

Example 5.3

$$\beta = \alpha^1\gamma^1\alpha^2\gamma^2\ldots$$

By Definition 5.2 we have

$$\begin{aligned} I(\alpha : \beta) &= \sup_n I(\alpha_n : \beta_n) \\ &= \sup_n (K(\alpha_n) + K(\beta_n) - K(\alpha_n, \beta_n)) \\ &\geq \sup_n (n + n - (n + n/2)) = \infty. \end{aligned}$$

As $I(\beta : \alpha) = I(\alpha : \beta)$ then $I(\beta : \alpha) = \infty$.

However, intuitively β contains more information about α than the other way around, since from the sequence β we can totally reconstruct α but from α we can only recover half of β, namely, the bits with odd indexes.

This seems to be a lacuna in Definition 5.2. The definition says more when the information is finite but that is precisely when we do not need an accurate result. Notice that if the sequences are finite we can argue that they are independent. In the infinite case, one should be able to classify the cases where the mutual information is infinite. Two infinite sequences may have infinite mutual information and yet infinite information still lacks in order to reconstruct each of them. In the previous example α fails to provide all the information of β related to γ, which has infinite information. In this section we will present two approaches to reformulate the definition of "mutual information" in order to fulfill our intuition. In order to have a proportion of information as the prefixes grow we need to do some normalization in the process.

5.1 The Mutual Information Point of View

We are looking for a normalized mutual information measure I_m that applied to Example 5.3 gives

$$I_m(\alpha : \alpha) = 1; \quad I_m(\alpha : \beta) = 1/2; \quad I_m(\beta : \alpha) = 1; \quad I_m(\beta : \beta) = 1$$

Contrarily to the definition of Levin of infinite mutual information, and accordingly to our intuition, the above conditions imply that the normalized version must be non-symmetric.

Definition 5.4 (First attempt) *Given two infinite sequences α and β the normalized mutual information that β has about α is defined as*

$$I_m(\beta : \alpha) = \lim_{n\to\infty} \lim_{m\to\infty} \frac{I(\beta_m : \alpha_n)}{I(\alpha_n : \alpha_n)}$$

The major drawback of this definition is the fact that the limit does not always exist.[2] However, it does exist for the Example 5.3 with the desired properties. Furthermore, we obtain for the same α and β

$$I_m(\alpha : \alpha) = 1; \qquad I_m(\beta : \beta) = 1;$$

$$I_m(\alpha : \beta) = \lim_{n\to\infty} \lim_{m\to\infty} \frac{m + n - (m + n - n/2)}{n} = \frac{1}{2};$$

$$I_m(\beta : \alpha) = \lim_{n\to\infty} \lim_{m\to\infty} \frac{m + n - m}{n} = 1;$$

[2] Notice that there are sequences α for which $\lim_n \frac{n}{K(\alpha_n)}$ does not exist.

Definition 5.5 (Normalized mutual information for infinite sequences) *Given two infinite sequences α and β we define the lower normalized mutual information that β has about α as*

$$I_{m*}(\beta:\alpha) = \liminf_{n\to\infty} \lim_{m\to\infty} \frac{I(\beta_m:\alpha_n)}{I(\alpha_n:\alpha_n)}$$

and the upper normalized mutual information that β has about α as

$$I^*_m(\beta:\alpha) = \limsup_{n\to\infty} \lim_{m\to\infty} \frac{I(\beta_m:\alpha_n)}{I(\alpha_n:\alpha_n)}$$

Notice that these definitions also fulfill the requirements presented in the beginning of this section with respect to the example 5.3.

We now can define independence with respect to normalized mutual information:

Definition 5.6 *Two sequences, α and β, are independent if $I^*_m(\alpha:\beta) = I^*_m(\beta:\alpha) = 0$.*

In [Lut00, Lut02], the author developed a constructive version of Hausdorff dimension. That dimension assigns to every binary sequence α a real number $\dim(\alpha)$ in the interval $[0,1]$. Lutz claims that the dimension of a sequence is a measure of its information density. The idea is to differentiate sequences by non-randomness degrees, namely by their dimension. Our approach is precisely to introduce a measure of density of information that one sequence has about the other, in the total amount of the other's information. So we differentiate non-independent sequences, by their normalized mutual information.

Mayordomo [May02] redefined constructive Hausdorff dimension in terms of Kolmogorov complexity.

Theorem 5.7 (Mayordomo) *For every sequence α,*

$$\dim(\alpha) = \liminf_{n\to\infty} \frac{K(\alpha_n)}{n}$$

So, now the connection between constructive dimension and normalized information measure introduced here is clear. It is only natural to accomplish results about the Hausdorff constructive dimension of a sequence, knowing the dimension of another, and their normalized information.

Lemma 5.8 *Let α and β be two infinite sequences. Then*

$$I^*_m(\alpha:\beta)\cdot\dim(\beta) \geq \dim(\alpha) + \liminf_{n\to\infty} -\frac{K(\alpha_n|\beta_n)}{n}$$

Proof.

$$\begin{aligned}
I^*_m(\alpha,\beta)\cdot\dim(\beta) &= \limsup_n \lim_m \frac{I(\alpha_m:\beta_n)}{I(\beta_n:\beta_n)} \cdot \liminf_n \frac{K(\beta_n)}{n} \\
&\geq \liminf_n \liminf_m \frac{I(\alpha_m:\beta_n)}{n} \\
&\geq \liminf_n \liminf_m \frac{I(\alpha_m:\beta_n)}{m} \\
&\geq \liminf_n \liminf_m \frac{K(\alpha_m) - K(\alpha_m|\beta_m)}{m} \\
&\geq \liminf_m \frac{K(\alpha_m)}{m} + \liminf_m \frac{-K(\alpha_m|\beta_m)}{m} \\
&= \dim(\alpha) + \liminf_m -\frac{K(\alpha_m|\beta_m)}{m} \quad \diamond
\end{aligned}$$

We present now the time bounded version of $\dim(\alpha)$. This definition will be important later on this paper.

Definition 5.9 *The t-bounded dimension of an infinite sequence α is defined as*

$$\dim^t(\alpha) = \liminf_{n\to\infty} \frac{K^t(\alpha_n)}{n}$$

5.2 The Hausdorff constructive dimension point of view

In this subsection we define the common information between two sequences based on Hausdorff constructive dimension and establish a connection to it.

Definition 5.10 *The dimensional mutual information of the sequences α and β is defined as*

$$I_{dim}(\alpha,\beta) = \dim(\alpha) + \dim(\beta) - 2\dim\langle\alpha,\beta\rangle$$

This measure of mutual information is symmetric. The definition considers twice $\dim\langle\alpha,\beta\rangle$ because when encoding the prefixes α_n and β_n the result is a $2n$-length string. Notice that,

$$\begin{aligned}
I_{dim}(\alpha,\beta) &= \dim(\alpha) + \dim(\beta) - 2\dim\langle\alpha,\beta\rangle \\
&= \liminf_{n\to\infty} \frac{K(\alpha_{n/2})}{n/2} + \liminf_{n\to\infty} \frac{K(\beta_{n/2})}{n/2} - 2\liminf_{n\to\infty} \frac{K(\langle\alpha,\beta\rangle_n)}{n} \\
&\leq \liminf_{n\to\infty} \frac{K(\alpha_{n/2}) + K(\beta_{n/2}) - K(\alpha_{n/2},\beta_{n/2})}{n/2} \\
&= \liminf_{n\to\infty} \frac{I(\alpha_n:\beta_n)}{n} \\
&\leq \liminf_{n\to\infty} \frac{I(\alpha_n:\beta_n)}{K(\beta_n)} \\
&\leq \liminf_{n\to\infty} \lim_{m\to\infty} \frac{I(\alpha_m:\beta_n)}{K(\beta_n)} \\
&= I_{m*}(\alpha:\beta)
\end{aligned}$$

The third inequality is true because:

$$I(\beta_n:\alpha_m) = K(\beta_n) - K(\beta_n|\alpha_m) \geq K(\beta_n) - K(\beta_n|\alpha_n) = I(\beta_n:\alpha_n).$$

By the symmetry of the definition we also have that $I_{dim}(\alpha,\beta) \leq I_{m*}(\beta:\alpha)$. These two facts prove the following lemma:

Lemma 5.11 *Let α and β be two sequences. Then*

$$I_{dim}(\alpha,\beta) \leq \min(I_{m*}(\alpha:\beta), I_{m*}(\beta:\alpha))$$

One can easily modify the definitions introduced in this section by considering the limits when n goes to the length of the string, or the maximum length of the strings being considered. One should also notice that when x and y are finite strings and $K(y) \geq K(x)$, $I_{m*}(x:y)$ is $1 - d(x,y)$, where $d(x,y)$ is the normalized information distance studied in [Li03].

6 Depth of infinite strings

We now study depth of infinite sequences and prove some results concerning these definitions. To motivate our definitions we recall the definitions of the classes of weakly (vs. strongly deep) sequences and weakly useful (vs. strongly useful) sequences.

Definition 6.1 ([Ben88]) *An infinite binary sequence α is defined as*

- *weakly deep if it is not computable in recursively bounded time from any algorithmically random infinite sequence.*
- *strongly deep if at every significance level b, and for every recursive function t, all but finitely many initial segments α_n have logical depth exceeding $t(n)$.*

Definition 6.2 ([FLMR05]) *An infinite binary sequence α is defined as*

- *weakly useful if there is a computable time bound within which all the sequences in a non-measure 0 subset of the set of decidable sequences are Turing reducible to α.*
- *strongly useful if there is a computable time bound within which every decidable sequence is Turing reducible to α.*

The relation between logical depth and usefulness was studied by Juedes, Lathrop and Lutz [JLL94] who defined the conditions for weak and strong usefulness and showed that every weakly useful sequence is strongly deep. This result generalizes Bennett's remark on the depth of diagonal of the halting problem, strengthening the relation between depth and usefulness. Latter Fenner *et al.* [FLMR05] proved the existence of sequences that are weakly useful but not strongly useful.

The Hausdorff constructive dimension has a close connection with the information theories for infinite strings studied before, see for example [FLMR05], [Lut00], [Lut02] and [May02]. Therefore, in this section we define the dimensional computational depth of a sequence in order to study the nonrandom information on a infinite sequence.

Definition 6.3 *The dimensional depth of a sequence α is defined as*

$$\mathrm{depth}^t_{\mathrm{dim}}(\alpha) = \liminf_{n\to\infty} \frac{\delta(\alpha_n/2^{-K^t(\alpha_n)})}{n}.$$

Lemma 6.4

$$depth^t_{\mathrm{dim}}(\alpha) \leq \dim^t(\alpha) - \dim(\alpha)$$

Proof.

$$\begin{aligned}\mathrm{depth}^t_{\mathrm{dim}}(\alpha) &= \liminf_{n\to\infty} \frac{\delta(\alpha_n/2^{-K^t(\alpha_n)})}{n} \\ &= \liminf_{n\to\infty} \frac{K^t(\alpha_n) - K(\alpha_n)}{n} \\ &\leq \dim^t(\alpha) - \dim(\alpha).\end{aligned}$$

The last inequality holds since the sequence of values $K(\alpha_n)/n$ is non negative and then $\liminf_n -K(\alpha_n)/n \leq -\liminf_n K(\alpha_n)/n$. ◇

Now, in the definition of strongly deep sequences, instead of considering a fixed significance level s we consider a significance level function $s : \mathbb{N} \to \mathbb{N}$. Naturally, we want $s(n)$ to grow very slowly so we assume for example that $s = o(n)$. With this replacement we obtain a tighter definition as deepness decreases with the increase of the significance level.

Definition 6.5 *A sequence is called super deep if for every significance level function $s : \mathbb{N} \to \mathbb{N}$, such that $s = o(n)$, and for every recursive function $t : \mathbb{N} \to \mathbb{N}$, all but finitely many initial segments α_n have logical depth exceeding $t(n)$.*

We have already characterized super deep sequences using their dimensional depth in theorem 3.3. In fact we have

$$\mathrm{ldepth}_b(x) = t(|x|), \text{ with } b \text{ minimal} \Rightarrow \mathrm{depth}^t(x) \geq b + O(1)$$

Theorem 6.6 *A sequence α is super deep if and only if $\mathrm{depth}^t_{\mathrm{dim}}(\alpha) > 0$ for all recursive time bound t.*

Proof. Let α be a super deep sequence. Then for every significance level function s, such that $s = o(n)$ and every recursive function t we have that for almost all n, $\mathrm{ldepth}_{s(n)}(\alpha_n) > t(n)$. Then

$$\mathrm{depth}^{t(n)}(\alpha_n) > s(n).$$

Now if for some time bound g, $\mathrm{depth}^g_{\mathrm{dim}}(\alpha) = 0$ then there exists a bound S, such that $S = o(n)$, and, infinitely often

$$\mathrm{depth}^{g(n)}(\alpha_n) < S(n).$$

This is absurd and therefore for all recursive time bound t, $\mathrm{depth}^t_{\mathrm{dim}}(\alpha) > 0$.

Conversely if $\mathrm{depth}^t_{\mathrm{dim}}(\alpha) > 0$ then there is some $\epsilon > 0$ such that for almost all n, $\mathrm{depth}^{t(n)}_{\mathrm{dim}}(\alpha_n) > \epsilon n$. This implies that

$$\mathrm{ldepth}_{s(n)}(\alpha_n) > \mathrm{ldepth}_{\epsilon n}(\alpha_n) > t(n)$$

for all significance function $s = o(n)$ and almost all n. So α is super deep. ◇

In [JLL94] several characterizations of strong computational depth are obtained. We can prove analogous characterizations for super deepness.

