# UNIVERSAL SOURCE CODING

A thesis submitted to the University of Copenhagen
for the degree of Master of Science
in the Faculty of Mathematics

July 2001

By
Peter Andreasen
University of Copenhagen
Department of Mathematics
Universitetsparken 5
DK-2100 Copenhagen

**Abstract**

The material covered in this thesis falls mainly in two parts. In the first part, which is found in chapter II, the foundations of source coding are considered. A careful introduction to the notion of codes for symbols, blocks and infinite sequences is given. The classical notion of a stationary ergodic source is then introduced as a natural concept arising when assuming limiting statistical consistency of infinite sequences. Using the recently developed techniques of packing and counting, a proof of the entropy-rate theorem (the Shannon-McMillan-Brieman theorem) is given, which ultimately leads to an almost sure version of Shannons coding theorem due to Barron. As a parallel to this, a result of Ryabko, which might be thought of as a coding theorem for combinatorial (non-probabilistic) sources, is presented. The first part of the thesis closes with a brief overview relating the theoretical bounds to codes that are practically implementable on a computer.

In the second part, which is found in chapters III and IV, the framework developed in the first part is applied to the well known and widely used Lempel-Ziv codes. Using the 1978 variation as prime example, a description of the algorithm is given, and a proof that the corresponding code is weakly universal on the class of stationary ergodic sources is made. This proof is based, once again, on the packing and counting techniques forwarded by Ornstein, Weiss and Shields. In order to gain better insight into the Lempel-Ziv code a symbol-wise analysis of the code is conducted and a recently developed proof (due to Yang and Kieffer) of the second order performance of the Lempel-Ziv code when applied to an i.i.d. source is given. Finally, a survey of recent results from the non-probabilistic domain is presented. It covers the challenges of individual coding, and the ideas of Shields, Ryabko and Suzuki regarding the use of individual sequences and Hausdorff dimension as a new method for comparing universal codes.

# Contents

# Preface

*On two occasions I have been asked [by members of Parliament!], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.*

CHARLES BABBAGE

The most important questions may very well be those that we ask ourselves. In writing this thesis I have been privileged to be able to pose what I thought was important questions about information theory. While some questions turned out to be the "wrong" questions (in that their answers were trivial or the questions were not so well posed), other questions compensated amply for this by leading to new insights, to theorems that I had not heard of before and sometimes even to new questions. Studying the research literature is always exciting but it is particular rewarding when done in order to satisfy ones own curiosity. However, it would be impractical to arrange the material of this thesis as a list of questions and corresponding answers, so a more traditional structure has been chosen. It is my hope, that the following pages still carries an air of exploration that will be enjoyed by the reader as it has been by the author.

## Mathematical conventions

Logarithms are written $\log x$ and taken to the base of 2, that is, $2^{\log x} = x$ for any positive value $x$. For the natural logarithm, we write $\ln x$. We will write $[0, 1]$ for the interval of the real numbers ranging from 0 to 1 including both endpoints and use $[0, 1[$ to denote the half-open interval from 0 to 1 including 0 but not 1. When dealing with higher order functions, for example $f : X \times Y \to Z$, we will for any given $x \in X$ write $f(x, \cdot)$ for the function which maps $y \mapsto f(x, y)$. We write $\mathbb{N}$ for the natural numbers $(1, 2, 3, \ldots)$ and $\mathbb{N}_0$ for the non-negative integers.

Sequences of symbols play an important role in information theory. In order to maximize the clarity, we use the following convention: A bold typeface like this, $\mathbf{x}$, is used for infinite sequences, $\mathbf{x} \in A^\infty$. A prefix of $\mathbf{x}$ is denoted $x_1^n$. Indeed, any finite sequence of length, say $m$, is written as $x_1^m$. The notation $x_i$ is used for the i'th symbol of either $\mathbf{x}$ of $x_1^n$, whichever applies in the context. Note, that the term *sequence* may be used for both finite and infinite (indexed by the positive integers) sequences of symbols. The term *string* is used as a synonym for *sequence*.

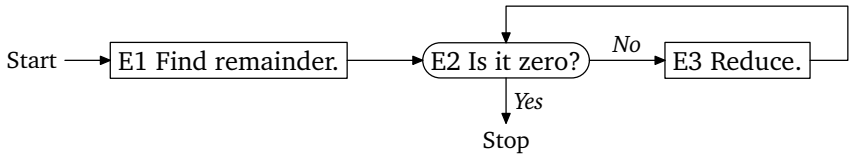



Figure 1: Flowchart for the example algorithm, Euclid's algorithm, described in §1. Algorithms that are presented in the text will by accompanied by flowcharts like this.

## Typographical conventions

In this thesis most paragraphs are numbered for easy cross-reference. All numbered paragraphs have a name, for example **definition**, **theorem** or **proof**. Apart from these well-known mathematical terms, a few other names are used fairly often: Paragraphs named **observation** contain mathematical discussions while the name **remark** is used for material of more auxiliary nature, such as references to the literature, meta-comments about why a specific formulation has been chosen and so forth. **Examples** can be either illustrations of mathematical concepts already presented or descriptions of applications. Paragraphs with names other than these are mostly found in the sections entitled *Notes and references* which are located at the end of each chapter.

The description of algorithms presents a particular difficult typographical set of problems and the choice of any particular method will bring advantages as well as disadvantages. The notation we have chosen, is not close to any real computer language, which is consistent with the fact that the emphasis of this thesis is not on the implementational aspects of the algorithms. The notation we use is very similar to that of [Knuth, 1997], and consists of an in-text description of the steps that makes up the algorithm together with a corresponding flow-chart. As introduction to the method we have chosen to present the well-known Euclid's algorithm:

§1 **Algorithm** (EUCLID'S ALGORITHM). Given two positive integers $m$ and $n$, find their *greatest common divisor*, that is, the largest positive integer that evenly divides both $m$ and $n$. The greatest common divisor of $m$ and $n$ is written $\gcd(m,n)$.

**Input:** Two positive integers $m$ and $n$.

**Output:** A positive integer, $\gcd(m,n)$.

**E1** [Find remainder.] Divide $m$ by $n$ and let $r$ be the remainder. (We will have $0 \leq r < n$.)

**E2** [Is it zero?] If $r = 0$, the algorithm terminates; $n$ is the output.

**E3** [Reduce.] Set $m \leftarrow n$, $n \leftarrow r$ and go back to step **E1**.

It is easy to follow the steps in the algorithm and carry out the algorithm "by hand", that is, using no other tools than pencil and paper. The flow-chart of the algorithm is found in figure 1. Once we have acquired this understanding of the workings of the algorithm, we may choose to implement it in the computer language of our preference.

## Acknowledgements

This thesis could not have been completed without help from a number of people, and thus the preface is not complete without a section of acknowledgements. I would like to thank: My advisor, Flemming Topsøe, for teaching me the foundations of information theory and for creating an inspiring environment of seminars, workshops etc. here in Copenhagen. Boris Ryabko, for writing the papers[1] that inspired the theme of the thesis and for many exciting and fruitful discussions. My local typesetting wizard, Christian Tønsberg, for helping out with more or less arcane LaTeX macros. Morten Højsgaard for looking at the manuscript from a different angle and for much sound advice. And Jacob Weismann, for our invaluable discussion sessions at the white-board, your detailed proofreading and comments on drafts, and, of course, for introducing me to information theory in the first place; it has been quite a ride!
Finally, I owe many thanks to my wife, Maj-Britt, for putting up with me occupying every visible desk space with notes and articles, and in general being a nuisance; I'll try to snap out of it now!

Peter Andreasen
Copenhagen, July 2001

[1] Mainly [Ryabko, 1986] and [Hojo et al., 1998].

CHAPTER I

# Introduction

*My sources are unreliable, but their information is fascinating.*

ASHLEIGH BRILLIANT

After more than fifty years, the field of information theory retains a sense of freshness and creativity. One reason for this is that a large part of the research still takes place in the fertile area where theory meets application. On a regular basis new theoretical developments provides new applications, just as practical needs arises and calls for new directions in the theoretic realm to be explored. Another reason is that information theory interacts actively with a number of other theoretical fields such as probability theory, fractal geometry and algorithmic complexity theory.

While this composite nature of information theory is a great inspiration for the participants in the field, at the same time, it makes it harder to agree on the—for lack of a better phrase—foundational definitions, upon which the theory is built. This goes in particular for the relation to physical applications (like data compression in computers) which are the foundation that we historically build on (see [Shannon, 1948]). But also when using information theoretical tools in purely theoretical situations, like when interacting with other branches of mathematics, we sometimes find results hard to compare or apply because of differences in the basic assumptions. Hence contrary to more abstract and/or self-contained branches of mathematics, where the promise of interesting theorems is enough to justify the starting definitions, we may feel the need for a more careful argument for the soundness of the basic definitions of information theory. Indeed, if for no other reason than the fact that the people of information theory come from such diverse backgrounds as probability theory, statistics, computer science and physics, we may at least expect that a bit of work is required to put the results in a common framework.

In a section below we state, in a more formal way, the goals of this thesis. Before doing so, the author would like to present the backdrop of contemplations that inspired these goals.