Theorem 6.7 *For every sequence α the following conditions are equivalent.*

1. *α is super deep;*
2. *For every recursive time bound $t : \mathbb{N} \to \mathbb{N}$ and every significance function $g = o(n)$, $\mathrm{depth}^t(\alpha_n) > g(n)$ for all except finitely many n;*
3. *For every recursive time bound $t : \mathbb{N} \to \mathbb{N}$ and every significance function $g = o(n)$, $Q(\alpha_n) \geq 2^{g(n)}Q^t(\alpha_n)$ for all except finitely many n;*

In [JLL94] the authors proved that every weakly useful sequence is strongly deep. Following the ideas in [JLL94] we can also prove that every weakly useful sequence is super deep.

Theorem 6.8 *Every weakly useful sequence is super deep.*

Corollary 6.9 *The characteristic sequences of the halting problem and the diagonal halting problem are super deep.*

Acknowledgement

We thank Harry Buhrman, Lance Fortnow, and Ming Li for comments and suggestions.

References

[Stock85] L. Stockmeyer, On approximation algorithms for #P, SIAM J. Computing 14 (1985), 849-861

[ACMV07] L. Antunes and A. Costa and A. Matos and P. Vitnyi. Computational Depth: A Unifying Approach. Submitted, 2007.

[AFPS06] L. Antunes, L. Fortnow, A. Pinto, and A. Souto. Low-depth witnesses are easy to find. Technical Report TR06-125, *ECCC*, 2006.

[AF05] L. Antunes and L. Fortnow. Time-bounded universal distributions. Technical Report TR05-144, *ECCC*, 2005.

[AFMV06] L. Antunes and L. Fortnow and D. van Melkebeek and N. V. Vinodchandran. Computational depth: concept and applications. *Theor. Comput. Sci.*, 354 (3): 391–404, 2006.

[Ben88] C. H. Bennett. Logical depth and physical complexity. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 227–257. Oxford University Press, 1988.

[FLMR05] Stephen A. Fenner and Jack H. Lutz and Elvira Mayordomo and Patrick Reardon. Weakly useful sequences. *Information and Computation* 197 (2005), pp. 41-54.

[Gac74] P. Gcs. On the symmetry of algorithmic information. *Soviet Math. Dokl.*, 15 (1974) 1477–1480.

[JLL94] David W. Juedes, James I. Lathrop and Jack H. Lutz. Computational Depth and Reducibility. *Theoret. Comput. Sci.* 132 (1994), 37-70.

[LL99] James I. Lathrop and Jack H. Lutz. Recursive computational depth. *Information and Computation* 153 (1999), pp. 139-172.

[Lev73] Leonid A. Levin. Universal Search Problems. *Problems Inform. Transmission*, 9(1973), 265-266.

[Lev74] Leonid A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Probl. Inform. Transm.*, vol. 10, pages 206–210, 1974.

[Lev80] Leonid A. Levin. A concept of independence with applications in various fields of mathematics MIT, Laboratory for Computer Science, 1980.

[Lev84] Leonid A. Levin. Randomness conservation inequalities: information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.

[Li03] Ming Li and Xin Chen and Xin Li and Bin Ma and Paul M.B. Vitányi The similarity metric. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[LV97] Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition, 1997.

[Lut00] J. H. Lutz. Dimension in complexity classes. *Proceedings of the 15th IEEE Conference of Computational Complexity*, IEEE Computer Society Press, 2000.

[Lut02] J. H. Lutz. The dimensions of individual strings and sequences. *Technical Report cs.CC/0203017, ACM Computing Research Repository*, 2002.

[May02] E. Mayordromo. A Kolmogorov complexity characterization of construtive Hausdorff dimension. *Information Processing Letters*, 84:1-3, 2002.

On the largest monochromatic combinatorial rectangles with an application to Communication Complexity

Armando Matos[1,3,⋆]
Andreia Teixeira[1,2,⋆⋆]
André Souto[1,2,⋆⋆⋆]

[1] Universidade Porto
[2] Instituto de Telecomunicações
[3] LIACC

Abstract. The concept of combinatorial rectangle is of fundamental importance in Communication Complexity.
Given a finite set R, a combinatorial rectangle is a set of the form $A \times B$ where A and B are subsets of R. A square $R \times R$ together with a function $c\colon R \times R \to \{0,1\}$ is called a colored square. A colored square is called random if each $c(a,b)$ is an independent random variable; in this case, c is called a random function. A colored combinatorial rectangle $A \times B$ is called monochromatic if $c(a,b)$ has the same value (either 0 or 1) for every $a \in A$ and $b \in B$.
We prove that the decision problem associated with the problem of finding the maximum area of a monochromatic combinatorial rectangle (mcr) of a given colored square is NP-complete.
Most of this paper deals with the asymptotic maximum area of the mcrs of a random square. Let p be the probability that any cell has the value 1. We establish several improvements and generalizations of the results previously known, namely: (i) better bounds for several statistical parameters associated with the asymptotic maximum areas of mcrs; (ii) the generalization for arbitrary values of p of the results obtained for $p = 1/2$, the only value previously studied. These results can be useful in Information Theory and Communication Complexity; we use them to obtain a lower bound of the communication complexity of "random functions".

Keywords: Combinatorial problems, Communication Complexity.

1 Introduction

A fundamental concept in Communication Complexity is the combinatorial rectangle; for instance, a deterministic protocol for the computation of a function by two communicating parties corresponds to a partition of a "colored square" into monochromatic combinatorial rectangles [3].

Given a finite set R, a combinatorial rectangle is a set $A \times B$ where A and B are subsets of R; its area is $|A| \cdot |B|$. A square $R \times R$ together with a function $c\colon R \times R \to \{0,1\}$ is called a colored square. A combinatorial rectangle $A \times B$ is called monochromatic ("mcr") if $c(a,b)$ has the same value (0 or 1; the corresponding rectangles are denoted by "0-mcr" and "1-mcr") for every $a \in A$ and $b \in B$.

We study the decision problem "given a colored square Q and an integer k, does Q contain a mcr with area at least k?" and prove its NP-completeness. It is interesting to notice that the corresponding problem for *geometrical* rectangles can easily be solved in polynomial time; to see this, it is enough to observe that the total number of geometrical rectangles is polynomial in $|R|$.

Most of this paper is a study of mcrs in squares randomly colored with 2 colors, say 0 and 1. We prove lower and upper asymptotic bounds for the maximum area of the mcrs belonging to such squares, improving and generalizing the results previously proved in [1]. Notice that when studying areas, one must take into account not only how that area can be decomposed as a product of the lengths of the 2 sides, but also the different ways in which a given number of rows or columns can be selected in the square.

Let p be the probability that any specific cell of the square has the color 1 and let n be the side of the square. The special but important case of $p = 1/2$, is studied first; it corresponds to the "maximum randomness" of the function being studied. It is proved that, if both sides of a combinatorial rectangle exceed $2\log n$, it is almost sure that the rectangle is not monochromatic; thus, in this case, the maximum area of a mcr is very small, namely it does not exceed $4\log^2 n$.

We then allow one of the sides to be smaller than $2\log n$ and discover that mcrs with maximum area (which is $n/2 + o(n)$) have a side with a length which is either 1 or 2. However, due to the fact that the standard deviation is different for those two cases, the largest mcrs turn out to have height (or width) 2; the length of the other side is then $\frac{n}{4} + o(n)$. The main results for the case $p = 1/2$ are Theorems 2 and 3.

These results are then generalized for an arbitrary value of p. First we prove that no mcrs exist with both sides larger than $(2/\log(1/p))\log n$ and from this fact we obtain an upper bound for the area. Then we allow one of the sides to be smaller than $(2/\log(1/p))\log n$ and discover again that the maximum area is obtained for rectangles with a height which is independent of n but dependent on p. The main results for the general case are Theorems 4 and 5. These results are then used to obtain a lower bound for the deterministic communication complexity of the "random function", see Theorem 6; for $p = 1/2$ this bound is necessarily almost optimal once it almost equals the trivial upper bound $\lceil \log(n) + 1 \rceil$, see [3].

2 Preliminaries

We present the basic definitions and results that will be needed later in this paper. The reader can consult [2] for more details. For the basic concepts of Communication Complexity we refer the reader to the first two chapters of [3].

The class of problems decidable in polynomial time by a deterministic (respectively non-deterministic) Turing machine is called P (respectively NP). The decision problem A *reduces polynomially* to the decision problem B, written $A \leqslant_p B$, if there is a function $f\colon \Sigma_A^\star \to \Sigma_B^\star$ (where Σ_A and Σ_B are the alphabets, $A \subseteq \Sigma_A^\star$, $B \subseteq \Sigma_B^\star$) computable in polynomial time such that, for every $x \in \Sigma^\star$, we have $x \in A$ iff $f(x) \in B$. The decision problem A is NP-*complete* if $A \in$NP and $B \leqslant_p A$ for every problem $B \in$NP.

If e is an event, prob(e) denotes its probability. A boolean random variable $P(n)$ which depends on a parameter n, holds *almost surely* or *in the limit* or *asymptotically* if $\lim_{n\to\infty} \text{prob}(P(n)) = 1$.

The mean of a random variable x will be denoted by $\mu(x)$ or by $E(x)$. The variance and the standard deviation will be denoted by $\nu(x)$ and by $\sigma(x)$, respectively. When there is no possibility of confusion, we use μ, ν and σ instead of $\mu(x)$, $\nu(x)$ and $\sigma(x)$ respectively.

A random variable x is (μ,σ)-*normal* if its density function and distribution function are respectively $f(x) = \frac{e^{(x-\mu)^2/(2\sigma^2)}}{\sqrt{2\pi}\sigma}$ and $F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{x-\mu}{\sigma}} e^{-t^2/2} dt$. A random variable x is *normal* if it is $(0,1)$-normal. Any (μ,σ)-normal random variable can be reduced by translation and change of scale to a normal variable $z = \frac{x-\mu}{\sigma}$. Let $\varphi(x)$ and $\Phi(x)$ be the density function and the distribution function respectively of the normal variable.

The following inequality will be useful and follows easily using the substitution $z = y^2/2$ in the distribution function: for $x > 0$, $1 - \Phi(x) \leqslant \varphi(x)/x$.

In particular we have

$$\text{for } x \geqslant 1,\ 1 - \Phi(x) \leqslant \varphi(x) \tag{1}$$

Consider a (possibly biased) coin such that heads turn up with a probability p and tails turn up with probability $q = 1 - p$. Suppose that the coin is tossed n times; the number x of heads is a random variable with probability $\text{bin}_{n,p}(x) = \binom{n}{x} p^x q^{(n-x)}$. This is the binomial distribution. The mean, variance, and standard deviation are respectively $\mu = pn$; $\nu = npq$; $\sigma = \sqrt{npq}$.

Using Moivre's Theorem, a consequence of the Central Limit Theorem, if n is large enough and x has a binomial distribution with parameters n and p, the distribution probability function $f_n(z)$ of the random variable $z = (x - pn)/\sqrt{npq}$ is approximately normal in the sense that $\forall \varepsilon > 0,\ \exists n_0,\ \forall n \geqslant n_0,\ \forall z:\ |f_n(z) - \varphi(z)| \leqslant \varepsilon$.

⋆ acm@dcc.fc.up.pt; All the authors are partially supported by CSI² (PTDC/EIA-CCO/099951/2008).The author is also partially funded by LIACC through Programa de Financiamento Plurianual, Fundao para a Cincia e Tecnologia (FCT)
⋆⋆ andreiasofia@dcc.fc.up.pt; The author is also supported by the grant SFRH/BD/33234/2007 of FCT
⋆⋆⋆ andresouto@dcc.fc.up.pt; The author is also supported by the grant SFRH/BD/28419/2006 of FCT

3 Finding the maximum area of mcr's is NP-hard

We study the complexity of the problem of finding the largest 1-mcr of a given colored square and prove that the corresponding decision problem is NP-complete.
The 1-mcr decision problem is the following.

1MMR, maximum area 1-mcr
INSTANCE: $\langle (Q, c), n, k\rangle$ where (Q, c) is a colored square with side n and $k \in \mathbb{Z}^+$.
QUESTION: Does Q contain an 1-mcr with area at least k?

Definition 1. *Given a graph $G = (V, E)$, a set $A \subseteq V$ is* independent *if $x, y \in A \Rightarrow (x, y) \notin E$.*

Definition 2. *A* biclique *of an undirected graph $G = (V, E)$ is a disjoint pair (A, B) of subsets of V such that $(a, b) \in E$ for every $a \in A$ and $b \in B$. The number of vertices of the biclique is $|A| + |B|$. If A and B are independent sets, the biclique will be called i-biclique (independent biclique). The number of edges of an i-biclique is $|A| \cdot |B|$.*

The concepts of colored square, binary relation, 1-mcr and biclique are related. In particular, a colored square (Q, c) with side n can be seen as a binary relation and an 1-mcr $A \times B$ belonging to a colored square Q, can be seen as a biclique of the corresponding graph.

Whenever we talk about bipartite graphs $G = (V_1 \cup V_2, E)$, it is assumed that V_1 and V_2 are independent and disjoint. Notice that, if (A, B) is a non empty biclique of a bipartite graph $G = (V_1 \cup V_2, E)$, then $A \subseteq V_1$ and $B \subseteq V_2$ or $A \subseteq V_2$ and $B \subseteq V_1$.

The following NP-complete decision problem related with 1MMR has been studied before [5].

MEiB, maximum edge independent biclique
INSTANCE: $\langle (V_1 \cup V_2, E), k\rangle$ where $(V_1 \cup V_2, E)$ is a bipartite graph and $k \in \mathbb{Z}^+$.
QUESTION: Does G contain an i-biclique (A, B) such that $|A| \cdot |B| \geqslant k$?