As mentioned, the foundation of source coding is historically to be found in practical applications. Hence a word about the application (in the traditional, physical sense) of mathematics is in order. Commonly we base applications on the following methodology: First we select an appropriate mathematical model which describes (to some extend) the physical reality we encounter. Secondly we deduce mathematical results (theorems and such) about the model. Finally we apply the results to the physical reality, hoping to achieve a certain effect and gain greater insight. This methodology is illustrated in figure 1.

The success of this method depends partly on how well the mathematical model describes the reality and partly on the kind of results we are able to prove about the model. We shall deftly

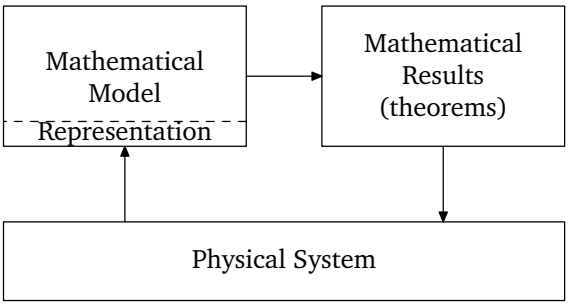



Figure 1: A simplistic view of how application of mathematics works

sidestep the philosophical question of whether a well-fitting model combined with strong theorems automatically guarantees relevant applications to reality!

This whole issue is reflected as early as Shannon, who writes:

> *We can think of a discrete source as generating the message, symbol by symbol. It chooses successive symbols according to certain probabilities depending, in general, on preceding choices as well as the particular symbols in question. A physical system, or a mathematical model of a system which produces such a sequence of symbols governed by a set of probabilities is known as a stochastic process. Conversely, any stochastic process which produces a discrete sequence of symbols chosen from a finite set may be considered a discrete source.* *[Shannon, 1948, Section I.2]*

Here we find the notions of *physical system* as well as that of a *mathematical model*, and we see how one often use terms originating from the physical system ("produce", "choose" and so forth) in the context of the model. But what is probably even more important is the two assumptions made about the setup, namely that we only consider discrete sources and that we assume the message to be produced according to a stochastic law. In practice these two assumptions takes on different flavors: The first is usually made with great confidence: If the message ticks in on a teletype machine, we do not expect Chinese symbols to show up, nor do we expect two symbols to arrive simultaneously. Both would be physically impossible. The second assumption, however, is usually made with less confidence: apart from messages produced by tossing a coin or rolling a dice, most messages are not clearly of stochastic nature. Typing in Shakespeares works is not what we would typically describe as " ... [choosing] successive symbols according to certain probabilities depending, in general, on preceding choices ... ", to quote [Shannon, 1948] again.

Furthermore, most applications would break completely down if the assumption about a discrete source producing a finite number of distinct symbols did not hold. On the other hand, in many applications we gladly employ a stochastic assumption even when data is not produced by anything similar to a coin tossing. The Morse code illustrates this: Even though the code was designed with the English language in mind (by keeping the Morse code short for letters frequently used in English) it works for any message, as long as it is written using the Latin letters.

For many reasons the distinction between the mathematical representation and the mathematical model is even today not particularly prevalent. Often the representation and the model is introduced at the same time and with the same degree of obviousness, even though (as explained above) only one of these are easily defensible on physical grounds. This does not imply that the stochastic model is wrong, only that it is not exclusively true, and that alternatives might exist. It was this interest in the possibility of other source models which led to the works of Billingsley, Ryabko and others, regarding the use of Hausdorff dimension in information theory. Thus, in informal terms, the goal of the thesis has from the onset been to study the foundations of source coding, in particular he ubiquitous notion of the source.

## Goal

The goal of this thesis is to conduct a study of the foundations of universal source coding through presentation of stochastic as well as non-stochastic results. By the selection of results and techniques of proofs, we will explore the vast depth and scope of the source concept, pushing the concept of the source to its limit. It is our hope, that our contribution will be: firstly, a single framework capable of containing all the results; secondly, a presentation of state-of-the art results and methods of proofs until now only found in the research literature; thirdly, a number of observations pointing in the direction of future research.

As indicated, the method for reaching the goal will not be a philosophical discussion of the foundations of universal source coding, but a treatment of a selection of results that expands our understanding of the subject. The means to the end will more specifically be,

- To state the foundational definitions of source coding in a common framework leading to well posed coding problems. In the case of the stochastic setup, to test the hypothesis that we may introduce this setup by means of infinite sequences over a finite alphabet; and to apply recently developed techniques in the proof of the classical theorems. In the non-stochastic case, to prove the coding theorem for combinatorial sources and to obtain insight into the connection between the two setups.
- To conduct a study of the Lempel-Ziv codes using recently developed techniques and to arrive at an understanding of to what extend comparison of weakly universal codes is possible in the stochastic setup and how using Hausdorff dimension may provide an alternative method of comparison.

While these topics are of interest in their own right, we encourage the reader to keep the main goal in mind as well.

> *The confusion of models and reality has often caused people to try an ascribe a coin flipping process an inherent probability, ... Such a thinking is wrong and besides quite unproductive. It is never the case that a real world machinery producing the observed data is or defines a 'true' distribution and a random variable.*
>
> JORMA RISSANEN IN [RISSANEN, 1989].

CHAPTER II

# The framework

*"Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."*

LEWIS CARROLL

## Alphabets, sequences and sources

§2 **Definition** (LETTERS AND SEQUENCES). Let $A$ be a finite set, called the *alphabet*. The elements of $A$ are known as *letters* or *symbols*. When convenient, we will enumerate the symbols of $A$ and write $A = \{0, \dots, k-1\}$ where $k = |A|$ is the cardinality of $A$. A *sequence* of $n$ letters from $A$ is written as either $x_1 \cdots x_n$ or, shorter, $x_1^n$, or simply $x$. If $x = x_1^n = x_1 \cdots x_n$ and $y = y_1^m = y_1 \cdots y_m$ are sequences, we write $xy$ for the *concatenation* $xy = x_1 \cdots x_n y_1 \cdots y_m$. The empty sequence (that is, the sequence of $0$ letters) is denoted $\lambda$ and we have $x\lambda = \lambda x = x$ for any sequence $x$.

§3 **Definition** (SETS OF SEQUENCES). A sequence of symbols from $A$ is known as a sequence over $A$. The set of all sequences over $A$ of length $n$ is denoted $A^n$ and the union of these sets, that is, the set of all finite length sequences of symbols from $A$, is denoted $A^*$. We write $A^\infty$ for the set of infinite sequences of symbols from $A$ indexed by the positive integers, that is, $A^\infty \stackrel{\text{def}}{=\!=} A^{\mathbb{N}}$.

§4 **Definition** (THE PREFIX PROPERTY). The sequence $x$ is a *prefix* of the sequence $y$ if $y = xz$ for some sequence $z$. The sequence $x$ is a *suffix* of the sequence $y$ if $y = zx$ for some sequence $z$. A set $P \subset A^*$ is said to be *prefix free* (or simply *prefix*) if no element of $P$ is a prefix of another element of $P$. A prefix set $P \subset A^*$ is said to be *complete*, if adding any sequence to $P$ would ruin the prefix condition, that is, if

$$w \in A^* \backslash P \Rightarrow P \cup \{w\} \text{ is not a prefix set.}$$

§5 **Definition**. A *source instance* over the alphabet $A$ is an infinite sequence of symbols from $A$, that is, it is an element $\mathbf{x} \in A^\infty$ or equivalently a map $\mathbf{x} : \mathbb{N} \to A$.

§6 **Observation**. The previous definition (§5) is an attempt to create a mathematical representation in agreement with the comments made in the introduction. We want to treat the source as a black box and only put into the definition what can be seen from outside which is, that as time progresses the source is producing a sequence of symbols. At time $t \in \mathbb{N}$ the message emitted so far by the source instance $\mathbf{x}$ is thus the sequence $x_1 x_2 \cdots x_t$. We should note, that it is entirely

possible to define a *two-sided* source instance, by means of doubly infinite sequences, $A^{\mathbb{Z}}$. In some cases this makes sense from a mathematical point of view, but the practical interpretation of an infinite, unknown past is unclear.

§7 **Remark**. We use the term *representation* as opposed to *model* for the concept of source instances. This is to enable us to distinguish between those fundamental assumptions which we consider inherent to the reality encountered and those assumptions which we make in the process of modeling the source instance. The following example illustrates the difference we put in the meaning of the two terms.

§8 **Example**. At time $t = 1$ we start an automatic "coin-flipper", that flips a coin labeled '0' on one side and '1' on the other. This physical system is *represented* as a single sided source instance, $x$, over the alphabet $A = \{0, 1\}$, that is, $x \in A^{\mathbb{N}}$. Say we want to predict the output of the coin-flipper. To do this we need a *model* capable of producing guesses. Now we are quite interested in being able to prove theorems about this model. Are there better models? Is it a good guesser? On the other hand, we do not even consider the case that the coin suddenly shows a side labelled $\alpha$. We never ask if there is a better representation because we are convinced that the physical system will not suddenly behave different in this respect. This corresponds to many (but definitely not all) practical encountered situations.

Assume that, at the same time as in the above example, we start another physical system consisting of a black box which emits one of the symbols $\{0, 1\}$ whenever the first machine emits a symbol. This second system is represented as another source instance, $x'$. Notice how our definition of a source instance only covers the *output* of a physical system and does not say anything about what goes on inside the system. The new system might even be an exact replica of the first one, in which case the origin of the word *instance* in the term source instance becomes clear. In short, the source instance concept enables us to represent a system which we know nothing about just as easily as a system which is in a clear box and maybe even clearly stochastic of nature.