In [5], the proof that 3SAT $\leqslant_p$ MEiB uses the decision problem Clique/2 and characterizes the polynomial reductions 3SAT $\leqslant_p$ Clique/2 and Clique/2 $\leqslant_p$ MEiB;

The problems 1MMR and MEiB are quite similar; the graph associated with a colored square with side n is bipartite; compared with MEiB, the problem 1MMR has the additional restriction $|V_1| = |V_2|$. In order to prove that 1MMR is NP-complete, we now describe a polynomial reduction MEiB$\leqslant_p$ 1MMR:

$$\begin{gathered}\text{MEiB} \leqslant_p \text{1MMR}\\ \langle (V_1 \cup V_2, E), k\rangle \to \langle Q, n, c, k'\rangle\end{gathered}$$

Without loss of generality assume that $|V_1| \geqslant |V_2|$ and let $V_1 = \{x_1, x_2, \ldots, x_m\}$ and $V_2 = \{y_1, y_2, \ldots, y_p\}$, with $m \geqslant p$. The reduction is defined by: $n = m = |V_1|$; $Q = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$; $c(i, j) = 1$ iff $(i, j) \in E$ and $c(i, j) = 0$ for $p < j \leqslant n$; $k' = k$.

Then $(V_1 \cup V_2, E)$ has an i-biclique with k edges iff Q has an 1-mcr with area k' and that the transformation associated with the reduction can be implemented in polynomial time. Thus,

Theorem 1. *1MMR is NP-complete.*

4 The maximum areas when the probability is 1/2

4.1 Random colored squares

Let $N = \{1, 2, \ldots, n\}$ and consider a random colored square (Q, c) where $Q = N \times N$.

We say that $s(n)$ is an asymptotic upper bound of the maximum area s of an 1-mcr if

$$\lim_{n\to\infty} \text{prob}\{\text{there is an 1-mcr with area} \geqslant s(n)\} = 0$$

We now look for good upper bounds $s(n)$. Denote by $E(a, b)$ the expected number of 1-mcrs with height a and width b in a random colored square. We have $E(a, b) = 2^{-ab}\binom{n}{a}\binom{n}{b}$.

4.2 When both sides are large

Suppose that the values of a and b are such that $a, b > c \log n$, where c is a positive constant to be defined later. Hence $E(a, b) = 2^{-ab}\binom{n}{a}\binom{n}{b} \leqslant 2^{-ab} n^{a+b}$. Write $a = c \log(n) + \alpha$ and $b = c \log(n) + \beta$ where α and β are arbitrary positive constants. Then $E(a, b) \leqslant 2^{-ab} n^{a+b} = 2^{-ab+(a+b)\log n}$. Notice that the exponent of 2 can be written as

$$\begin{gathered}-ab + (a + b) \log n =\\ -(c \log n + \alpha)(c \log n + \beta) + (c \log n + \alpha + c \log n + \beta) \log n\\ = -c(c - 2) \log^2 n - (c - 1)(\alpha + \beta) \log n - \alpha\beta \leqslant -\gamma \log^2 n\end{gathered}$$

where γ is positive if $c > 2$; here and henceforth by "$c > 2$" we mean that $c \geqslant 2+\gamma$ for some positive constant γ, not depending on n. Thus $E(a, b) \leqslant 2^{-\gamma \log^2 n}$ and $\lim_{n\to\infty} E(a, b) = 0$. Thus,

Lemma 1. *If $c > 2$ and a, b are such that $a, b > c \log n$, then* $\lim_{n\to\infty} E(a, b) = 0$.

Areas instead of sides. Suppose now that, instead of the sides of the 1-mcr, a certain area $s(n)$ is given and we ask if the probability that there exists some 1-mcr with an area at least $s(n)$ is still vanishingly small; the answer is yes, as proved below.

The number of factorizations of a number m is usually denoted by $\tau(m)$ and it is known (see [4]) that $\forall \delta > 0\, \exists c > 0 :\ \tau(m) \leqslant c m^\delta$. As the maximum length of a side is n, the number of factorizations of s is at most n; this number will be denoted by $\tau_n(s)$.

Consider the case $a, b > c \log n$ with $c > 2$. Can we expect that at least an 1-mcr with a given area exists? Given that $\tau_n(s) \leqslant n$, the expected number $E(s)$ of such rectangles satisfies

$$\begin{aligned}E(s) &\leqslant \sum_{\{a,b:ab=s\wedge a,b>c\log n\}} E(a, b) \leqslant \tau_n(s) E(c \log n, c \log n)\\ &\leqslant \tau_n(s) 2^{-\gamma \log^2 n} = \tau_n(s) n^{-\gamma \log n} \leqslant n^{-\gamma \log n+1}\end{aligned}$$

where γ is a positive constant. Thus the answer to the question is no. Hence we have

Theorem 2. *For any constant $c > 2$ and area s, the following event has asymptotic probability 0: "there is at least one monochromatic combinatorial rectangle with area s and sides both larger than $c \log n$".*

4.3 When at least one side is small

We study the maximum areas for the case where at least one of the sides is small ($\leqslant 2 \log n$).

Fixed subset of rows Let us consider first a fixed subset with a rows. The probability that there are only 1's at the intersection of all these rows with some column is $p = 2^{-a}$; the number x of columns containing only 1's at those rows has a binomial probability function with

$$E(x) = \frac{n}{2^a}; \quad \nu(x) = n\frac{1 - 1/2^a}{2^a} \tag{2}$$

Using Moivre result, we normalize x by a change of variable $z = (x-\mu)/\sigma$, see page 2; the probability distribution of this normalized random variable is, in the limit, a normal probability distribution.

We want that, if $z \geqslant f(n)$, the probability prob$(z \geqslant f(n))$ is exponentially small. Based on inequality (1) a possibility is $f(n) = \log^2 n$.

Choosing the rows in every possible way By assumption $a \leqslant 2 \log n$, so that the number of ways of choosing the a rows satisfies $\binom{n}{a} \leqslant n^a \leqslant n^{2 \log n} = e^{(2 \log e) \ln^2 n}$. The probability of an union of events is at most the sum of their probabilities; thus the probability of having $z \geqslant \log^2 n$ is bounded by $g(n) = \frac{1}{\sqrt{2\pi}} e^{-\log^4 n/2 + (2 \log e) \ln^2 n}$ and $\lim_{n\to\infty} g(n) = 0$.

Areas instead of sides The probability $h(n)$ that there exists 1-mcrs with a given area s satisfies

$$\begin{aligned} h(n) &\leqslant \tau_n(s)g(n) \leqslant \tfrac{1}{\sqrt{2\pi}}\tau_n(s)e^{-\log^4 n/2+(c\log e)\ln^2 n} \\ &\leqslant \tfrac{1}{\sqrt{2\pi}}ne^{-\log^4 n/2+(c\log e)\ln^2 n} \\ &= \tfrac{1}{\sqrt{2\pi}}e^{-\log^4 n/2+(c\log e)\ln^2 n+\ln n} \end{aligned}$$

and again we have $\lim_{n\to\infty} h(n) = 0$.

4.4 The "1 line" versus "2 lines" competition

The asymptotic maximum area of an 1-mcr is $n/2 + o(n)$. Two shapes have, in the limit, that area: $1 \times n/2$ and $2 \times n/4$. How do the probabilities of each of these shapes fall out when the horizontal sides are slightly larger than the mean? Which of them (height 1 or 2), if any, dominates?

Let us review the main statistical parameters for the heights $a = 1$ and $a = 2$. Let x be the number of monochromatic columns; the area of an 1-mcr is ax. We have

	$a=1$	$a=2$	General case
$E(x)$	$n/2$	$n/4$	$n/2^a$
$E(s)$	$n/2$	$n/2$	$na/2^a$
$v(x)$	$n/4$	$3n/16$	$n(1-1/2^a)/2^a$
$\sigma(x)$	$\sqrt{n}/2$	$\sqrt{3n}/4$	$\sqrt{n(1-1/2^a)/2^a}$
$v(s)$	$n/4$	$3n/4$	$na^2(1-1/2^a)/2^a$
$\sigma(s)$	$\sqrt{n}/2$	$\sqrt{3n}/2$	$a\sqrt{n(1-1/2^a)/2^a}$

Heights larger than 2 can be ignored since the average value of the area is less than $n/2$.

Consider the standard deviations for $a = 2$ and $a = 1$; the former exceeds the second by a factor of $\sqrt{3}$. Thus, although the average value of the area is the same, we have

Theorem 3. *In a random colored square, the largest monochromatic combinatorial rectangles have area* $n/2 + o(n)$ *and shape* $2 \times (n/4 + o(n))$.

5 The maximum areas (arbitrary probability)

Consider again a random colored square (Q, c), but now suppose that each $c(i, j)$ is a random variable z that takes the value 1 with probability $p > 0$ and takes the value 0 with probability $1 - p > 0$. All these random variables being independent.

5.1 When both sides are large (arbitrary probability)

The main result of this sub-section is Theorem 4 which is a generalization of Lemma 1 (page 4).

As in Sec. 4.2 suppose that the values of a and b are such that $a, b > c\log n$ where c is a positive constant to be defined later. Denote $\log(1/p)$ by k. We have

$$E(a,b) = p^{ab}\binom{n}{a}\binom{n}{b} \leqslant p^{ab}n^{a+b} \tag{3}$$

$$= 2^{-\log(1/p)ab+(a+b)\log n} = 2^{-kab+(a+b)\log n} \tag{4}$$

Write $a = c\log(n)+\alpha$ and $b = c\log(n)+\beta$ where $\alpha, \beta \geqslant \epsilon > 0$. Thus, the exponent of 2 in (4) can be rewritten as

$$\begin{aligned} -kab + (a+b)\log n &= -k(c\log n+\alpha)(c\log n+\beta) + (2c\log n+\alpha+\beta)\log n \\ &= -kc^2\log^2 n - kc(\alpha+\beta)\log n - k\alpha\beta + 2c\log^2 n + (\alpha+\beta)\log n \\ &= -c(kc-2)\log^2 n - (kc-1)(\alpha+\beta)\log(n) - k\alpha\beta \end{aligned}$$

The exponent is asymptotically negative if $kc - 2$ is a positive constant if $c > 2/\log(1/p)$. Thus we have found that the constant c in (4.2) equals $2/\log(1/p)$. In the limit, the coefficient of $\log n$ and the independent term are irrelevant.

As we have seen in the case $p = 1/2$, the number of ways of choosing a lines and the number of shapes of the maximum area rectangles do not influence the asymptotic results, and we get the following generalization of Lemma 1.

Theorem 4. *If* a, b *are such that* $a, b > c\log n/\log(1/p)$ *with* $c > 2$*, then* $\lim_{n\to\infty} E(a, b) = 0$ *so, areas greater than* $4\log^2 n/\log^2(1/p)$ *do not exist in the limit.*

Small values of p correspond to small upper bounds for the areas while values of p near 1 correspond to larger upper bounds.

5.2 When at least one side is small (arbitrary probability)

Suppose that one of the sides, say a, is small in the sense that $a \leqslant \frac{2}{\log(1/p)}\log n$. As it happened in Sec 4.3, the maximum area is obtained for a fixed value of a, that is, a value independent of n (but dependent on p). The mean value, variance and standard deviation of the number of columns of an 1-mcr are $E_{p,a}(x) = np^a$; $v_{p,a}(x) = np^a(1-p^a)$ and $\sigma_{p,a}(x) = \sqrt{np^a(1-p^a)}$. The first two expressions are generalizations of equalities (2).

Let us find the value of a that maximizes the area. The mean value of the area is $E_{p,a}(s) = nap^a$, compare with particular case $p = 1/2$ (see the table in page 5). As a real function of a, $E_{p,a}(s)$ has a maximum that can be found by solving the equation $dE_{p,a}(s)/da = 0$; we find that the maximum area occurs for $a = 1/\ln(1/p)$.

Theorem 5. *The height of the maximum area of an 1-mcr is either* $a_1 = \lfloor 1/\ln(1/p)\rfloor$ *or* $a_2 = \lceil 1/\ln(1/p)\rceil$. *The maximum area is* $\max\{na_1p^{a_1}, na_2p^{a_2}\}$.

We have seen that, with an error less than 1, the value of a that maximizes the area is $a = 1/\ln(1/p)$. This maximum area is

$$n\frac{1}{\ln(1/p)}p^{1/\ln(1/p)} = n\frac{1}{\ln(1/p)}e^{\ln p/\ln(1/p)} = \frac{n}{e\ln(1/p)} \tag{5}$$

As expected, this value increases uniformly with p.

6 An application to communication complexity

The lower bound proved in Theorem 5 can be useful in Communication Complexity. Consider for instance the random function $f_p(x, y)$ that takes the value 1 with probability $p > 0$ and takes the value 0 with probability $1 - p > 0$. We will apply the results obtained in Sec. 5 to establish a lower bound for the deterministic communication complexity of $f_p(x, y)$.

It is well known (see for instance [3]) that a deterministic protocol for f_p induces a partition $Q = R_1 \cup R_2 \cup \ldots \cup R_m$ of the $n \times n$ square Q into monochromatic rectangles. If A_{max} is the maximum area of a monochromatic rectangle of Q, the number m of rectangles of any such partition satisfies $m \geqslant n^2/A_{max}$; therefore, we have the following inequality for the communication complexity $D(f_p)$ of $f_p(x, y)$: $D(f_p) \geqslant \lceil\log(n^2/A_{max})\rceil$.

The total area with color 1 is approximately pn^2; thus, using the equality (5) the number of 1-mcrs is at least $pn^2/(n/(e\ln(1/p))) = enp\ln(1/p)$; considering also 0-mcrs, we get the following lower bound for the number of monochromatic rectangles: $en[p\ln(1/p) + (1-p)\ln(1/(1-p))]$. Thus we have

Theorem 6. *The asymptotic deterministic communication complexity of a random function* $f_p(x, y)$ *satisfies*

$$D(f) \geqslant \lceil\log n + \log(p\ln(1/p) + (1-p)\ln(1/(1-p))) + \log e\rceil$$

For any boolean function $f(x, y)$, an upper bound of $D(f)$ is $\lceil\log(n)+1\rceil$ which corresponds to the following trivial protocol, see [3]:

Protocol P: 1) Alice sends x to Bob (at most $\lceil\log n\rceil$ bits are needed); 2) Bob computes $f(x, y)$ and sends this result (1 bit) to Alice.