§9 **Remark**. Even though the concept of source instances is a firm mathematical foundation, it is not very useful to prove theorems about single source instances. We would rather have results holding for a whole *class* of source instances. Note, however, that often a the class of source instances is brought into play in a quite non-constructive way, for example by stating that $S$ is the subset of $A^\infty$ for which property $P$ holds. Hence we must take care to allow for arbitrary classes of source instances.

§10 **Definition**. Any subset $S \subseteq A^\infty$ of the infinite sequences over $A$ is known as a *combinatorial source* or simply as a *source*.

§11 **Summary**. We have introduced the notational tools for dealing with sequences of symbols over finite alphabets. By considering the practical aspect of the concept of a source, we have argued that the notion of source instances is intuitively attractive. The broadest possible notion of a source has been introduced. It remains to be seen, of course, if these foundational definitions are useful, that is, if it is possible to prove interesting theorems based on them.

## Codes

§12 **Remark**. It has been said that information is like energy in that we are concerned about the *conversion*, *transmission* and *storage* of both. One way to convert information is coding. Loosely speaking, coding is about converting a sequence of symbols (the message) from one representation into another representation (the codeword), that may even be expressed in another alphabet. There are several possible goals one may seek by the encoding. The most commonly studied is to get a *shorter* representation of the information, and this goal will be our focus. Thus whenever we mention the concept of encoding information it should be understood that the underlying wish is to transform the information into a shorter representation.
The present section serves three purposes: Firstly, it provides the notational framework for studying codes. Secondly, it presents the concept of prefix codes as well as the well-known theorems of Kraft and McMillan. This theme albeit well known is, we believe, of some importance because the Kraft and McMillan theorems are often introduced as "those theorems that link coding with entropy"; or the whole prefix concept is presented as "the property we need in order to get instant decoding". This may lead to the understanding that the prefix property is something needed for practical purposes or as a necessary technical assumption for the mathematics to work out. It is our hope to show that although the above quoted statements are true, we are on even firmer ground than that; at least when we consider the goal of coding to be compact representation. Thirdly, we introduce a code concept on $A^\infty$.

§13 **Example** (PER-SYMBOL REPRESENTATION). Consider the task of encoding the two messages, "`kolmogorov`" and "`shannon`" into binary codewords. In a first attempt, we might consider using some injective map $\varphi : A \to \{0,1\}^*$, where $A$ denotes the usual English alphabet, and proceed to create the following encoding:

| Message | Codeword |
|---|---|
| `kolmogorov` | $\varphi(\mathtt{k})\varphi(\mathtt{o})\varphi(\mathtt{l})\varphi(\mathtt{m})\varphi(\mathtt{o})\varphi(\mathtt{g})\varphi(\mathtt{o})\varphi(\mathtt{r})\varphi(\mathtt{o})\varphi(\mathtt{v})$ |
| `shannon` | $\varphi(\mathtt{s})\varphi(\mathtt{h})\varphi(\mathtt{a})\varphi(\mathtt{n})\varphi(\mathtt{n})\varphi(\mathtt{o})\varphi(\mathtt{n})$ |

However, there is a fundamental problem with this approach. While the injectiveness of the map $\varphi$ ensures that each symbol of $A$ is represented by a unique binary sequence, this does not imply uniqueness of the codewords for each message. In fact the two messages `kolmogorov` and `shannon` might be encoded into the very same codeword! The catch here is that we are loosing information in our representation of the messages, namely that of where the individual symbols in the message begin and where they end. In the original message there is no such ambiguity since the very nature of sequences ensures that the symbols are distinct and clearly separated from each other. Yet by our transformation into codewords, this information is lost. We may think of it as writing englishwithoutspaces or sending Morse signals without pauses: ———. Inserting a delimiting symbol (like this: english_without_spaces and ·_·_·_—_—_·_·_·) whould apparently solve the problem, but it would also correspond to the extension of the target alphabet so that $\varphi : A \to \{0,1,\_\}$ so this clearly leads nowhere.

§14 **Example** (PER-MESSAGE-LENGTH REPRESENTATION). As shown by the previous example, it may not be completely trivial to capture the concept of *unique representation* in mathematical terms. We now give a second try by seeking representations of the messages themselves in place of just

the letters they are made up from. We introduce injective mappings[1] $\varphi_n : A^n \to \{0,1\}^*$, one map for each $n \in \mathbb{N}$. The encoding now looks like this:

| Message | Codeword |
|---|---|
| `kolmogorov` | $\varphi_{10}(\texttt{kolmogorov})$ |
| `shannon` | $\varphi_7(\texttt{shannon})$ |

Unfortunately this method suffers similar problems as the previous one. By definition the message to be represented has a specific length, but this information is lost when using $\varphi_{10}$, resp. $\varphi_7$, and thus from looking at the codewords we can not know which map was used. Just as before we might end up with two identical binary codewords, and only information about the lenght of the message that each of them represent will permit us to know the message.

§15 **Example** (GLOBAL REPRESENTATION). Our final approach begins with an injective map $\varphi_* : A^* \to \{0,1\}^*$. This time the codewords look like this,

| Message | Codeword |
|---|---|
| `kolmogorov` | $\varphi_*(\texttt{kolmogorov})$ |
| `shannon` | $\varphi_*(\texttt{shannon})$ |

,

and this method is fundamentally better. The codewords of the two messages are guaranteed to be different, and this captures our original intent which was to be able to express *any* message as an unique codeword.

§16 **Remark**. The previous examples (§13-§15) might seem overly trivial, but they do provide rationale for some important concepts leading to prefix codes and the Kraft inequality. While the representation of messages on a per-symbol level (example §13) or on a per-message-length level (example §14) are attractive in their simplicity (viz. their finiteness), they have some defects as we have seen. We are, however, going to fix these problems and we will see, that in doing so the three types of representations become equivalent in a certain sense. In the following we consider first the problem of example §13, in search for a solution to the concatenation problem. Afterwards we treat the case of example §14 handling the problem of message lengths.

§17 **Definition**. Let $B$ be a finite alphabet. A set $U \subset B^*$ is *uniquely concatenable* if the mapping $(w_1, \ldots, w_k) \mapsto w_1 \cdots w_k$ creates a one to one correspondance between the elements of the union of the cartesian products, $\cup_{k=1}^{\infty} U^k$, and the set of concatenated strings $\cup_{k=1}^{\infty} \{w_1 \cdots w_k \mid (w_1, \ldots, w_k) \in U^k\}$.

§18 **Example**. Let $B = \{0,1\}$ and set $U = \{00, 100, 10\}$, then $U$ is an uniquely concatenable set. The set $V = \{0, 10, 010\}$, on the other hand, is not uniquely concatenable because the two different elements $(0, 10)$ and $(010)$, both of which are members of $V \cup (V \times V)$, are mapped to the same sequence: 010. Also note that any alphabet $B$ is trivially uniquely concatenable when considered as the set of sequences of length exactly 1.

§19 **Observation**. A prefix set (see definition §4) is uniquely concatenable, for assume $V \subseteq B^*$ is a

[1] The notation $\varphi_n$ is used in place of the tempting $\varphi^n$ to indicate that in general there is no correlation between, say, $\varphi_n$ and $\varphi_{n+1}$. This should be compared to the notation $A^n$ where we have $A^n \times A = A^{n+1}$.

prefix set of sequences over the alphabet $B$ and that $w_1 \cdots w_k = v_1 \cdots v_l$ for $w_i, v_j \in V$. Then $w_1$ must be a prefix of $v_1$ or $v_1$ a prefix of $w_1$ or they must be equal. Because $w_1, v_1 \in V$ and $V$ is prefix free, it follows that $w_1 = v_1$. Repeating the argument shows that $k = l$ and that $w_i = v_i$ for all $1 \leq i \leq k$. Hence the map of definition §17 is a bijection and then $V$ is uniquely concatenable. On the other hand, we have already encountered uniquely concatenable sets, which are not prefix sets, namely the set $U$ of example §18. We are going to see, however, that prefix sets are canonical examples of uniquely concatenable sets.

§20 **Observation**. The concept of uniquely concatenable sets can be used to solve the problem of by-letter representation of example §13. If the set $\varphi : A \to B^*$ is an injective map between for finite alphabets $A$ and $B$, and if $\varphi(A)$ is uniquely concatenable, then any message $a_1^n \in A^n$ can be uniquely represented as a word over $B$, namely $\varphi(a_1) \cdots \varphi(a_n)$. This effect is formalized in the following theorem.

§21 **Theorem**. *Let $A$ and $B$ be finite alphabets and let $\varphi : A \to B^*$ be an injective map. If $\varphi(A)$ is an uniquely concatenable set then the map $\varphi_* : A^* \to B^*$ defined by $\varphi_*(x_1^n) \stackrel{\text{def}}{=\!=} \varphi(x_1) \cdots \varphi(x_n)$ is injective. Conversely, if $\varphi_* : A^* \to B^*$ is an injective map and if $\varphi_*(xy) = \varphi_*(x)\varphi_*(y)$ for any words $x$ and $y$ from $A^*$, then $\varphi_*(A)$ is an uniquely concatenable set.*

§22 **Proof**: Let $\varphi : A \to B^*$ be an injective map. Assume $\varphi_*(u) = \varphi_*(v)$ for two words $u, v \in A^*$. By the construction of $\varphi_*$ we have $\varphi(u_1) \cdots \varphi(u_l) = \varphi(v_1) \cdots \varphi(v_m)$. Both sides of the equation is made up of concatenations of words from $\varphi(A)$, thus, by definition §17 we have $l = m$ and $\varphi(u_i) = \varphi(v_i)$ for all $i$. Since $\varphi$ is injective, we conclude that $v = u$.
On the other hand, assume $\varphi_* : A^* \to B^*$ is injective and that

$$\varphi_*(xy) = \varphi_*(x)\varphi_*(y), \tag{1}$$

for any words $x$ and $y$. Then $\varphi_{*|A}$ is an injective map and $\varphi_*(A)$ is an uniquely concatenable set. To see this, set $\varphi = \varphi_{*|A}$ and let $\varphi(x_1) \cdots \varphi(x_l) = \varphi(y_1) \cdots \varphi(y_m)$ where $x_i, y_j \in A$. Because $\varphi_*$ respects (1) we have $\varphi_*(x_1^l) = \varphi_*(y_1^m)$ which leads us to the conclusion that $x_1^l = y_1^m$, and thus $\varphi(x_i) = \varphi(y_i)$. This shows that $\varphi_*(A)$ is an uniquely concatenable set.

§23 **Theorem** (McMillan).*Let $V = \{w_1, \ldots, w_k\} \subset B^*$ with $B = \{0, 1\}$ and assume $V$ is uniquely concatenable. The following inequality, which is known as "Kraft's inequality", holds:*

$$\sum_{w \in V} 2^{-|w|} \leq 1. \tag{2}$$

§24 **Proof**: Let $V \subset B^*$ and assume

$$\sum_{w \in V} 2^{-|w|} \geq s, \quad s > 1.$$

For any $m \in \mathbb{N}$ let $V^m$ be the cartesian product. We want to find $w = (w_1, \ldots, w_m) \in V^m$ and $v = (v_1, \ldots, v_m) \in V^m$, such that $w \neq v$ and yet their concatenations are equal, i.e. $w_1 \cdots w_m = v_1 \cdots v_m$. For this task we employ a "chest of drawers" argument. Observe that any concatenation $w_1 \cdots w_m$ of an element $(w_1, \ldots, w_m) \in V^m$ has length no greater than $mL_{max}$, with $L_{max} = \max\{|w| \mid w \in V\}$. Now, for each integer $L$, $1 \leq L \leq L_{max}$ we consider a drawer, labelled

$L$. We now stuff the elements of $V^m$ into the drawers by the following simple rule: An element $(w_1, \ldots, w_m)$ goes into the drawer $|w_1| + \cdots + |w_m|$. Observe that

$$s^m \leq \left( \sum_{w \in V} 2^{-|w|} \right)^m = \sum_{(w_1,\ldots,w_m) \in V^m} 2^{-(|w_1|+\cdots+|w_m|)} = \sum_{L=1}^{mL_{max}} c(L) 2^{-L}.$$

The last equality is just another way of writing the same sum. We sum over the drawers, $1, \ldots, mL_{max}$, instead of the cartesian product $V^m$. The number $c(L)$ is the number of elements in the drawer $L$. If the above sum becomes greater than $mL_{max}$ there is a drawer $L'$ where $c(L') > 2^{L'}$. Since there are only $2^{L'}$ different binary words of length $L'$, there is two different elements $w = (w_1, \ldots, w_m)$ and $v = (v_1, \ldots, v_m)$, both from $V^m$, such that $w_1 \cdots w_m = v_1 \cdots v_m$.

Because $s > 1$ there exists a number $m$ such that indeed $s^m > mL_{max}$, and this completes the proof.

§25 **Theorem** (KRAFT). *Let $P = \{w_1, \ldots, w_k\} \subset B^*$ be a prefix set of words over the binary alphabet $B$. Setting $l_i = |w_i|$, then Kraft's inequality holds,*

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1, \tag{3}$$

*with equality if and only if $P$ is a complete prefix set. Furthermore, for any set $\{l_1, \ldots, l_k\}$ of natural numbers for which (3) holds, there exists a prefix set $P = \{w_1, \ldots, w_k\} \subset B^*$ such that $l_i = |w_i|$ for $1 \leq i \leq k$.*

§26 **Proof (of theorem §25)**: Assume $P$ is a prefix set. As noted in observation §19, a prefix set is also uniquely concatenable, and thus theorem §23 tells us that the Kraft inequality is fulfilled.

Now assume $P$ is a complete prefix set, and let $w' \in P$. By adding the word $w' \cdots w'$ we will definitely get a set which is not uniquely concatenable. Therefore, while $\sum_{w \in P} 2^{-|w|} \leq 1$ by the prefix property of $P$, $\sum_{w \in P} 2^{-|w|} + 2^{-k|w'|} > 1$, by theorem §23. The actual value of $w^{-k|w'|}$ can be chosen arbitrarily small, thus we conclude $\sum_{w \in P} 2^{-|w|} = 1$. If on the other hand we assume $P$ to be a prefix set for which $\sum_{w \in P} 2^{-|w|} = 1$, it is immediately seen that for any word $w' \in A^*$, the set $P \cup \{w'\}$ cannot be a prefix set. This proves the completeness of $P$.

Finally, assume we have a set $L = \{l_1, \ldots, l_k\}$ of natural numbers such that the Kraft inequality holds for these numbers. Without loss of generality we may assume the integers to be ordered, so that $l_1 \leq l_2$ and so forth. If $k = 1$ we may easily construct a prefix set $P$ corresponding to L, namely $P = \{0 \cdots 0\}$, the set of a single word of $l_1$ zeros. Assume $k > 1$ and proceed by induction: For the set $L' = \{l_1, \cdots, l_{k-1}\}$ we may by the induction hypothesis construct a prefix set $P'$ of words with lengths corresponding to $L'$. Furthermore, by the assumption that $L$ fulfills the Kraft inequality, we know

$$\left( \sum_{i=1}^{k-1} 2^{-l_i} \right) + 2^{-l_k} \leq 1$$

Thus the prefix set $P'$ is not complete, so we may still add words and keep the prefix property. We must check, however, if there is a word of length $l_k$ left, such that adding it will not break the prefix property. There are a total of $2^{l_k}$ binary words of the right length. Of these some may

be used already, and some may be unavailable because a prefix of them is in $P'$. Because of the ordering of $L$ we need not worry about picking a prefix of another word. That is, any word in $P'$ of length $l$ makes $2^{l_k - l}$ words of length $l_k$ unavailable. So for us to be able to add a word of length $l_k$ we must see that

$$2^{l_k} > \sum_{i=1}^{k-1} 2^{l_k - l_i},$$

which follows easily from the inequality just above, by dividing by $2^{l_k}$.

§27 **Observation**. The Kraft inequality shows how the possible number of elements in a prefix set is related to the lengths of the elements. It is important to recognize the *number* of words as a parameter as well as the *lengths* of the words. Focusing on these parameters (number of words and their lengths), the second part of Kraft's theorem, §25, shows that for any uniquely concatenable set there is a prefix set with the same number of words and the same length of words. There is a tug of war between the possibility of having a large number of words and the possibility of having short words in a prefix set. Still, as the last part of the theorem tells us, within the restriction of the Kraft inequality we are free to choose the number of words and their lengths.

§28 **Observation**. Combining theorem §23 and theorem §25, we see that for any uniquely concatenable set $U \subset B^*$, there is a prefix free set $V \subset B^*$ with the same number of elements as $U$ and with each element of $V$ the same length as the corresponding one in $U$. For example, to the uniquely concatenable set from example §18, $U = \{00, 100, 10\}$, corresponds, say, the prefix free set $V = \{00, 100, 01\}$. Thus, when studying representation of messages constructed on a per-letter basis as in example §13 we may, whenever the length of the representation is our prime concern, assume $\varphi(A)$ to be a prefix set.

§29 **Remark**. This completes our treatment of the per-symbol representation (c.f. example §13). We now turn to the per-message-length representation as introduced in example §14.

§30 **Theorem** (ELIAS). *There exists an injective map $\mathcal{E} : \mathbb{N} \to B^*$, such that $\mathcal{E}(\mathbb{N})$ is a prefix free set and such that $|\mathcal{E}(n)| = \log n + o(\log n)$ as $n \to \infty$.*

§31 **Proof (sketch)**: The proof is quite easy once the main idea has been presented. Details can be found in e.g. [Shields, 1996] or [Rissanen, 1989]. The construction is best explained by an example: The binary sequence corresponding to a number, say 17, is defined as:

$$\overbrace{000}^{x}\overbrace{101}^{y}\overbrace{10001}^{z}$$

where $z$ is 17 in the usual binary notation (10001) and $y$ is $|z| = 5$ in usual binary notation (101) and $x$ is $|y| = 3$ zeroes (000). Thus $\mathcal{E}(17) = 00010110001$ and $|\mathcal{E}(17)| = 11$. The lengths of the three parts of the codeword for 17 may be calculated as $|z| = \lceil \log 17 + 1 \rceil = 5$ and $|y| = \lceil \log |z| + 1 \rceil = 3$ and $|x| = |y| = 3$. In general the formula for the length of the Elias encoding of $n$ is

$$|\mathcal{E}(n)| = \lceil 1 + \log n \rceil + 2\lceil 1 + \log \lceil 1 + \log n \rceil \rceil$$

from which the theorem follows.