Theorem 6 shows that, for any fixed p with $0 < p < 1$, this protocol is almost optimum for the random function in the sense that

$$\begin{aligned} D_P(f) - D(f) &\leqslant \lceil\log(n)\rceil + 1 - \lceil\log n + \log\left(p\ln(\tfrac{1}{p}) + (1-p)\ln(\tfrac{1}{1-p})\right) + \log e\rceil \\ &< 2 - \log e - \log(p\ln(1/p) + (1-p)\ln(1/(1-p))) \end{aligned}$$

This difference is tabulated below for $p = 0.1, 0.2, \ldots, 0.9$

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$D_P(f) - D(f)$	2.18	1.56	1.27	1.13	1.09	1.13	1.27	1.56	2.18

We conclude that, in the limit, no protocol for the random function is significantly better than the trivial protocol P.

References

1. Carsten Damm, Ki Hang Kim and Fred Roush, *On Covering and Rank Problems for Boolean Matrices and Their Applications*, in LNCS number 1627, page 123, 1999.
2. M. Garey and D. Johnson *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*, Series of Books in the Mathematical Sciences, W.H. Freeman and Company, San Francisco, 1979.
3. Eyal Kushilevitz and Noam Nisan, *Communication Complexity*, Cambridge University Press, New York, Springer-Verlag, 1996.
4. William LeVeque, Fundamentals of Number Theory, Dover Publications, 1996.
5. Ren Peeters, *The Maximum edge biclique problem is* NP*-complete*, Discrete Applied Mathematics **131**(3), pp 651-654, 2003.

On a relationship between non-deterministic communication complexity and instance complexity

Armando B. Matos (acm@ncc.up.pt),
Andreia C. Teixeira (andreiasofiat@hotmail.com), and
André C. Souto (andresouto@ncc.up.pt)*

DCC-FC & LIACC, Universidade do Porto
Rua do Campo Alegre 1021/1055, 4169-007 Porto, Portugal

Abstract. We study the relationship between non-deterministic communication complexity of uniform functions [8, 9, 4] and instance complexity [7] (see the definitions in 2.1 and 2.2 respectively). For that purpose, the witness of the non-deterministic communication protocol executed by Alice and Bob is interpreted by Alice as a program p that, for t sufficiently large, "corresponds exactly" (see Definition 4) to the instance complexity $\mathrm{ic}^t(\overline{y} : Y_1(\overline{x}))$; in the previous expression $\overline{x}$ and $\overline{y}$ are the parts of the input known by Alice and Bob respectively and $Y_1(\overline{x})$ is the set of all values of y such that $f(\overline{x}, y) = 1$.

The main results of this paper are $\max_{|x|=|y|=n}\{\mathrm{ic}^{t(n)}_{\mathrm{yes}}(y : Y_1(x))\} = N^1(f)$ and $\max_{|x|=|y|=n}\{\mathrm{ic}^{t(n)}(y : Y_1(x))\} = N(f)$ where $\mathrm{ic}^t_{\mathrm{yes}}(a : S)$ is a variant of instance complexity (see Definition 5) and the non-deterministic communication complexities $N^1(f)$ and $N(f)$ are defined in [4], Chap. 2. We also present in Sec. 5 a simple inequality relating individual communication complexity with instance complexity.

Keywords: communication complexity, instance complexity, non-determinism.

1 Introduction

Communication complexity and instance complexity seem at first to be totally unrelated concepts. In a typical setup for communication complexity the two parties, Alice and Bob, have unbounded computational power and one wants to find how many bits they need to exchange in order to compute the value of a given function of their inputs, $f : X \times Y \to \{0, 1\}$; on the other hand, the instance complexity $\mathrm{ic}^t(x : A)$ is the length of the shortest program that runs in time t, answers correctly the question "$x \in A$?" and does not "lie" about set A (the program may answer "$\perp$" meaning "don't know"). Communication complexity is about the communication cost while instance complexity is related with computational complexity.

In this paper we establish a relationship between these two concepts. Let $\overline{x}$ and $\overline{y}$ be the inputs of Alice and Bob respectively, $n = |\overline{x}| = |\overline{y}|$ and let $Y_1(\overline{x})$ be the set of all y such that $f(\overline{x}, y) = 1$. Theorem 2 states that, apart from a constant and for t sufficiently large, $\max_{|x|=|y|=n}\{\mathrm{ic}^t_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))\}$, where $\mathrm{ic}_{\mathrm{yes}}$ is a "one-sided" version of instance complexity (Definition 5), equals the non-deterministic communication complexity $N^1(f)$ of the uniform function f; similarly the maximum value of $\mathrm{ic}^t(\overline{y} : Y_1(\overline{x}))$ equals the non-deterministic communication complexity $N(f)$, see Theorem 3. The main ingredient for the proof of this result is a protocol in which Alice uses the non-deterministic word p as a program that eventually corresponds to $\mathrm{ic}^t(\overline{y} : Y_1(\overline{x}))$; Alice runs $p(y)$ for all $y \in Y$ for a maximum of $t(|y|)$ steps. We should notice two facts that may help the understanding the rest of this paper: (i) Neither Alice nor Bob alone (without communication and without the help of the oracle) can compute $\mathrm{ic}^t(\overline{y} : Y_1(\overline{x}))$; the reason is that Alice only knows $\overline{x}$ and Bob only knows $\overline{y}$. (ii) $\mathrm{ic}^{t(n)}(\overline{y} : Y_1(\overline{x}))$, like $N(f)$, can be much smaller than n.

We mention two previous works where the communication complexity has been analyzed in a non-standard way: the paper [2] on individual communication complexity in which Kolmogorov complexity is used as the main analysis tool and [5] where "distinguishers" are used to obtain bounds on communication complexity.

This paper is organized as follows. Next section contains some background on communication complexity and instance complexity. We study one-sided protocols in Sec. 3 and two-sided protocols in Sec. 4. These two sections contain the main results of this paper, namely Theorems 2 and 3. Sec. 5 contains some comments on the relationship between individual communication complexity and instance complexity. Finally in Sec. 6 some future lines of research are suggested.

* The author is supported by the grant SFRH/BD/28419/2006 from FCT

2 Preliminaries

The set of natural numbers (including 0) is denoted by $\mathbb{N}$. An alphabet Σ is an nonempty finite set whose members are called letters. The alphabet used in this paper is $\{0, 1\}$. A word is a sequence of 0 or more letters; words are denoted by x, y and w, possibly overlined. The length and the i-th letter of the word x are denoted by $|x|$ and x_i respectively.

2.1 Communication complexity

We define several forms of non-deterministic communication complexity, for more details see [4]. Let f: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a boolean function. Two players, Alice and Bob want to compute $f(\overline{x}, \overline{y})$; Alice only knows $\overline{x}$ while Bob only knows $\overline{y}$. A ("two-sided") non-deterministic protocol P for f has output $P(w, \overline{x}, \overline{y}) \in \{0, 1, \perp\}$ where $\perp$ means "don't know" and w is the guess; the protocol P satisfies the following conditions

$$[f(\overline{x}, \overline{y}) = 1] \Rightarrow [\exists w : P(w, \overline{x}, \overline{y}) = 1] \quad (1)$$
$$[f(\overline{x}, \overline{y}) = 0] \Rightarrow [\forall w : P(w, \overline{x}, \overline{y}) \neq 1] \quad (2)$$
$$[f(\overline{x}, \overline{y}) = 0] \Rightarrow [\exists w : P(w, \overline{x}, \overline{y}) = 0] \quad (3)$$
$$[f(\overline{x}, \overline{y}) = 1] \Rightarrow [\forall w : P(w, \overline{x}, \overline{y}) \neq 0] \quad (4)$$

For $z \in \{0, 1\}$ a "one-sided" protocol P^z has output either z or $\perp$ and satisfies

$$[f(\overline{x}, \overline{y}) = z] \Rightarrow [\exists w : P^z(w, \overline{x}, \overline{y}) = z] \quad (5)$$
$$[f(\overline{x}, \overline{y}) \neq z] \Rightarrow [\forall w : P^z(w, \overline{x}, \overline{y}) = \perp] \quad (6)$$

It is easy to build a non-deterministic protocol for f using the one-sided protocols P^0 and P^1.

We should emphasize that in any protocol, Alice and Bob must be convinced of the output of the protocol, in the sense that "false guesses" must be detected and rejected (output $\perp$); this requirement corresponds to the "$\forall \ldots$" predicates above. In other words, Alice and Bob do not trust the oracle. Otherwise the problem would be trivial.

Some of the variants of non-deterministic communication complexity are as follows.

Definition 1 (non-deterministic communication complexities). *Standard and individual (non-deterministic) communication complexities are denoted by N and $\mathcal{N}$ respectively.*

- Individual communication complexity of protocol P with output set $\{1, \perp\}$:
  $\mathcal{N}^1_P(f, \overline{x}, \overline{y}) = \min_w\{|c(w)| : P(w, \overline{x}, \overline{y}) = 1\}$ *where w is the guess and $c(w)$ ("conversation") is the sequence of bits exchanged between Alice and Bob when the guess is w. Notice that $\mathcal{N}^1_P(f, \overline{x}, \overline{y})$ is only defined if $f(\overline{x}, \overline{y}) = 1$. Notice also that the behavior of the protocol P for the other inputs (x, y) is irrelevant.*
- Individual communication complexity with output set $\{1, \perp\}$:
  $\mathcal{N}^1(f, \overline{x}, \overline{y}) = \min_P\{\mathcal{N}^1_P(f, \overline{x}, \overline{y})\}$ *where the protocols P considered for minimization are protocols with output set $\{1, \perp\}$ for the function f.*
- Communication complexity of protocol P with output set $\{1, \perp\}$:
  $N^1_P(f) = \max_{\overline{x},\overline{y}}\{\mathcal{N}^1_P(f, \overline{x}, \overline{y})\}$:
- Communication complexity of function f with output set $\{1, \perp\}$:
  $N^1(f) = \min_P\{N^1_P(f)\}$.

The complexities $\mathcal{N}^0_P(f, \overline{x}, \overline{y})$, $N^0_P(f)$, and $N^0(f)$, are defined in a similar way.
Define also $\mathcal{N}_P(f, \overline{x}, \overline{y}) = \mathcal{N}^0_P(f, \overline{x}, \overline{y})$ if $f(\overline{x}, \overline{y}) = 0$ and $\mathcal{N}_P(f, \overline{x}, \overline{y}) = \mathcal{N}^1_P(f, \overline{x}, \overline{y})$ if $f(\overline{x}, \overline{y}) = 1$; $N_P(f) = \log(2^{N^0_P(f)} + 2^{N^1_P(f)})$; $N(f) = \min_P\{N_P(f)\}$.
A witness *is a guess that causes the protocol to output a value different from $\perp$.* □

In the literature it is possible to find the following definition of the individual (non-deterministic) communication complexity associated with protocol P, see for instance [1]: $\mathcal{N}^1_P(f, \overline{x}, \overline{y}) = \min_w\{|w| + |c(w)| : P(w, \overline{x}, \overline{y}) = 1\}$; comparing with Definition 1, we see that the corresponding values differ by at most a factor of 2.

The definition $N_P(f) = \log(2^{N^0_P(f)} + 2^{N^0_P(f)})$ is from [4]; we have

$$\max\{N^0_P(f), N^1_P(f)\} < N_P(f) \leq \max\{N^0_P(f), N^1_P(f)\} + 1$$

therefore $N_P(f) \approx \max\{N^0_P(f), N^1_P(f)\}$.

The following result from [4] shows that for every function there is a simple optimal non-deterministic protocol.

Theorem 1. *For every boolean function f there is an optimal one-sided non-deterministic protocol P for f, that is, a protocol P such that $N^1_P(f) = N^1(f)$, with the following form where the witness w, $1 \le w \le m$, is the index of the first rectangle $R_w = A \times B$ containing $(\overline{x}, \overline{y})$ in the first minimum 1-cover: (1) Alice guesses w and checks if $\overline{x} \in A$. (2) Alice sends w to Bob. (3) Bob checks if $\overline{y} \in B$.*

Define the sets: $X_0(\overline{y}) = \{x : f(x,\overline{y}) = 0\}$, $X_1(\overline{y}) = \{x : f(x,\overline{y}) = 1\}$, $Y_0(\overline{x}) = \{y : f(\overline{x},y) = 0\}$ and $Y_1(\overline{x}) = \{y : f(\overline{x},y) = 1\}$. Notice that Alice knows $Y_0(\overline{x})$ and $Y_1(\overline{x})$ while Bob knows $X_0(\overline{y})$ and $X_1(\overline{y})$. The set Y_1 is often mentioned in this paper.

Theorems 2 and 3 apply only to "uniform" functions.

Definition 2. *A function is* uniform *if it is computed by a fixed (independent of the length of the input) algorithm.* □

Every function that can be described by an algorithm is uniform; for instance equality, conjunction and parity are uniform functions. An example of a function which with almost certainty is not uniform is the random function defined as $f(x,y) = 0$ or $f(x,y) = 1$ with probability $1/2$.

2.2 Instance complexity

We define several forms of instance complexity; for a more complete presentation see [7]. It is assumed that programs always terminate, and output either 0, 1 or $\perp$ ("don't know").

Definition 3. *A program p is* consistent *with a set A if $x \in A$ whenever $p(x) = 1$ and $x \notin A$ whenever $p(x) = 0$.* □

Definition 4 (instance complexity). *Let $t : \mathbb{N} \to \mathbb{N}$ be a function, A a set and x an element. Consider the following conditions: (C1) for all y, $p(y)$ runs in time not exceeding $t(|y|)$, (C2) for all y, $p(y)$ outputs 0, 1 or $\perp$, (C3) p is consistent with A and (C4) $p(x) \neq \perp$. The* t-bounded instance complexity *of x relative to the set A is*

$$\mathrm{ic}^t(x : A) = \min\{|p| : p \text{ satisfies C1, C2, C3, and C4}\}$$

A program p corresponds *to $\mathrm{ic}^t(x : A)$ if it satisfies conditions C1, C2, C3, and C4; if moreover $|p| = \mathrm{ic}^t(x : A)$ we say that p* corresponds exactly *to $\mathrm{ic}^t(x : A)$.* □

Relaxing the condition "$p(x) \neq \perp$" we get two weaker forms of instance complexity:

Definition 5 (inside instance complexity). *Consider the following conditions: (C1) for all y, $p(y)$ runs in time not exceeding $t(|y|)$, (C2) for all y, $p(y)$ outputs either 1 or $\perp$, (C3) p is consistent with A and (C4) $x \in A \Rightarrow p(x) = 1$. The* t-bounded inside instance complexity *of x relative to the set A is*

$$\mathrm{ic}^t_{\mathrm{yes}}(x : A) = \min\{|p| : p \text{ satisfies C1, C2, C3, and C4}\}$$

A program p corresponds *to $\mathrm{ic}^t_{\mathrm{yes}}(x : A)$ if it satisfies conditions C1, C2, C3, and C4; if moreover $|p| = \mathrm{ic}^t_{\mathrm{yes}}(x : A)$ we say that p* corresponds exactly *to $\mathrm{ic}^t_{\mathrm{yes}}(x : A)$.* □

Definition 6 (outside instance complexity). *Consider the following conditions: (C1) for all y, $p(y)$ runs in time not exceeding $t(|y|)$, (C2) for all y, $p(y)$ outputs either 0 or $\perp$, (C3) p is consistent with A and (C4) $x \notin A \Rightarrow p(x) = 0$. The* t-bounded outside instance complexity *of x relative to the set A is*

$$\mathrm{ic}^t_{\mathrm{no}}(x : A) = \min\{|p| : p \text{ satisfies C1, C2, C3, and C4}\}$$

A program p corresponds *to $\mathrm{ic}^t_{\mathrm{no}}(x : A)$ if it satisfies conditions C1, C2, C3, and C4; if moreover $|p| = \mathrm{ic}^t_{\mathrm{no}}(x : A)$ we say that p* corresponds exactly *to $\mathrm{ic}^t_{\mathrm{no}}(x : A)$.* □

Notice that if $x \notin A$ then $\mathrm{ic}^t_{\mathrm{yes}}(x : A)$ is a constant (independent of x), because the program $p(x) \equiv \perp$ has fixed length and is consistent with every set; similarly if $x \in A$ then $\mathrm{ic}^t_{\mathrm{no}}(x : A)$ is a constant. Notice also that for every element x, set A and function t we have $\mathrm{ic}^t_{\mathrm{yes}}(x : A) \le \mathrm{ic}^t(x : A)$ and $\mathrm{ic}^t_{\mathrm{no}}(x : A) \le \mathrm{ic}^t(x : A)$. On the other hand, from a program p corresponding to $\mathrm{ic}^{t_1}_{\mathrm{yes}}(x : A)$ and a program p' corresponding to $\mathrm{ic}^{t_2}_{\mathrm{no}}(x : A)$ we can define a program r as follows: $r(x) = 1$ if $p(x) = 1$, $r(x) = 0$ if $p'(x) = 0$ and $r(x) = \perp$ otherwise, concluding that

$$\mathrm{ic}^{f(t_1,t_2)}(x : A) \le \mathrm{ic}^{t_1}_{\mathrm{yes}}(x : A) + \mathrm{ic}^{t_2}_{\mathrm{no}}(x : A) + O(\log(\min\{\mathrm{ic}^{t_1}_{\mathrm{yes}}(x : A), \mathrm{ic}^{t_2}_{\mathrm{no}}(x : A)\}))$$

where the function f represents the time overhead needed for the simulation of $p(x)$ for t_1 steps followed by simulation of $p'(x)$ for t_2 steps; the logarithmic term comes from the need to delimit p from p' in the concatenation pp'.

Notation. To emphasize that t is a function of n, we write $\mathrm{ic}^{t(n)}(y : A(x))$, $\mathrm{ic}^{t(n)}_{\mathrm{yes}}(y : A(x))$ and $\mathrm{ic}^{t(n)}_{\mathrm{no}}(y : A(x))$.

3 One-sided protocols

As an illustration we first consider in sub-section 3.1 a somewhat simplified analysis of the function $\overline{x} \neq \overline{y}$ (also called "NEQ"), and show how to use programs corresponding to instance complexity as guesses of (optimal) non-deterministic protocols. This usage is later generalized to any uniform function in sub-section 3.2.

3.1 Inequality: an optimal "$\mathrm{ic}_{\mathrm{yes}}$-protocol"

Consider the predicate NEQ and suppose that $\overline{x} \neq \overline{y}$; then for some i, $1 \le i \le n$, we have $\overline{x}_i \neq \overline{y}_i$. A possible program p_i corresponding to $\mathrm{ic}^t_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))$ is $p_i(y) = 1$ if $y_i \neq \overline{x}_i$, $p_i(y) = \perp$ if $y_i = \overline{x}_i$. The reader may compute the set $Y'_1 = \{y : p_i(y) = 1\}$ and show that $Y'_1 \subset Y_1(\overline{x})$. If $p(\overline{y}) = 1$ and if $|p|$ is minimum, this program corresponds exactly to $\mathrm{ic}^t_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))$.

Consider now the following protocol P_t for NEQ where t is a time bound sufficiently large (see more details in sub-section 3.2). Alice receives a word p as a guess; p may eventually be the program p_i above. Then she runs $p(y)$ for every $y \in Y$ until the program halts or until $t(|y|)$ steps have elapsed. If $p(y)$ does not halt in time $t(|y|)$, the word p is not a valid witness and the protocol halts. Otherwise Alice defines the set $Y'_1 = \{y : p(y) = 1\}$. If $Y'_1 \subseteq Y_1(\overline{x})$, i.e., if p is consistent with $Y_1(\overline{x})$, she sends p to Bob, otherwise outputs $\perp$ and halts. Bob tests if $p(\overline{y}) = 1$; if yes, outputs 1, otherwise outputs $\perp$.

Correctness conditions:

(1) `If` $\overline{x} \neq \overline{y}$ `, there is a witness` p `that corresponds to` $\mathrm{ic}^t_{\mathrm{yes}}(\overline{y}, Y_1(\overline{x}))$. We have $\overline{x}_i \neq \overline{y}_i$ for some i, $0 \le i \le n$. Then, if p happens to be the program p_i above, the protocol P outputs 1 so p corresponds to $\mathrm{ic}^t_{\mathrm{yes}}(\overline{y}, Y_1(\overline{x}))$, that is, we must have Y'_1 consistent with $Y_1(\overline{x})$ (verified by Alice) and $p(\overline{y}) = 1$ (verified by Bob).

(2) `If a guess is wrong, the output is` $\perp$. If the guess is wrong, then either some $p(y)$ runs for more than $t(|y|)$ steps or p is not consistent with $Y_1(\overline{x})$ or $p(\overline{y}) = \perp$; if $t(n)$ is sufficiently large, the output is $\perp$ in all these cases.

(3) `If` $\overline{x} = \overline{y}$`, no guess` p `can cause output 1.` This follows directly from the definition of the protocol.

Complexity: The length of p_i need not to exceed $\log n + O(1)$ and $\max_{0 \le i \le n}\{|p_i|\}$ is $\log n + O(1)$. Thus the complexity of the protocol P is $\log(n) + O(1)$. But the non-deterministic communication complexity of NEQ is also $\log n + O(1)$ (see [4]), thus the protocol is optimal.

3.2 "$\mathrm{ic}_{\mathrm{yes}}$-protocols" are optimal

Consider now the one-sided protocol of Figure 1. In the general case, the function f, which is known by Alice and Bob, is arbitrarily complex; therefore the description of f can not be included into an "instance complexity program" p unless $\lim_{n\to\infty} |p| = \infty$ (see also the comments after the proof of Theorem 2). However, a much simpler situation arises if we consider only uniform functions.

Theorem 2 ($\mathrm{ic}_{\mathrm{yes}}$-protocols are optimal). *Let f be an uniform function. There is a computable function $t(n)$ such that*

$$N^1(f) = \max_{|x|=|y|=n} \{\mathrm{ic}^{t(n)}_{\mathrm{yes}}(y : Y_1(x))\} + O(1) \tag{7}$$

Proof. Let p be the non-deterministic word given to Alice by the oracle; the protocol $P_{t(n)}$ is described in Figure 1 where $t(n)$ is an appropriate time bound (see below). Notice that the protocol specifies that Alice should interpret p as a program and execute p for a maximum time $t(n)$.

The program p, being an arbitrary word, may behave in many different ways; in particular, if $f(\overline{x},\overline{y}) = 1$, the behavior described in Figure 2 will cause $P_{t(n)}$ to output 1.

If i is chosen so that $(\overline{x},\overline{y}) \in R_i$ (if $f(\overline{x},\overline{y}) = 1$ there is at least one such i, otherwise there is none) then p is consistent with $Y_1(\overline{x})$ and $p(\overline{y}) = 1$. Then $|p| \ge \mathrm{ic}^{t(n)}_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))$ for $t(n)$ sufficiently large. Moreover, if p is not "correct", that fact can be detected by Alice or by Bob; thus, conditions (5) and (6) (see page 2) are verified.

As f is assumed to be uniform, the length of a program which is accepted as a witness, needs not to exceed $\log m + O(1)$.

How much time $t(n)$ must Alice run $p(y)$ (for each y) so that, there is at least a witness for every pair $(\overline{x},\overline{y})$ with $f(\overline{x},\overline{y}) = 1$? As f is uniform, it is possible to obtain an upper bound $t(n)$ in a constructive way by detailing and analyzing the algorithm that the witness p should implement, see Figure 2. Therefore we may suppose that $t(n)$ is a well defined function.

Alice:

Receive program $p(y)$ (as a possible witness)
Test if, for every $y \in Y$,
  $p(y)$ halts in time not exceeding $t(n)$ with output 1 or $\perp$
If not, output $\perp$ and halt
Compute the set $B = \{y : p(y) = 1\}$
Find the set of smallest 1-covers
Select the first (in lexicographic order) such
cover $\langle R_1, R_2, \ldots R_m \rangle$
Select a rectangle $R_i = A \times B$ from that cover
  where $B \subseteq Y$ is the set computed above
  As the cover is minimum, there can be at most one such
  rectangle. If there is none, output $\perp$ and halt
Comment. At this point we know that p is consistent with $Y_1(\overline{x})$
Test if $\overline{x} \in A$
If not, output $\perp$ and halt
Send p to Bob

Bob:

Verify if $p(\overline{y}) = 1$
If yes, output 1 and halt
Output $\perp$ and halt

Fig. 1. A family of one-sided non-deterministic protocols $P_{t(n)}$. The guess is based on a program p that corresponds to $\mathrm{ic}^t_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))$.

Program p, input y:

From $d(f)$ and i:
  Find the set S_1 of smallest 1-covers
  Select the first (in lexicographic order) cover $\langle R_1, R_2, \ldots R_m \rangle \in S_1$
  Select rectangle $R_i = A \times B$ in that cover
With input y, output
  $p(y) = 1$ if $y \in B$
  $p(y) = \perp$ otherwise

Fig. 2. A possible behavior of the program p which may cause the protocol $P_{t(n)}$ (see Figure 1) to output 1. A string p with this behavior can be specified in length $|d(f)| + \log m$. The existence of this program, which has length $\log m + O(1)$ where m is the size of the minimum covers, justifies the step between equation (9) and inequality (10).

Suppose now that f is uniform, and that $f(\overline{x}, \overline{y}) = 1$. If the protocol accepts $(\overline{x}, \overline{y})$ with guess p, we have $|p| \leq \log m + O(1)$ and $\max_{|\overline{x}|=|\overline{y}|=n}\{|p|\} \leq \log m + O(1)$. Thus

$$N^1(f) = \log C^1(f) + O(1) \tag{8}$$
$$= \log m + O(1) \tag{9}$$
$$\geq \max_{|\overline{x}|=|\overline{y}|=n} \{|p|\} + O(1) \tag{10}$$
$$\geq \max_{|\overline{x}|=|\overline{y}|=n} \{\mathrm{ic}^{t(n)}_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))\} + O(1) \tag{11}$$

On the other hand, *there exists* a non-deterministic protocol with complexity $\max_{|\overline{x}|=|\overline{y}|=n}\{\mathrm{ic}^{t(n)}_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))\} + O(1)$; this is the protocol of Figure 3 when $t(n)$ is sufficiently large. Notice that program p can be *any* program running in time $t(n)$ which is consistent with $Y_1(\overline{x})$ and such that $p(\overline{y}) = 1$ (and, if $f(\overline{x}, \overline{y}) = 1$, there is at least one such program, as we have seen above); thus it can be the shortest such program, $|p| = \mathrm{ic}^{t(n)}_{\mathrm{yes}}(\overline{y}, Y_1(\overline{x}))$. Taking the maximum over all $\overline{x}$ and $\overline{y}$ with $|\overline{x}| = |\overline{y}| = n$ (see Definition 1) we get $N^1(f) \leq \max_{|\overline{x}|=|\overline{y}|=n}\{\mathrm{ic}^{t(n)}_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))\} + O(1)$ because $N^1(f)$ is the smallest complexity among all the protocols for f. Combining this result with inequation (11) we get $N^1(f) = \max_{|\overline{x}|=|\overline{y}|=n}\{\mathrm{ic}^{t(n)}_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))\} + O(1)$. □

A note on the uniformity condition

At first it may be not obvious why the validity of equality (7) of Theorem 2 depends on the uniformity of f. Let us argue that (7) may be false for non uniform functions, using the Kolmogorov complexity as a tool. Denote by $C(x)$ the (plain) Kolmogorov complexity of x which is defined as $C(x) = \min\{|p| : U(p) = x\}$ where U is some fixed universal Turing machine, see [6].