§32 **Example**. In the following table we list the Elias encoding of small integers, as defined by the map $\mathcal{E}$ from above (§31):

| $n$ | $\mathcal{E}(n)$ | $n$ | $\mathcal{E}(n)$ |
|---|---|---|---|
| 1 | 011 | 5 | 0011101 |
| 2 | 001010 | 6 | 0011110 |
| 3 | 001011 | 7 | 0011111 |
| 4 | 0011100 | 8 | 0001001000 |

The map $\mathcal{E}$ is just one possible prefix code of the natural numbers with the property that the codeword length is (essentially) proportional to the logarithm of the size of the number being encoded. See §156 in the Notes and References section for some further remarks on this kind of constructions and related results.

§33 **Corollary**. *From any collection* $\{\varphi_n\}_{n\in\mathbb{N}}$ *of injective maps* $\varphi_n : A^n \to B^*$ *a collection* $\{\varphi'_n\}_{n\in\mathbb{N}}$ *of injective maps* $\varphi'_n : A^n \to B^*$ *can be constructed such that* $\varphi'_n(A^n)$ *is a prefix free set and such that for any* $n$ *and any* $x_1^n \in A^n$

$$|\varphi'_n(x_1^n)| - |\varphi_n(x_1^n)| = \log|\varphi_n(x_1^n)| + o(\log|\varphi_n(x_1^n)|)$$

*as* $n \to \infty$. *Furthermore, the collection* $\{\varphi_n\}_{n\in\mathbb{N}}$ *may be used to create an injective map* $\varphi_* : A^* \to B^*$ *by the construction*

$$\varphi_*(x_1^n) \stackrel{\text{def}}{=\!=} \mathcal{E}(n)\varphi_n(x_1^n).$$

§34 **Proof**: Let $n \in \mathbb{N}$ and define $\varphi'_n(x_1^n) \stackrel{\text{def}}{=\!=} \mathcal{E}(|\varphi_n(x_1^n)|)\varphi_n(x_1^n)$. To see that $\varphi'_n(A^n)$ is a prefix free set, assume $\varphi'_n(x_1^n) = \varphi'_n(y_1^n)w$ for some $x_1^n, y_1^n \in A^n$ and some $w \in B^*$. That is,

$$\mathcal{E}(|\varphi_n(x_1^n)|)\varphi_n(x_1^n) = \mathcal{E}(|\varphi_n(y_1^n)|)\varphi_n(y_1^n)w$$

Since $\mathcal{E}(\mathbb{N})$ is a prefix free set, we conclude that $\mathcal{E}(|\varphi_n(x_1^n)|) = \mathcal{E}(|\varphi_n(y_1^n)|)$ and thus $|\varphi_n(x_1^n)| = |\varphi_n(y_1^n)|$. This implies that $|w| = 0$ and then $\varphi'_n(x_1^n) = \varphi'_n(y_1^n)$ so that $\varphi'_n(A^n)$ is prefix free. Indeed, we have also shown that $\varphi_n(x_1^n) = \varphi_n(y_1^n)$ which, since $\varphi$ is injective, shows that $x_1^n = y_1^n$. Thus $\varphi'_n$ is also injective.
By theorem §30 we easily find

$$|\varphi'_n(x_1^n)| - |\varphi_n(x_1^n)| = |\mathcal{E}(|\varphi_n(x_1^n)|)| = \log|\varphi_n(x_1^n)| + o(\log|\varphi_n(x_1^n)|).$$

as $n \to \infty$. Finally, the map $\varphi_* : A^* \to B^*$ is injective, for assume $\varphi_*(x_1^n) = \varphi_*(y_1^m)$ for some $x_1^n, y_1^m \in A^*$. By the prefix free property of $\mathcal{E}(\mathbb{N})$ we conclude, that $n = m$ and then the injectiveness of $\varphi_n$ yields the desired result.

§35 **Observation**. Although we do not have complete equivalence between representations that are prefix free as in the per-letter case (see observation §28), we do have the following asymptotic result: If $\{\varphi_n\}_{n\in\mathbb{N}}$ is a collection of injective maps from $A^n$ to $B^*$ and we consider the per-symbol length of the representation, $|\varphi_n(x_1^n)|/n$, we may for all asymptotic purposes (where $n \to \infty$) assume the sets $\varphi_n(A^n)$ to be prefix free. This follows easily from corollary §33 an concludes the case of per-message-length representation, c.f. example §14.

§36 **Remark**. We are now ready to state the definition of a code[2]. We have purposefully postponed this moment until now in order to ensure that we have argued thoroughly that the code concept

[2]Other definitions of codes than the one we are about to give are known in the litterature. In [Berstel and Perrin, 1985], for example, the term *code* is used for sets of the type we know as uniquely concatenable.

captures any possibility of creating short representations of messages. In other words, we want to make sure that any problem involving the minimization of code length is in essence the problem of minimizing the length of any representation of the message.

§37 **Definition**. Let $A$ and $B$ be finite alphabets. A *code on* $A$ is an injective map $\varphi : A \to B^*$ such that $\varphi(A)$ is an uniquely concatenable subset of $B^*$. A $n$-*code on* $A$ is a code on $A^n$. A *code sequence* on $A$ is a collection $\{\varphi_n\}_{n\in\mathbb{N}}$, such that $\varphi_n$ is a $n$-code on $A$. A *global code* on $A$ is a code on $A^*$.

§38 **Observation**. It is clear from the definition, that an $n$-code on $A$ is a 1-code on $A^n$. On the other hand, a 1-code on $A^n$ does not imply any 1-code on $A$. Similarly, any global code, $\varphi_*$ contains a code sequence by the construction $\varphi_n \stackrel{\text{def}}{=\!=} \varphi_{*|A^n}$, i.e. $\varphi_n$ is $\varphi_*$'s restriction with respect to $A^n$. Conversely, a code sequence $\{\varphi_n\}$ does not directly imply a global code since in general the construction $\varphi_*(x_1^n) \stackrel{\text{def}}{=\!=} \varphi_n(x_1^n)$ will not produce a uniquely concatenable set of codewords.

§39 **Definition**. If $\varphi_n(A^n)$ is a prefix free set we say that the code is a *prefix code*. If all codes in a code sequence are prefix codes we say that the sequence is a *prefix code sequence*.

§40 **Remark**. The notion of *code* we have encountered so far, closely imitates the popular meaning of a code, namely that of a table of clear text symbols or messages and their corresponding codewords. The corresponding mathematical concept is a special type of mappings, which we have named codes. In the following we will consider how codes on $A^\infty$ may be treated in terms of data compression performance. Codes on infinite sequences does not have any popular meaning; certainly the image of a table of clear and encoded words is meaningless. In particular it is unclear how we should measure the performance (in terms of compression) of codes on $A^\infty$ since all sequences involved are of equal length, namely infinite. These questions leads us to ask the question: Why bother with codes on $A^\infty$ at all? As we will see, however, it is both possible and rewarding to study these generalized codes; possible in the sense that we get a good representation of methods of compression, just as we did in the case of the finite codes, and rewarding in the sense that it is possible to prove a very nice coding theorem for this type of codes. Before we make the actual definition of a code on $A^\infty$, we proceed to develop some intuition about how to create a representation of the elements of $A^\infty$ – just as we did in the finite case.

§41 **Observation**. Let $\mathbf{x} \in A^\infty$ and consider how we might map $\mathbf{x}$ to an element in $B^\infty$. Thanks to the work done so far, we have several options. If $\{\varphi_n\}_{n\in\mathbb{N}}$ is a prefix code sequence on $A$, we could write

$$\varphi_\infty(\mathbf{x}) \stackrel{\text{def}}{=\!=} \varphi_n(x_1^n)\varphi_n(x_{n+1}^{2n})\cdots \tag{4}$$

where $n$ is any positive integer and $\varphi_\infty$ would then be a code on $A^\infty$. In general, for any sequence of positive integers, $n_1, n_2, \cdots$, we may define

$$\varphi_\infty(\mathbf{x}) \stackrel{\text{def}}{=\!=} \varphi_{n_1}(x_1^{n_1})\varphi_{n_2}(x_{n_1+1}^{n_1+n_2})\cdots$$

These constructions enable us to encode any source instance $\mathbf{x} \in A^\infty$. We might even have a setup where the exact values of $n_1, n_2, \ldots$ is dependent on $\mathbf{x}$, e.g.

$$\varphi_\infty(\mathbf{x}) = \begin{cases} \varphi_1(x_1)\varphi_1(x_2)\cdots & \text{if } x_1 \neq x_2 \\ \varphi_2(x_1^2)\varphi_2(x_3^4)\cdots & \text{if } x_1 = x_2 \end{cases}$$

or more complicated schemes. The prefix property of $\{\varphi_n\}$ ensures that the resulting map $\varphi_\infty$ is indeed an injective map from $A^\infty$ to $B^\infty$.

§42 **Observation**. Let $\varphi_\infty : A^\infty \to B^\infty$ be an injective map. We may ask the question: To what extend can we view $\varphi_\infty$ as a *code* on $A^\infty$? The concept of codeword length has no meaning in this case and the property of being uniquely concatenable is hard to define because concatenation in itself is meaningless in $A^\infty$. Yet, as we have just seen, for any prefix code sequence $\{\varphi_n\}_{n\in\mathbb{N}}$ we may pick any sequence $n_1, n_2, \ldots$ with $n_i \in \mathbb{N}$ and define $\varphi_\infty(\mathbf{x}) \stackrel{\text{def}}{=\!=} \varphi_{n_1}(x_1^{n_1})\varphi_n(x_{n_1+1}^{n_1+n_2})\cdots$. Observing the infinite codeword $\varphi_\infty(\mathbf{x})$ will now enable us to decode the original sequence $\mathbf{x}$, in blocks of length $n_i$, each time utilizing a subsequence of the infinite codeword of length $|\varphi_{n_i}(x_{n_1+\cdots+n_{i-1}}^{n_1+\cdots+n_i})|$. Thus we use the codeword lengths of the underlying *finite* code when evaluating how well $\varphi_\infty$ encode the message. The insight that will give us a generalized version of this idea is that we do not need to explicitly state the underlying finite code. This generalized concept of code length is defined below. It is known as the *cost* function and while the concept may seem a bit tricky at first glance we provide a number of examples and comments in the subsequent paragraphs showing that the definition is indeed reasonable.

§43 **Definition**. Let $A$ and $B$ be finite alphabets. Let $\varphi_\infty : A^\infty \to B^\infty$ be an injective map. We say that $\varphi_\infty$ is a *code* on $A^\infty$. For any $\mathbf{x} \in A^\infty$ and any $n \in \mathbb{N}$ let

$$c(\varphi_\infty, \mathbf{x}, n) \stackrel{\text{def}}{=\!=} \min\left\{k \in \mathbb{N} \mid \varphi_\infty^{-1}\left(\varphi_\infty(\mathbf{x})_1^k B^\infty\right) \subseteq x_1^n A^\infty\right\} \tag{5}$$

with $\min\{\} \stackrel{\text{def}}{=\!=} \infty$. The notation $x_1^n A^\infty$ is short for the set $\{\mathbf{y} \in A^\infty \mid y_1^n = x_1^n\}$ and $\varphi_\infty(\mathbf{x})_1^k$ means the first $k$ symbols of the infinite codeword $\varphi_\infty(\mathbf{x})$. Define

$$c(\varphi_\infty, \mathbf{x}) \stackrel{\text{def}}{=\!=} \liminf_{n\to\infty} \frac{c(\varphi_\infty, \mathbf{x}, n)}{n}, \tag{6}$$

a quantity known as the *cost of* $\varphi_\infty$ *on* $\mathbf{x}$. For a combinatorial source $S \subseteq A^\infty$, we define the *cost of* $\varphi_\infty$ *on* $S$ to be

$$c(\varphi_\infty, S) \stackrel{\text{def}}{=\!=} \sup_{\mathbf{x}\in S} c(\varphi_\infty, \mathbf{x}). \tag{7}$$

§44 **Observation**. We may say that $c(\varphi_\infty, \mathbf{x}, n)$ is the minimum number of letters of $\varphi_\infty(\mathbf{x})$ needed in order to uniquely decode the first $n$ letters of the sequence $\mathbf{x}$. The value of $c(\varphi_\infty, \mathbf{x}, n)$ is non-decreasing in $n$. Informally this means that when decoding more of the source message we need to look at more of the codeword, a proposition which seems obvious when considered in the light of the informal definition of $c$ given at the start of this paragraph. To check this more formally, we observe that if $k$ code symbols allows us to decode $n$ source symbols, that is, $\varphi_\infty^{-1}(\varphi_\infty(\mathbf{x})_1^k B^\infty) \subseteq x_1^n A^\infty$ then $k$ symbols will also allow us to decode $m < n$ symbols, because $x_1^n A^\infty \subseteq x_1^m A^\infty$. Hence we conclude by (5) that $c(\varphi_\infty, \mathbf{x}, n) \geq c(\varphi_\infty, \mathbf{x}, m)$ for $n > m$.
It is not always possible to decode the first $n$ letters of $\mathbf{x}$ by looking at a finite part of $\varphi_\infty(\mathbf{x})$, and in this case the value of $c(\varphi_\infty, \mathbf{x}, n)$ is positive infinity. The next example illustrates this.

§45 **Example**. Let $A = \{0, 1, 2\}$, let $C = \{0, 2\}^\infty$ and define the map $\varphi_\infty : A^\infty \to A^\infty$ as follows:

$$\varphi_\infty(\mathbf{x}) \stackrel{\text{def}}{=\!=} \begin{cases} \mathbf{x} & \text{if } \mathbf{x} \in C \\ \overline{x}_1\overline{x}_2\cdots & \text{if } \mathbf{x} \notin C \end{cases}$$

where $\overline{0} = 2$, $\overline{2} = 0$ and $\overline{1} = 1$. It is clear, that $\varphi_\infty$ is injective. On the other hand, for any $x \in C$ and any $k \in \mathbb{N}$ we find that the set $\varphi_\infty^{-1}(\varphi_\infty(x)_1^k A^\infty)$ in addition to $x_1^k A^\infty$ contains also $\overline{x}_1\overline{x}_2\cdots\overline{x}_k A^\infty$. We conclude that $c(\varphi_\infty, x, n) = \infty$ for any $x \in A^\infty$ and any $n \in \mathbb{N}$. Thus $\varphi_\infty$ is indeed a code on $A^\infty$, but we need to look at *all* the symbols in the codeword to determine the encoded message. From a practical point of view codes on $A^\infty$ for which $c(\varphi_\infty, x, n) = \infty$ is therefore not interesting. As the reader might have noticed, the set $C$ is equivalent[3] to the Cantor set. The fact that no finite prefix of $\varphi_\infty(x)$ allows for decoding of just a single symbol of $x$ is related to the fact that the Cantor set contains no subinterval however small. The map $\varphi_\infty$ is the identity on elements of the Cantor set, and it mirrors all other elements through the point $\frac{1}{2}$. Hence it is a bijection of the unit interval onto itself.

§46 **Example**. Let $A = B = \{0, 1\}$ and consider encoding the sequence $x = 0101010101\cdots$. Let $\varphi_n : A^n \to B^*$ be defined for any even, positive integer $n$ such that

$$\varphi_n(a_1^n) = \begin{cases} 1\underbrace{0\cdots0}_{n/2}1b_1b_2 & \text{if } a_1^n = \underbrace{b_1b_2b_1b_2\cdots}_{n} \\ 1\mathcal{E}(n)a_1^n & \text{otherwise} \end{cases}$$

Here $\mathcal{E}$ is the Elias prefix code of the integers known from theorem §30. Define $\varphi_\infty : A^\infty \to B^\infty$ in the following manner: $\varphi_\infty(x) = \varphi_2(x_1^2)\varphi_4(x_3^6)\varphi_8(x_7^{14})\cdots$. We observe that the code sequence $\varphi_\infty(x)$ equals $10101, 100101, 10000101, 100\cdots$, where the codewords originating from the different $\varphi_i$'s have been indicated by commas so that $\varphi_\infty(x)$ is made up of from a sequence of blocks. In decoding we note that the whole block is needed, that is, $c(\varphi_\infty, x, n) = 5$ for $n = 1, 2$; $c(\varphi_\infty, x, n) = 11$ for $n = 3, \ldots, 6$; $c(\varphi_\infty, x, n) = 19$ for $n = 7, \ldots, 14$ and so on. In general we have $c(\varphi_\infty, x, n) = 4i + 2^i - 1$ for $2^i - 2 < n \leq 2^{i+1} - 2$, where $i$ is the number of blocks. For $n_i = 2^i - 1$ we find $\lim_{i\to\infty} c(\varphi_\infty, x, n_i)/n_i = \lim_{i\to\infty}(4i + 2^i - 1)/(2^i - 1) = 1$. On the other hand, taking $m_i = 2^{i+1} - 2$ yields $\lim_{i\to\infty} c(\varphi_\infty, x, m_i)/m_i = \lim_{i\to\infty}(4i + 2^i - 1)/(2^{i+1} - 2) = 1/2$. This example shows the consequence of using limes inferior in the definition of $c(\varphi_\infty, x)$, and that the definition is in agreement with our intuitive understanding that $\varphi_\infty$ is indeed capable of encoding $x$ into about half the number of bits, although using a somewhat clumsy partition into blocks of explosive size.