Consider a monochromatic cover of a non uniform function such that (i) the number m of rectangles in the cover is very small and (ii) the horizontal side B of the first rectangle in the cover has a Kolmogorov random length, $C(|B|) \approx n$. The length B can be obtained from p, thus $C(|B|) \leq C(p) + O(1)$ which implies $C(p) \geq n + O(1) \gg \log m$; thus the step (9) → (10) in the proof is not valid.

Alice:

Receive program $p(y)$ (as a possible witness)
Test if, for every y, $p(y)$ halts in at most $t(n)$ steps
If not, output $\perp$ and halt
Test if $\{y : p(y) = 1\} \subseteq Y_1(\overline{x})$ (p is consistent with $Y_1(\overline{x})$)
If not, output $\perp$ and halt
Send p to Bob

Bob:

Compute $r = p(\overline{y})$ and test if $r = 1$
If not, output $\perp$ and halt
Output 1

Alice:

Output the message received

Fig. 3. A family of one-sided non-deterministic protocols $P'_{t(n)}$. The guess may be any program p that corresponds to $\mathrm{ic}^t_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x}))$, that is p must satisfy only $\{y : p(y) = 1\} \subseteq Y_1(\overline{x})$ and $p(\overline{y}) = 1$.

4 Two-sided protocols

Now we consider the two-sided protocols for non-deterministic communication complexity. If $t(n)$ is sufficiently large, there are optimum protocols whose guesses correspond exactly to $\mathrm{ic}^t(\overline{y} : Y_1(\overline{x}))$.

Theorem 3. *Let f be an uniform function. There is a computable function $t(n)$ such that*

$$N(f) = \max_{|\overline{x}|=|\overline{y}|=n} \{\mathrm{ic}^{t(n)}(y : Y_1(x))\} + O(1)$$

See the Appendix for comments on the proof of this result.

5 About individual communication complexity

The one sided individual communication complexity satisfies

$$\mathcal{N}^1(f, \overline{x}, \overline{y}) \geq \mathrm{ic}^t_{\mathrm{yes}}(\overline{y} : Y_1(\overline{x})) + O(1)$$

The complexity $\mathcal{N}^1(f, \overline{x}, \overline{y})$ is obtained from a minimization over all protocols which must of course "work correctly" for every pair (x, y) and not only for $(\overline{x}, \overline{y})$ while no such restriction exists in the definition of instance complexity. The individual communication complexity may in a few rare cases (if i has a very short description), be much smaller than $\log m$.

Finally we present a result relating the individual non-deterministic communication complexity with the instance complexity.

Theorem 4. *(Individual upper bound) For every function f and values x and y there is a $t_0 \in \mathbb{N}$ such that the individual non-deterministic communication complexity $\mathcal{N}(f, x, y)$ satisfies*

$$\forall t \geq t_0 : \mathcal{N}(f, x, y) = \mathrm{ic}^t(y : Y_1(x)) + O(1) \leq N(f) + O(1)$$

6 Conclusions and future work

We have established that, for uniform functions f and for $t(n)$ sufficiently large, the maximum value of $\mathrm{ic}^{t(n)}(\overline{y} : Y_1(\overline{x}))$ (where $Y_1(\overline{x}) = \{y : f(\overline{x}, y) = 1\}$) equals the non-deterministic communication complexity $N(f)$. The work done here can be continued along several lines of research and, in particular, it would be interesting to study more deeply the relationship between *individual* communication complexity and instance complexity and to search for the existence and properties of protocols that, besides being optimal (in the worst case), minimize the communication for all pairs $(\overline{x}, \overline{y})$. It would also be interesting to relate the time bound of instance complexity with the complexity class of the function f.

In more general terms we think that it is important to study in depth the relationship between measures of communication cost and measures of computational complexity.

References

1. Sanjeev Arora, Boaz Barak, *Computational Complexity: A Modern Approach*, Princeton University, 2006, url: `http://www.cs.princeton.edu/theory/complexity/communicatechap.pdf`
2. Harry Buhrman, Hartmut Klauck, Nikolai Vereshchagin and Paul Vitány, *Individual communication complexity*, STACS 2004: 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004.
3. Ilan Kremer, Noam Nisan, Dana Ron, *On randomized one-round communication complexity*, Proc. of 27th STOC, pp 596-605, 1995, url: `citeseer.ist.psu.edu/kremer95randomized.html`
4. Eyal Kushilevitz, Noam Nisan, *Communication Complexity*, Cambridge University Press, New York, Springer-Verlag, 1996.
5. Sophie Laplante, John Rogers, *Indistinguishability*, Technical report TR-96-26, 1996, url: `citeseer.ist.psu.edu/laplante98indistinguishability.html`
6. Ming Li e Paul Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag Graduate Texts In Computer Science Series, second edition, 1997.
7. Pekka Orponen, Ker-I Ko, Uwe Schöning, Osamu Watanabe, *Instance Complexity*, Journal of the ACM, 41:1, pp 96-121, 1994, url: `citeseer.ist.psu.edu/orponen94instance.html`.
8. Andrew Chi-Chih Yao, *Some complexity questions related to distributive computing*, Proceedings of the 11th Annual ACM Symposium on Theory of Computing, Atlanta, pp 209-213, 1979.
9. Andrew Chi-Chih Yao, *The entropic limitations on VLSI computations*, Proceedings of the 13th Annual ACM Symposium on Theory of Computing, Milwaukee, pp 308-311, 1981.

Appendix A. About the proof of Theorem 3

The proof of Theorem 3 is similar to the proof of Theorem 2; we make only a few observations. The reader should compare Figures 1 and 2 with Figures 4 and 5 respectively. The main difference in the proof is that we have now to consider a minimum cover of 0-rectangles and a minimum cover of 1-rectangles. Denote by $m = C^0(f)$ and $m' = C^1(f)$ the size of those covers; as the function f is assumed to be uniform, the witness (program) p has a description with length $\log(m + m') + O(1)$. It is not difficult to verify the correctness of conditions (1) to (4), see page 2.

Alice:

Receive program $p(y)$ (as a possible witness)
Test if, for every $y \in Y$,
  $p(y)$ halts in time not exceeding $t(n)$ with output 0, 1 or $\perp$
Compute the set $B = \{y : p(y) \neq \perp\}$
Test if B is monochromatic and not empty
Find the set S_0 of smallest 0-covers
  and the set S_1 of smallest 1-covers
Select the first (in lexicographic order) sequence
  $s = \langle R_1, \ldots R_m, R_{m+1}, \ldots R_{m+m'} \rangle$
  where $\langle R_1, \ldots R_m \rangle \in S_0$ and $\langle R_{m+1}, \ldots R_{m+m'} \rangle \in S_1$
Select a rectangle $R_i = A \times B$ from s
  Comment. There is at most one such rectangle
Test if $\overline{x} \in A$
Send p to Bob

Bob:

Compute $r = p(\overline{y})$
Output r

Fig. 4. A family of two-sided non-deterministic protocols $P_{t(n)}$. The guess is based on a program p that corresponds to $\mathrm{ic}^t(\overline{y} : Y_1(\overline{x}))$. Compare with Figure 1. For simplicity we assume that whenever a test fails, the protocol outputs $\perp$ and halts.

Program p, input y:

From $d(f)$ and i:
  Find the set S_0 of smallest 0-covers and
    the set S_1 of smallest 1-covers
  Select the first (in lexicographic order) sequence
    $s = \langle R_1, \ldots R_m, R_{m+1}, \ldots R_{m+m'} \rangle$
    where $\langle R_1, \ldots R_m \rangle \in S_0$ and $\langle R_{m+1}, \ldots R_{m+m'} \rangle \in S_1$
  Select the ith rectangle $R_i = A \times B$ from s
With input y, output:
  $p(y) = z$ if $y \in B$ and rectangle $A \times B$ has color $z \in \{0, 1\}$
  $p(y) = \perp$ otherwise

Fig. 5. A possible behavior of the program p which may cause the protocol $P_{t(n)}$ of Figure 4 to output a value different from $\perp$. A string p with this behavior can be specified in length $|d(f)| + \log(m + m')$.

Entropy Measures vs. Algorithmic Information

Andreia Teixeira
Universidade do Porto and
Instituto de Telecomunicações
andreiasofia@ncc.up.pt

André Souto
Universidade do Porto and
Instituto de Telecomunicações
andresouto@dcc.fc.up.pt

Armando Matos
Universidade do Porto and
LIACC
acm@dcc.fc.up.pt

Luís Antunes
Universidade do Porto and
Instituto de Telecomunicações
lfa@dcc.fc.up.pt

***Abstract*—Algorithmic entropy and Shannon entropy are two conceptually different information measures, as the former is based on size of programs and the later in probability distributions. However, it is known that, for any recursive probability distribution, the expected value of algorithmic entropy equals its Shannon entropy, up to a constant that depends only on the distribution. We study if a similar relationship holds for Rényi and Tsallis entropies of order α, showing that it only holds for Rényi and Tsallis entropies of order 1 (i.e., for Shannon entropy). Regarding a time bounded analogue relationship, we show that, for distributions such that the cumulative probability distribution is computable in time $t(n)$, the expected value of time-bounded algorithmic entropy (where the alloted time is $nt(n)\log(nt(n))$) is in the same range as the unbounded version. So, for these distributions, Shannon entropy captures the notion of computationally accessible information. We prove that, for universal time-bounded distribution $\mathbf{m}^t(x)$, Tsallis and Rényi entropies converge if and only if α is greater than 1.**

I. Introduction

Algorithmic entropy or Kolmogorov complexity, $K(x)$, measures rigorously the amount of information contained in an individual object (usually a string) x, by the size of the smallest program that generates it. It naturally defines a probability distribution over Σ^* (the set of all finite binary strings), assigning a probability of $2^{-K(x)}$ for any string x. This probability distribution is called universal probability distribution and it is denoted by $\mathbf{m}$.

Shannon entropy of a random variable X, $H(X)$, is a measure of its average uncertainty. It is the smallest number of bits required, on average, to describe x, the output of the random variable X.

Algorithmic entropy and Shannon entropy are conceptually different, as the former is based on the length of programs and the later in probability distributions. However, it is known that, for any recursive probability distribution (i.e. distributions that are computable by a Turing machine), the expected value of the algorithmic entropy equals the Shannon entropy, up to a constant term depending only on the distribution (see [7]).

Several information measures or entropies have been introduced since Shannon seminal paper [9]. We are interested in two different generalizations of Shannon entropy:

- Rényi entropy, an additive measure based on a specific form of the Kolmogorov-Naguno mean of the elementary information gain: instead of using the arithmetic mean, Rényi used the Kolmogorov-Naguno mean associated with the function $f(x) = c_1 b^{(1-\alpha)x} + c_2$ where c_1 and c_2 are constants, b is a real greater than 1 and α is a non-negative parameter;
- Tsallis entropy, a non additive measure, often called a non-extensive measure, in which the probabilities are scaled by a positive power α, that may either reinforce the large (if $\alpha > 1$) or the small (if $\alpha < 1$) probabilities.

Let $R_\alpha(P)$ and $T_\alpha(P)$ denote, respectively, the Rényi and Tsallis entropies associated with the probability distribution P. Both are continuous functions of the parameter α and both are (quite different) generalizations of the Shannon entropy, in the sense that $R_1(P) = T_1(P) = H(P)$ (see [1]). It is well known that for any recursive probability distribution P over Σ^*, the average value of $K(x)$ and the Shannon entropy $H(P)$ are close, in the sense that

$$0 \leq \sum_x P(x)K(x) - H(P) \leq K(P) \qquad \text{(I.1)}$$

where $K(P)$ is the length of the shortest program that describes the distribution P. We study if this property also holds for Rényi and Tsallis entropies. The answer is no. If we replace H by R or T, the inequalities I.1 are no longer true (unless $\alpha = 1$). We also study the convergence of Tsallis and Rényi entropies of the universal time-bounded distribution $\mathbf{m}^t(x) = 2^{-K^t(x)}$, proving that both entropies converge if and only if $\alpha > 1$. Finally, we analyze the validity of the relationship I.1, replacing algorithmic entropy by its time-bounded version, proving that it holds for distributions such that the cumulative probability distribution is computable in an alloted time. So, for these distributions, Shannon entropy equals the expected value of the time-bounded algorithmic entropy.

The rest of this paper is organized as follows: in the next Section, we present the notions and results that will be used. In Section 3, we study if the inequalities I.1 can be generalized for Rényi and Tsallis entropies and we also establish a similar relationship for the time-bounded algorithmic entropy. In Section 4, we analyze the entropies of the universal time-bounded distribution.

II. Preliminaries

$\Sigma^* = \{0,1\}^*$ is the set of all finite binary strings. The empty string is denoted by ϵ. Σ^n is the set of strings of length n and $|.|$ denotes the length of a string. Strings are lexicographically ordered. The logarithm of x in base 2 is denoted by $\log(x)$. The real interval between a and b, including a and excluding b is represented by $[a,b)$. A sequence of real numbers r_n is denoted by $(r_n)_{n\in\mathbb{N}}$.

A. Algorithmic entropy

Algorithmic entropy also known as Kolmogorov complexity was introduced independently, in the 60's by Solomonoff [10], Kolmogorov [5], and Chaitin [2]. Only essential definitions and basic results are given here, for further details see [7]. The model of computation used is the prefix-free Turing machine, i.e., Turing machines with a prefix free domain. A set of strings A is prefix-free if no string in A is prefix of another string of A. Kraft's inequality guarantees that for any prefix-free set A, $\sum_{x\in A} 2^{-|x|} \leq 1$.