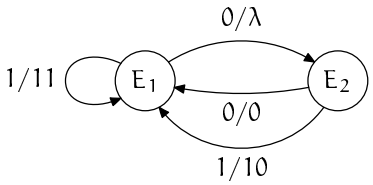
§47 **Example**. The last example is also the most realistic example in the sense that it is close to what might be found in real applications. Is is also an example where we see the strength of the concept of codes on $A^\infty$; no block code can capture the encoding method presented below. For this example we need to introduce the concept of a finite state encoder. We do so only briefly; one may consult e.g. [Ziv and Lempel, 1978] for details. Let $E = (S, A, B, g, f)$ be a *finite state encoder*, where $S$ is the finite set of states of which one state $s_0 \in S$ is defined to be the starting state, $A$ is the input alphabet, $B$ is the output alphabet, $g : S \times A \to S$ is the state transition map and $f : S \times A \to B^* \cup \{\lambda\}$ is the output map. Thus, when feeding $E$ with an infinite sequence of letters $x = x_1x_2\cdots$ from $A$, it produces an infinite sequence of words $y = y_1y_2\cdots$ beginning with $y_1 = f(s_0, x_1)$ and with each $y_i$ from $B^* \cup \{\lambda\}$. The encoder $E$ is said to be ILF (*Information Lossless of Finite order*) if there exist a number $m \in \mathbb{N}$ such that for any state $s \in S$ and any

[3] The Cantor set $C$ may be obtained by an iterative process of removing the 'middle thirds' from $[0, 1]$. This results in a sequence of sets beginning with $[0, 1]$, then $[0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$, then $[0, \frac{1}{9}] \cup [\frac{2}{9}, \frac{3}{9}] \cup [\frac{6}{9}, \frac{7}{9}] \cup [\frac{8}{9}, 1]$, and so forth. The intersection between all these sets is the Cantor set. Removing the middle third from $[0, 1]$ corresponds to removing the numbers whose 3-ary expansion starts with the digit 1. The next removal corresponds to the obliteration of numbers having 1 as digit number two, and so on.

sequence $x_1^m \in A^m$, feeding $x_1^m$ into the encoder starting from state $s$, produces an output, $y_1^m$, from which it is possible to uniquely determine the first letter of the message, $x_1$. As an example, consider the finite state encoder $E = (S, A, B, g, f)$ defined by

$$\begin{array}{lcl} S & = & \{E_1, E_2\} \\ A & = & \{0,1\} \\ B & = & \{0,1\} \end{array} \qquad \begin{array}{lcllcl} g(E_1,0) & = & E_2 & f(E_1,0) & = & \lambda \\ g(E_1,1) & = & E_1 & f(E_1,1) & = & 11 \\ g(E_2,0) & = & E_1 & f(E_2,0) & = & 0 \\ g(E_2,1) & = & E_1 & f(E_2,1) & = & 10 \end{array}$$

with $E_1$ the appointed starting state. It is somewhat more convenient to work with a graphical representation like this,




where the interpretation of labels is as follows: The label $0/\lambda$ going from state $E_1$ to $E_2$ should be read: "When the current state is $E_1$ and the next symbol to be encoded is 0, the output is $\lambda$ and the next state is $E_2$." In terms of the maps defining the encoder, this corresponds to $f(E_1, 0) = \lambda$ and $g(E_1, 0) = E_2$. One may easily check that $E$ is ILF, since for any input of length 1, the output will be either 11 or $\lambda$ when starting in state $E_1$, and either 10 or 0 when starting in state $E_2$. As described above, $E$ defines a map $\varphi_E : A^\infty \to B^\infty$. Thus if $\mathbf{x} = 0001\cdots$, the state transitions would be from $E_1$ to $E_2$ along the path labeled $0/\lambda$, then back to $E_1$ along the path labeled $0/0$, and so forth. Through this procedure the codeword for $\mathbf{x}$ will emerge as $\varphi_\infty(\mathbf{x}) = 010\cdots$. It is interesting to note, that $\varphi_\infty(000\cdots) = 000\cdots$ and $\varphi_\infty(111\cdots) = 111\cdots$. However, the cost function captures the performance of $\varphi_\infty$ in terms of compression; we have $c(\varphi_\infty, 000\cdots) = 1/2$ and $c(\varphi_\infty, 111\cdots) = 2$.

§48 **Observation**. As the preceding examples have shown, codes on $A^\infty$ can not always be thought of as being built from finite block codes like we did in observation §41, (4). The blocks, in which it is possible to decode the message, is not of fixed length, but vary according to the particular message $\mathbf{x}$ in question. However, under the reasonable requirement that $c(\varphi_\infty, \mathbf{x}) < \infty$ we find that $\varphi_\infty$ exhibit a sort of generalized blocking behavior known as a *parsing*. Indeed, if $c(\varphi_\infty, \mathbf{x}) = \alpha$ we have by the definition of the cost (see definition §43(6)) that $\liminf_{n\to\infty} c(\varphi_\infty, \mathbf{x}, n)/n = \alpha$ and hence there is an sequence of increasing natural numbers, $n_1, n_2, n_3, \ldots$, for which $\lim_{i\to\infty} c(\varphi_\infty, \mathbf{x}, n_i)/n_i = \alpha$. This sequence defines a *parsing*, that is, a division of $\mathbf{x}$ into finite subsequences, $x_1^{n_1}$, $x_{n_1+1}^{n_2}$, .... Let $\mathbf{y} = \varphi_\infty(\mathbf{x})$ and let $c_i \stackrel{\text{def}}{=\!=} c(\varphi_\infty, \mathbf{x}, n_i)$. We may then think of $\varphi_\infty$ as an encoder reading a finite piece of the message (a phrase), $x_1^{n_1}$, and then writing the codeword for the phrase, namely $y_1^{c_1}$. In general the encoding of the $i$'th phrase $x_{n_{i-1}+1}^{n_i}$ is $y_{c_{i-1}+1}^{c_i}$. Summing up our findings, we may say that while a code $\varphi_\infty$ on $A^\infty$ with $c(\varphi_\infty, \mathbf{x}) < \infty$ is not necessarily based on a parsing (the definition only call for an injective map), it must define one.

§49 **Observation** (SEQUENTIAL CODES). Informally a sequential code is a code $\varphi_\infty : A^\infty \to B^\infty$ which allows an increasing prefix of the message to be decoded by observing an increasing prefix of the codeword. Thus, using the terms from above, a sequential code is a code $\varphi_\infty$ on $A^\infty$ for which $c(\varphi_\infty, \mathbf{x}) < \infty$ for all $\mathbf{x} \in A^\infty$. This implies that for each $\mathbf{x} \in A^\infty$, the code will

define a sequence $n_1, n_2, \ldots$ of increasing positive integers and hence a parsing of $x$. The parsing need not be independent of $x$, see example §47, and the lengths of the individual phrases of the parsing may vary wildly, see example §46 where the phrase length grows explosively. Often we will not be satisfied by $\varphi_\infty$ just being a sequential code. The difference between successive $n_i$'s is of interest too. From a practical point of view this difference is of particular importance since if $n_{i+1} - n_i$ is large, the result is a large delay in the decoding.

§50 **Definition**. Let $\varphi_\infty : A^\infty \to B^\infty$ be a code on $A^\infty$. The code is said to be *sequential* if $c(\varphi_\infty, x) < \infty$ for all $x \in A^\infty$. A sequential code is said to have *limited delay* if for any $x \in A^\infty$ there are positive integers $n_1, n_2, \ldots$ and a number $\alpha$ such that $\lim_{i\to\infty} c(\varphi_\infty, x, n_i)/n_i = \alpha$ and such that

$$\lim_{i\to\infty} \frac{n_{i+1} - n_i}{n_i} = 0. \tag{8}$$

§51 **Lemma** (SEQUENTIAL CODE DELAY LEMMA). *Let $\varphi_\infty : A^\infty \to B^\infty$ be a sequential code on $A^\infty$ with limited delay. Then*

$$\liminf_{n\to\infty} \frac{c(\varphi_\infty, x, n)}{n} = \limsup_{n\to\infty} \frac{c(\varphi_\infty, x, n)}{n}.$$

§52 **Proof**: According to definition §50 let $n_1, n_2, \ldots$ be a sequence of positive integers and $\alpha$ be a number so that $\lim_{i\to\infty}(n_{i+1} - n_i)/n_i = 0$. Note that $\lim_{i\to\infty} n_{i+1}/n_i = 1$. By definition (see also observation §44) we have $c(\varphi_\infty, x, n) \le c(\varphi_\infty, x, n+1)$ and since for any $n$ there is a number $i$ so that $n_i \le n \le n_{i+1}$ it follows that $c(\varphi_\infty, x, n_i) \le c(\varphi_\infty, x, n) \le c(\varphi_\infty, x, n_{i+1})$. As a consequence

$$\frac{c(\varphi_\infty, x, n_i)}{n_i} \le \frac{c(\varphi_\infty, x, n)}{n_i} \le \frac{c(\varphi_\infty, x, n_{i+1})}{n_i} = \frac{c(\varphi_\infty, x, n_{i+1})}{n_{i+1}} \cdot \frac{n_{i+1}}{n_i}$$

for any $n$ and a suitable value of $i = i(n)$. It then follows that $\lim_{n\to\infty} c(\varphi_\infty, x, n)/n_i = \alpha$. Since $1 \le n/n_i \le n_{i+1}/n_i$ and $\lim_{i\to\infty} n_{i+1}/n_i = 1$ it is easily seen that $\lim_{n\to\infty} c(\varphi_\infty, x, n)/n = \alpha$.

§53 **Observation**. The requirement of definition §50(8), is a reasonable way of ensuring limited decoding delay. The sequential code delay lemma shows, that under the requirement of limited decoding delay, the limes inferior in the definition of the cost function (see definition §43(6)) is in fact a true limit. Example §46 illustrates the case where the delay of the code is too large for the lemma to hold; and indeed, as explained in the example, this is why the limes inferior is needed in the definition of $c(\varphi_\infty, x)$. In addition, we note that the fraction in the limit definition §50(8) is closely related to the *delay modules* of a parsing rule as introduced in [Kieffer and Yang, 1996]. Thus the class of codes which is studied there qualify as sequential codes with limited delay, using our terminology.