Definition 1 (Algorithmic entropy)**.** *Let U be a fixed prefix-free universal Turing machine. For any two strings $x, y \in \Sigma^*$, the algorithmic entropy or Kolmogorov complexity of x given y is $K(x|y) = \min_p\{|p| : U(p,y) = x\}$, where $U(p,y)$ is the output of the program p with auxiliary input y when it is run in the machine U. For any time constructible t, the t-time-bounded algorithmic entropy of x given y is, $K^t(x|y) = \min_p\{|p| : U(p,y) = x$ in at most $t(|x|)$ steps$\}$.*

The default value for y, the auxiliary input is the empty string ϵ; for simplicity $K(x)$ and $K^t(x)$ denote $K(x|\epsilon)$ and $K^t(x|\epsilon)$, respectively. The choice of the universal Turing machine affects the running time of a program by, at most, a logarithmic factor and the program length by, at most, a constant number of extra bits.

Definition 2. *Let c be a non-negative integer. We say that $x \in \Sigma^n$ is c-algorithmic random if $K(x) \geq n - c$.*

Proposition 3. *For all strings $x, y \in \Sigma^*$, we have:*

1) *$K(x) \leq K^t(x) + O(1) \leq |x| + 2\log(|x|) + O(1)$;*
2) *$K(x|y) \leq K(x) + O(1)$ and $K^t(x|y) \leq K^t(x) + O(1)$;*
3) *There are at least $2^n(1 - 2^{-c})$ c-algorithmic random strings of length n.*

As Solovay observed in [11], for infinitely many x, the time-bounded version of algorithmic entropy equals the unbounded version. Formally, we have:

Theorem 4. *For all time-bounds $t(n) \geq n + O(1)$ we have $K^t(x) = K(x) + O(1)$ for infinitely many x.*

As a consequence of this result, there is a string x of arbitrarily large algorithmic entropy such that $K^t(x) = K(x) + O(1)$.

Definition 5. *A semi-measure over a discrete set X is a function $f : X \to [0,1]$ such that, $\sum_{x\in X} f(x) \leq 1$. We say that a semi-measure is a measure if the equality holds. A semi-measure is constructive if it is semi-computable from below, i.e., for each x, there is a Turing machine that produces a monotone increasing sequence of rationals converging to $f(x)$.*

An important constructive semi-measure based on algorithmic entropy is defined by $\mathbf{m}(x) = 2^{-K(x)}$. This semi-measure dominates any other constructive semi-measure μ (see [6], [3]), in the sense that there is a constant $c_\mu = 2^{-K(\mu)}$ such that, for all x, $\mathbf{m}(x) \geq c_\mu\mu(x)$. For this reason, this semi-measure is called universal. Since it is natural to consider time-bounds on algorithmic entropy, we can define $\mathbf{m}^t(x)$, a time-bounded version of $\mathbf{m}(x)$.

Definition 6. *We say that a function f is computable in time t if there is a Turing machine that on the input x computes the output $f(x)$, in exactly $t(|x|)$ steps.*

Definition 7. *The t-time-bounded universal distribution is $\mathbf{m}^t(x) = c2^{-K^t(x)}$, where c is a real number such that $\sum_{x\in\Sigma^*} \mathbf{m}^t(x) = 1$.*

In [7], the authors proved that $\mathbf{m}^{t'}$ dominates distributions computable in time t, where t' is a time-bound that only depends on t. Formally:

Theorem 8 (Claim 7.6.1 [7])**.** *If μ^*, the cumulative probability distribution of μ, is computable in time $t(n)$ then for all $x \in \Sigma^*$, $\mathbf{m}^{t'}(x) \geq 2^{-K^{t'}(\mu^*)}\mu(x)$, where $t'(n) = nt(n)\log(nt(n))$.*

B. Entropies

Information theory was introduced in 1948 by C.E. Shannon [9]. Shannon entropy quantifies the uncertainty of the results of an experiment; it quantifies the average number of bits necessary to describe an outcome from an event.

Definition 9 (Shannon Entropy [9])**.** *Let $\mathcal{X}$ be a finite or infinitely countable set and let X be a random variable taking values in $\mathcal{X}$ with distribution P. The Shannon entropy of random variable X is: $H(X) = -\sum_{x\in\mathcal{X}} P(x)\log P(x)$.*

The Rényi entropy is a generalization of Shannon entropy based on a different concept of average.

Definition 10 (Rényi Entropy [8])**.** *Let $\mathcal{X}$ be a finite or infinitely countable set and let X be a random variable taking values in $\mathcal{X}$ with distribution P and let $\alpha \neq 1$ be a non-negative real number. The Rényi entropy of order α of the random variable X is defined as: $R_\alpha(X) = \frac{1}{1-\alpha}\log\left(\sum_{x\in\mathcal{X}} P(x)^\alpha\right)$.*

Another generalization of Shannon entropy is the Tsallis entropy.

Definition 11 (Tsallis Entropy [13])**.** *Let $\mathcal{X}$ be a finite or infinitely countable set and let X be a random variable taking values in $\mathcal{X}$ with distribution P and let $\alpha \neq 1$ be a non-negative real number. The Tsallis entropy of order α of the random variable X is defined as: $T_\alpha(X) = \frac{1}{\alpha-1}\left(1 - \sum_{x\in\mathcal{X}} P(x)^\alpha\right)$.*

It is easy to prove that $\lim_{\alpha\to 1} R_\alpha(X) = \lim_{\alpha\to 1} T_\alpha(X) = H(X)$.

Notice that we also use the notation $H(P)$, $R_\alpha(P)$ and $T_\alpha(P)$ to denote Shannon, Rényi and Tsallis entropies of distribution P, respectively.

Given the conceptual differences in the definitions of algorithmic entropy and Shannon entropy, it is interesting that under some weak restrictions on the distribution of the strings, they

are related. In fact, the value of Shannon entropy equals the expected value of algorithmic entropy, up to a constant term that only depends on the distribution.

Theorem 12. *Let $P(x)$ be a recursive probability distribution. Then,*

$$0 \leq \sum_x P(x)K(x) - H(P) \leq K(P)$$

Proof: (Sketch, see [7] for details). The first inequality follows directly from the known Noiseless Coding Theorem, that, for these distributions, states $H(P) \leq \sum_x P(x)K(x)$. Since $\mathbf{m}$ is universal, $\mathbf{m}(x) \geq 2^{-K(P)}P(x)$, for all x, which is equivalent to $\log P(x) \leq K(P) - K(x)$. Thus, we have:

$$\begin{aligned}\textstyle\sum_x P(x)K(x) - H(P) &= \textstyle\sum_x \big(P(x)(K(x) + \log P(x))\big)\\ &\leq \textstyle\sum_x \big(P(x)(K(x) + K(P) - K(x))\big) = K(P)\end{aligned}$$

■

III. Algorithmic entropy and entropy: how close?

Since Rényi and Tsallis entropies are generalizations of Shannon entropy, we now study if Theorem 12 can be generalized for these entropies. Then, we prove that for distributions such that the cumulative probability distribution is computable in time $t(n)$, Shannon entropy equals the expected value of the t-time-bounded algorithmic entropy.

First, we observe that the interval $[0, K(P)]$ of the inequalities of Theorem 12 is tight up to a constant term that only depends on the universal Turing machine chosen as reference. The following examples illustrate the tightness of this interval. We present a probability distribution that satisfies:

$\sum_x P(x)K(x) - H(P) = K(P) - O(1)$, with $K(P) \approx n$

and a probability distribution that satisfies:

$\sum_x P(x)K(x) - H(P) = O(1)$ and $K(P) \approx n$.

Example 13. *Fix $x_0 \in \Sigma^n$. Consider the distribution concentrated in x_0, i.e,*

$$P_n(x) = \begin{cases} 1 & \text{if } x = x_0 \\ 0 & \text{otherwise} \end{cases}$$

Notice that describing this distribution is equivalent to describe x_0. So, $K(P_n) = K(x_0) + O(1)$. On the other hand, $\sum_x P_n(x)K(x) - H(P_n) = K(x_0)$. Thus, if x_0 is c-algorithmic random, i.e. $K(x_0) \geq n - c$, then $K(P_n) \approx n$ and $\sum_x P_n(x)K(x) - H(P_n) \approx n$.

Example 14. *Let y be a string of length n that is c-algorithmic random, i.e., $K(y) \geq n - c$ and consider the following probability distribution over Σ^*:*

$$P_n(x) = \begin{cases} 0.y & \text{if } x = x_0 \\ 1 - 0.y & \text{if } x = x_1 \\ 0 & \text{otherwise} \end{cases}$$

where $0.y$ represents the real number between 0 and 1 which binary representation is y. Notice that we can choose x_0 and x_1 such that $K(x_0) = K(x_1) \leq c'$ where c' is a constant greater than 1 and hence does not depend on n. Thus,

1) *$K(P_n) \geq n - c$, since describing P_n is essentially equivalent to describe x_0, x_1 and y;*
2) *$\sum_x P_n(x)K(x) = (0.y)K(x_0) + (1 - 0.y)K(x_1) \leq 0.y \times c' + (1 - 0.y) \times c' = c'$;*
3) *$H(P_n) = -0.y \log 0.y - (1 - 0.y)\log(1 - 0.y) \leq 1$.*

Thus, $0 \leq \sum_x P_n(x)K(x) - H(P_n) \leq c' << K(P_n)$ and $K(P_n) \geq n - c$.

Now we address the question if an analogue of Theorem 12 holds for Rényi and Tsallis entropies. We show that the Shannon entropy is the only entropy that verifies simultaneously both inequalities of Theorem 12 and thus is the only one suitable to deal with information.

For every $\varepsilon > 0$, $0 < \varepsilon' < 1$, and any probability distribution P, with finite support, (see [1]), we have:

$$R_{1+\varepsilon}(P) \leq H(P) \leq R_{1-\varepsilon'}(P)$$

Thus:

1) For $\alpha \geq 1$, $0 \leq \sum_x P(x)K(x) - R_\alpha(P)$;
2) For $\alpha \leq 1$, $\sum_x P(x)K(x) - R_\alpha(P) \leq K(P)$.

It is known that for a given probability distribution with finite support, the Rényi and Tsallis entropies are monotonic increasing functions one of each other with respect to α (see [4]). Thus, for every $\varepsilon > 0$ and $0 < \varepsilon' < 1$, we also have a similar relation for the Tsallis entropy, i.e.,

$$T_{1+\varepsilon}(P) \leq H(P) \leq T_{1-\varepsilon'}(P)$$

Hence, it follows that:

1) For $\alpha \geq 1$, $0 \leq \sum_x P(x)K(x) - T_\alpha(P)$;
2) For $\alpha \leq 1$, $\sum_x P(x)K(x) - T_\alpha(P) \leq K(P)$.

In the next result we show that the inequalities above are, in general, false for different values of α.

Proposition 15. *There are recursive probability distributions P such that:*

1) *$\sum_x P(x)K(x) - R_\alpha(P) > K(P)$, where $\alpha > 1$;*
2) *$\sum_x P(x)K(x) - R_\alpha(P) < 0$, where $\alpha < 1$.*
3) *$\sum_x P(x)K(x) - T_\alpha(P) > K(P)$, where $\alpha > 1$;*
4) *$\sum_x P(x)K(x) - T_\alpha(P) < 0$, where $\alpha < 1$.*

Proof: For $x \in \Sigma^n$, consider the following probability distribution:

$$P_n(x) = \begin{cases} 1/2 & \text{if } x = 0^n \\ 2^{-n} & \text{if } x = 1x', x' \in \{0,1\}^{n-1} \\ 0 & \text{otherwise} \end{cases}$$

It is clear that this distribution is recursive. We use this distribution for some specific n to prove all items.

1) First observe that:

$$\begin{aligned} H(P_n) &= -\textstyle\sum_x P_n(x)\log P_n(x) \\ &= -\big(\tfrac{1}{2}\log\tfrac{1}{2} + \tfrac{1}{2^n}2^{n-1}\log\tfrac{1}{2^n}\big) \\ &= -\big(-\tfrac{1}{2} - \tfrac{1}{2^n}2^{n-1}n\big) = \tfrac{n+1}{2}\end{aligned}$$

Notice also that to describe P_n it is sufficient to give n, so $K(P_n) \leq c\log n$, where c is a real number.

By Theorem 12, we have,

$$\sum_x P_n(x)K(x) - H(P_n) \geq 0$$

which implies that:

$$\sum_x P_n(x)K(x) \geq \frac{n+1}{2} \qquad \text{(III.2)}$$

On the other hand, by definition:

$$\begin{aligned} R_\alpha(P_n) &= \tfrac{1}{1-\alpha}\log\textstyle\sum_x P_n(x)^\alpha \\ &= \tfrac{1}{1-\alpha}\log\big(\tfrac{1}{2^\alpha} + 2^{n-1} \times \tfrac{1}{2^{n\alpha}}\big) \\ &= \tfrac{1}{1-\alpha}\big(\log(2^{(n-1)\alpha} + 2^{n-1}) - n\alpha\big)\end{aligned}$$

To prove that $\sum_x P_n(x)K(x) - R_\alpha(P_n) > K(P_n)$, it is sufficiently to prove that:

$$\lim_n \textstyle\sum_x P_n(x)K(x) - R_\alpha(P_n) - K(P_n) > 0$$

i.e., the limit of the following expression is bigger than 0:

$$\tfrac{n+1}{2} - \tfrac{1}{1-\alpha}\big(\log(2^{(n-1)\alpha} + 2^{n-1}) - n\alpha\big) - c\log n$$

But,

$$\begin{aligned}&\lim_n \Big(\tfrac{n+1}{2} - \tfrac{\log(2^{(n-1)\alpha}+2^{n-1})}{1-\alpha} + \tfrac{n\alpha}{1-\alpha} - c\log n\Big)\\ &\geq \lim_n \Big(\tfrac{n+1}{2} + \tfrac{\log(2^{(n-1)\alpha})}{\alpha-1} - \tfrac{n\alpha}{\alpha-1} - c\log n\Big)\\ &= \lim_n \Big(\tfrac{n+1}{2} - \tfrac{\alpha}{\alpha-1} - c\log n\Big) = +\infty\end{aligned}$$

2) To prove this item we use the other inequality of Theorem 12: $\sum_x P_n(x)K(x) - H(P_n) \leq K(P_n)$, which implies that:

$$\sum_x P_n(x)K(x) \leq \frac{n+1}{2} + c\log n \qquad \text{(III.3)}$$

So,

$$\begin{aligned}&\textstyle\sum_x P_n(x)K(x) - R_\alpha(P_n) \leq\\ &\tfrac{n+1}{2} + c\log n - \tfrac{1}{1-\alpha}\big(\log(2^{(n-1)\alpha} + 2^{n-1}) - n\alpha\big)\\ &\leq \tfrac{n+1}{2} + c\log n - \tfrac{\log(2^{n-1})}{1-\alpha} + \tfrac{n\alpha}{1-\alpha}\\ &= -\tfrac{n}{2} + \tfrac{1}{2} + c\log n + \tfrac{1}{1-\alpha}\end{aligned}$$

Thus, taking n sufficiently large, the conclusion follows.

3) The Tsallis entropy of order α of distribution P_n is:

$$\begin{aligned} T_\alpha(P_n) &= \tfrac{1}{\alpha-1} - \tfrac{1}{\alpha-1}\textstyle\sum_x P_n(x)^\alpha \\ &= \tfrac{1}{\alpha-1} - \tfrac{1}{\alpha-1}\big(\tfrac{1}{2^\alpha} + \tfrac{1}{2^{n\alpha}} \times 2^{n-1}\big) \\ &= \tfrac{2^\alpha-1}{2^\alpha(\alpha-1)} - \tfrac{2^{n(1-\alpha)}}{2(\alpha-1)}\end{aligned}$$

Using the inequality III.2, we get:

$$\begin{aligned}&\textstyle\sum_x P_n(x)K(x) - T_\alpha(P_n) =\\ &\textstyle\sum_x P_n(x)K(x) - \tfrac{2^\alpha-1}{2^\alpha(\alpha-1)} + \tfrac{2^{n(1-\alpha)}}{2(\alpha-1)}\\ &\geq \tfrac{n+1}{2} - \tfrac{2^\alpha-1}{2^\alpha(\alpha-1)} + \tfrac{2^{n(1-\alpha)}}{2(\alpha-1)}\end{aligned}$$

Since $\alpha > 1$, for n sufficiently large,

$$\begin{aligned}&\tfrac{n+1}{2} - \tfrac{2^\alpha-1}{2^\alpha(\alpha-1)} + \tfrac{2^{n(1-\alpha)}}{2(\alpha-1)} \geq\\ &\geq \tfrac{n+1}{2} - \tfrac{2^\alpha-1}{2^\alpha(\alpha-1)} > c\log n = K(P_n)\end{aligned}$$

4) Using the inequality III.3, we get:

$$\begin{aligned}&\textstyle\sum_x P_n(x)K(x) - T_\alpha(P_n) \leq\\ &\leq \tfrac{n+1}{2} + c\log n - \tfrac{2^\alpha-1}{2^\alpha(\alpha-1)} + \tfrac{2^{n(1-\alpha)}}{2(\alpha-1)}\end{aligned}$$

Since $\alpha < 1$, for n sufficiently large, we conclude that:

$$\tfrac{n+1}{2} + c\log n + \tfrac{2^\alpha-1}{2^\alpha(1-\alpha)} - \tfrac{2^{n(1-\alpha)}}{2(1-\alpha)} < 0$$

■

The proof of the following theorem is similar to the previous proposition.

Theorem 16. *For every $\Delta > 0$ and $\alpha > 1$ there are recursive probability distributions P such that,*

1) *$\sum_x P(x)K(x) - R_\alpha(P) \geq (K(P))^\alpha$;*
2) *$\sum_x P(x)K(x) - R_\alpha(P) \geq K(P) + \Delta$;*
3) *$\sum_x P(x)K(x) - T_\alpha(P) \geq (K(P))^\alpha$;*
4) *$\sum_x P(x)K(x) - T_\alpha(P) \geq K(P) + \Delta$.*

The previous results show that only Shannon entropy is suitable for the inequalities of Theorem 12. Now, we analyze if a similar relation holds in a time-bounded algorithmic entropy scenario.

If instead of considering $K(P)$ and $K(x)$ in the inequalities of Theorem 12, we use their time-bounded version and imposing some computational restrictions on the distributions, we obtain a similar result. Notice that, for the class of distributions on the following theorem, the entropy equals (up to a constant) the expected value of time-bounded algorithmic entropy.

Theorem 17. *Let P be a probability distribution over Σ^n such that P^*, the cumulative probability distribution of P, is computable in time $t(n)$. Setting $t'(n) = O\big(nt(n)\log(nt(n))\big)$, we have, $0 \leq \sum_x P(x)K^{t'}(x) - H(P) \leq K^{t'}(P^*)$.*

Proof: The first inequality follows directly from the similar inequality of Theorem 12 and from the fact that $K^t(x) \geq K(x)$.

From Theorem 8, if P is a probability distribution such that P^* is computable in time $t(n)$, then for all $x \in \Sigma^n$ and $t'(n) = nt(n)\log(nt(n))$, $K^{t'}(x) + \log P(x) \leq K^{t'}(P^*)$. Then, summing over all x, we get:

$$\sum_x P(x)(K^{t'}(x) + \log P(x)) \leq \sum_x P(x)K^{t'}(P^*)$$

which is equivalent to $\sum_x P(x)K^{t'}(x) - H(P) \leq K^{t'}(P^*)$.

■

IV. On the entropies of the time-bounded universal distribution

If the time that a program can use to produce a string is bounded, we get the so called time-bounded universal distribution, $\mathbf{m}^t(x) = c2^{-K^t(x)}$. In this section, we study the convergence of the three entropies with respect to this distribution.

Theorem 18. *The Shannon entropy of the distribution* $\mathbf{m}^t$ *diverges.*

Proof: If $x \geq 2$ then $f(x) = x2^{-x}$ is a decreasing function. Let A be the set of strings such that $-\log \mathbf{m}^t(x) \geq 2$. A is recursively enumerable.

$$\begin{aligned} H(\mathbf{m}^t) &= \textstyle\sum_{x\in\Sigma^*} -\mathbf{m}^t(x)\log \mathbf{m}^t(x) \\ &\geq \textstyle\sum_{x\in A} -\mathbf{m}^t(x)\log \mathbf{m}^t(x) \\ &= \textstyle\sum_{x\in A} c2^{-K^t(x)}(K^t(x) - \log c) \\ &= -c\log c \textstyle\sum_{x\in A} 2^{-K^t(x)} + c\sum_{x\in A} K^t(x)2^{-K^t(x)} \end{aligned}$$

So, if we prove that $\sum_{x\in A} K^t(x)2^{-K^t(x)}$ diverges, the result follows. Assume, by contradiction, that $\sum_{x\in A} K^t(x)2^{-K^t(x)} < d$ for some $d \in \mathbb{R}$. Then, considering

$$r(x) = \begin{cases} \frac{1}{d}K^t(x)2^{-K^t(x)} & \text{if } s \in A \\ 0 & \text{otherwise} \end{cases}$$

we conclude that r is a semi-measure. Thus, there is a constant c' such that, for all x, $r(x) \leq c'\mathbf{m}(x)$. Hence, for $x \in A$, we have $\frac{1}{d}K^t(x)2^{-K^t(x)} \leq c'2^{-K(x)}$.
So, $K^t(x) \leq c'd2^{K^t(x)-K(x)}$, which is a contradiction since by Theorem 4, A contains infinitely many strings x of time-bounded algorithmic entropy larger than a constant such that $K^t(x) = K(x) + O(1)$. The contradiction followed from the assumption that the sum $\sum_{x\in A} K^t(x)2^{-K^t(x)}$ converges. So, $H(\mathbf{m}^t)$ diverges. ■
Now we show that, similarly to the behavior of Rényi and Tsallis entropies of universal distribution $\mathbf{m}$ (see [12]), $R_\alpha(\mathbf{m}^t) < \infty$ iff $\alpha > 1$ and $T_\alpha(\mathbf{m}^t) < \infty$ iff $\alpha > 1$. First obverse that we have the following ordering relationship between these two entropies for all probability distribution P:

1) If $\alpha > 1$, $T_\alpha(P) \leq \frac{1}{\alpha-1} + R_\alpha(P)$;
2) If $\alpha < 1$, $T_\alpha(P) \geq \frac{1}{\alpha-1} + R_\alpha(P)$.

Theorem 19. *The Tsallis entropy of order* α *of time-bounded universal distribution* $\mathbf{m}^t$ *converges iff* $\alpha > 1$.

Proof: From Theorem 8 of [12], it is known that $\sum_{x\in\Sigma^*}(\mathbf{m}(x))^\alpha$ converges if $\alpha > 1$. Since $\mathbf{m}^t$ is a probability distribution, there is a constant λ such that, for all x, $\mathbf{m}^t(x) \leq \lambda\mathbf{m}(x)$. So, $(\mathbf{m}^t(x))^\alpha \leq (\lambda\mathbf{m}(x))^\alpha$, which implies that $\sum_{x\in\Sigma^*}(\mathbf{m}^t(x))^\alpha \leq \lambda^\alpha \sum_{x\in\Sigma^*}(\mathbf{m}(x))^\alpha$ from which we conclude that for $\alpha > 1$, $T_\alpha(\mathbf{m}^t)$ converges.
For $\alpha < 1$, the proof is analogous to the proof of Theorem 18. Suppose that $\sum_{x\in\Sigma^*}(\mathbf{m}^t(x))^\alpha < d$ for some $d \in \mathbb{R}$. Hence, $r(x) = \frac{1}{d}(\mathbf{m}^t(x))^\alpha$ is a constructible semi-measure. Then, there is a constant τ such that for all $x \in \Sigma^*$, $r(x) = \frac{1}{d}\left(c2^{-K^t(x)}\right)^\alpha \leq \tau 2^{-K(x)}$, which is equivalent to $\frac{c^\alpha}{d\tau} \leq 2^{\alpha K^t(x)-K(x)}$. By Theorem 4, there are infinitely many strings x such that $K^t(x) = K(x) + O(1)$. Then it would follow that for these strings $\frac{c^\alpha}{d\tau} \leq 2^{(\alpha-1)K(x)}$, which is false since these particular strings can have arbitrarily large algorithmic entropy. ■

Theorem 20. *The Rényi entropy of order* α *of time-bounded universal distribution* $\mathbf{m}^t$ *converges iff* $\alpha > 1$.

Proof: For $\alpha > 1$, since $\sum_x 2^{-K^t(x)} < +\infty$, we have $\sum_x (2^{-K^t(x)})^\alpha < \infty$. Thus, $R_\alpha(\mathbf{m}^t)$ converges.
For $\alpha < 1$, suppose without loss of generality that α is rational (otherwise take another rational slightly larger than α). Assume that $\sum_x (2^{-K^t(x)})^\alpha < \infty$. Then by universality of $\mathbf{m}$ and since $(2^{-K^t(x)})^\alpha$ is computable, we would have $\mathbf{m}(x) \geq (2^{-K^t(x)})^\alpha$ which, by taking logarithms, is equivalent to $K^t(x) \geq \frac{1}{\alpha}K(x) + O(1)$. Since $(1/\alpha) > 1$, this would contradict Solovay's Theorem of page 2. ■

V. Conclusions

We showed that Shannon entropy is the only entropy that satisfies the relation with the expected value of algorithmic entropy stated in [7], exhibiting a probability distribution for which the relation fails for some orders of Tsallis and Rényi entropies. Furthermore, we proved that under the assumption that cumulative probability distribution is computable in an alloted time, a time-bounded version of the same relationship is verified with respect to Shannon entropy. Since it is natural to define a probability distribution based on time-bounded algorithmic entropy, we studied the convergence of this distribution under Shannon entropy and its two generalizations: Tsallis and Rényi entropies.

Acknowledgments

All the authors are partially supported by the program CSI^2 (PTDC/EIA-CCO/099951/2008) granted by FCT to Instituto de Telecomunicações. The first author is also supported by the grant SFRH/BD/33234/2007 of FCT. The second author is also supported by the grant SFRH/BD/28419/2006 of FCT. The third author is also partially funded by LIACC through Programa de Financiamento Plurianual, FCT.

References

[1] C. Cachin, P. Dr, and U. Maurer. Entropy Measures and Unconditional Security in Cryptography, 1997.
[2] G. Chaitin. On the Length of Programs for Computing Finite Binary Sequences. *J. ACM*, 13(4):547–569, 1966.
[3] P. Gacs. On the symmetry of algorithmic information. *Soviet Mathematics Doklady*, 1974.
[4] S. Hoffmann. Proceedings of ieeenon-uniqueness of non-extensive entropy under rényi's recipe. Technical report.
[5] A. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.
[6] L. Levin. Laws of Information Conservation (Nongrowth) and Aspects of the Foundation of Probability Theory. *Prob. Peredachi Inf.*, 10(3), 1974.
[7] M. Li and P. Vitányi. An Introduction to Kolmogorov Complexity and Its Applications. Springer Publishing Company, Incorporated, 2008.
[8] A. Rényi. On Measures of Entropy and Information. In *Berkeley Symposium Mathematics, Statistics, and Probability*, 1960.
[9] C. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.
[10] R. Solomonoff. A formal theory of inductive inference, part I. *Information and Control*, 7(1):1–22, 1964.
[11] R. Solovay. unpublished manuscript, IBM Thomas J. Watson Research Center. *Draft of a paper (or series of papers) on Chaitin's work*, 1975.
[12] K. Tadaki. The Tsallis entropy and the Shannon entropy of a universal probability. *CoRR*, abs/0805.0154, 2008.
[13] C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. Stat. Physics*, 52:479–487, 1988.