§54 **Observation**. We conclude this section by relating the cost of codes on $A^\infty$ with the finite state compression ratio, $\rho_E$, as introduced in e.g. [Ziv and Lempel, 1978]. Recall from example §47 that a finite state encoder $E = (S, A, B, g, f)$ defines the codeword of $x \in A^\infty$ by at each step taking as input a symbol from $x$ and outputting a sequence of elements $y_1 y_2 \cdots$ each of which is from $B^* \cup \{\lambda\}$ and such that $y = y_1 y_2 \cdots$ is the (infinite) codeword for $x$. The finite state compression ratio of $x_1^n$ when using the ILF finite state encoder $E$ is defined as

$$\rho_E(x_1^n) \stackrel{\text{def}}{=\!=} \frac{|y_1 \cdots y_n|}{|x_1 \cdots x_n|} = \frac{|y_1 \cdots y_n|}{n}.$$

Note that while each $x_i$ is a symbol from $A$, the output comes in (possibly empty) sequences, $y_i$, from $B^* \cup \{\lambda\}$. Sometimes the compression ratio is normalized by dividing by $\log|A|$. We have chosen not to do so here, in order to make the comparison to the cost more straightforward.

§55 **Lemma**. *Let $\mathbf{x} \in A^\infty$ and assume $E = (S, A, B, g, f)$ is an ILF finite state encoder such that $\limsup_{n\to\infty} \rho_E(x_1^n) = \alpha$ for a number $\alpha$. Let $\varphi_\infty$ be the code on $A^\infty$ defined by $E$. Then $c(\varphi_\infty, \mathbf{x}) \leq \alpha$.*

§56 **Proof**: We have $\varphi_\infty(\mathbf{x}) = y_1 y_2 \cdots$, where $y_i \in B^* \cup \{\lambda\}$, that is, $y_i$ is the (possible empty) word which $E$ produces upon reading $x_i$. Let $m$ be the delay-bound which follows from the ILF requirement and let $w \stackrel{\text{def}}{=\!=} \max\{|f(z,a)| : z \in S, a \in A\}$, that is, $w$ is the maximal length of the words that may be emitted from $E$ so that $|y_i| \leq w$ for all $i$. We observe, that

$$\frac{|y_1 \cdots y_n|}{n} \leq \frac{c(\varphi_\infty, \mathbf{x}, n)}{n} \leq \frac{|y_1 \cdots y_{m+n}|}{n} \leq \frac{|y_1 \cdots y_n| + mw}{n}$$

and taking the upper limit we find

$$\alpha = \limsup_{n\to\infty} \frac{|y_1 \cdots y_n|}{n} \leq \limsup_{n\to\infty} \frac{c(\varphi_\infty, \mathbf{x}, n)}{n} \leq \limsup_{n\to\infty} \frac{|y_1 \cdots y_n| + mw}{n} = \alpha.$$

which completes the proof.

§57 **Lemma**. *Let $\varphi_m$ be a prefix code on $A^m$ and let $S \subseteq A^\infty$ be a combinatorial source for which there is a number $\alpha$ such that*

$$\limsup_{n\to\infty} \frac{\sum_{i=0}^{n-1} |\varphi_m(x_{mi+1}^{mi+m})|}{nm} \leq \alpha$$

*for all $\mathbf{x} \in S$. Then there is a code $\varphi_\infty$ on $S$ such that $c(\varphi_\infty, S) \leq \alpha$.*

§58 **Proof**: Let $\varphi_\infty(\mathbf{x}) = \varphi_m(x_1^m)\varphi_m(x_{m+1}^{2m}) \cdots$. Then clearly,

$$c(\varphi_\infty, \mathbf{x}, nm) \leq \sum_{i=0}^{n-1} |\varphi_m(x_{mi+1}^{mi+m})|$$

for all $\mathbf{x} \in S$. We then have

$$\liminf_{n\to\infty} \frac{c(\varphi_\infty, \mathbf{x}, n)}{n} \leq \liminf_{n\to\infty} \frac{c(\varphi_\infty, \mathbf{x}, nm)}{nm} \leq \liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} |\varphi_m(x_{mi+1}^{mi+m})|}{nm} \leq \alpha$$

for all $\mathbf{x} \in S$. This concludes the proof.

§59 **Observation**. The two preceding lemmas show that codes on $A^\infty$ and their cost as defined by the function $c$, is related to other code concepts. From lemma §57 we see that our class of codes on $A^\infty$ in fact contains the codes made by the well known blocking technique. And from lemma §55 we conclude that the class of codes on $A^\infty$ contains the codes made by finite state encoders. On the other hand, even if $c(\varphi_\infty, S) = \alpha < 1$ where $S$ is some combinatorial source, we cannot conclude from the above results the existence of a block code $\varphi_m : A^m \to B^*$ so that we may may define $\varphi_\infty$ by writing $\varphi_\infty(\mathbf{x}) = \varphi_m(x_1^m)\varphi_m(x_{m+1}^{2m}) \cdots$ for all $\mathbf{x} \in S$. Nor can we guarantee the existence of a finite state encoder with compression ratio tending to $\alpha$. Thus, in a

sense the code concept for infinite sequences is more liberal than the traditional codes we have encountered so far. This should not be a complete surprise; after all we have seen some pretty wild codes like example §45, that definitely is fundamentally different from both blocking and finite state codes.

§60 **Remark**. The connection between codes on $A^\infty$ and the finite forms of codes has not been extensively treated here. There are probably a number of easily provable lemmas similar to the two given above, providing links between finite codes (for example global codes) and codes on $A^\infty$. It would be interesting to explore this interplay further.

§61 **Summary**. We have considered a number of different approaches to information representation leading up to the definition of codes. We have argued, that for the purpose of using codes to transform information, say, the message $x_1^n$, the 1-code, $\varphi$ needs the uniquely concatenable property to be useful[4]. The requirement of unique concatenation was shown to enable us to assume the prefix condition on the code, yielding in the process the Kraft inequality. Using $\varphi_n$ (that is, using a code tailored to the length of our message) is only interesting if we have at our disposal a whole code sequence, in which case we have seen nothing is gained in terms of code length when compared with prefix code sequences. We also noted, that any global code induce trivially a code sequence by the construction $\varphi_n(x_1^n) \stackrel{\text{def}}{=\!=} \varphi_*(x_1^n)$, and the previous argument can be applied to show that for all asymptotic purposes we may assume this code sequence to be prefix. Finally, we considered codes on $A^\infty$. Although a bit of work was needed in order to bring about a reasonable generalization of code length, we found this class of codes to contain many other known codes. It is important to note that even though a code on $A^\infty$ is strictly speaking just a injective map, the concept of codes on $A^\infty$ should be seen together with the cost mechanism of the $c$ function. Without the cost terms like compression as well as comparison with other codes would make no sense.

## The probabilistic source

§62 **Definition**. Let $x \in A^\infty$ be a source instance and let $m \in \mathbb{N}$ be a number. The *relative probability* is defined for each $a_1^m \in A^m$ and each $n \geq m$ as

$$p_m(a_1^m|x_1^n) \stackrel{\text{def}}{=\!=} \frac{\left|\{i \in [1, n-m+1] \mid x_i^{i+m-1} = a_1^m\}\right|}{n-m+1}. \tag{9}$$

The *limiting relative probability* is defined as

$$p_m(a_1^m|x) \stackrel{\text{def}}{=\!=} \lim_{n\to\infty} p_m(a_1^m|x_1^n),$$

provided the limit exists. When $B_m \subseteq A^m$ we write $p_m(B_m|x_1^n)$ and $p_m(B_m|x)$ for the sums $\sum_{b\in B_m} p_m(b|x_1^n)$ and $\sum_{b\in B_m} p_m(b|x)$ respectively.

§63 **Definition**. A source instance $x \in A^\infty$ is said to be *stochastic* if $p_m(a_1^m|x)$ exists for any $m \in \mathbb{N}$ and any $a_1^m \in A^m$. Let $\mathcal{S} \subset A^\infty$ denote the set of all stochastic source instances.

§64 **Observation**. Let $x$ be a stochastic source instance. As seen directly from the definition (9),

[4] This goes for $\varphi_m$ if $m < n$ as well. The use of a code $\varphi_m$ to encode $x_1^n$ is known as the *blocking* approach. We have chosen not to deal with that approach because it can be handled by an alphabet extension, i.e. by defining $A' \stackrel{\text{def}}{=\!=} A^m$ and using $A'$ as source alphabet.

$0 \leq p_1(a|x_1^n) \leq 1$ for any $a \in A$ and any $n \in \mathbb{N}$. The following three calculations reveal further properties of the relative probability. Firstly, we find

$$\sum_{a \in A} p_1(a|x_1^n) = \frac{\left|\{i \in [1,n] \mid x_i \in A\}\right|}{n} = 1 \tag{10}$$

for any $n \in \mathbb{N}$. Secondly, another calculation yields for $b_1^m \in A^m$ with $m < n$ and $n \in \mathbb{N}$,

$$\begin{aligned} \sum_{a \in A} p_{m+1}(b_1^m a|x_1^n) &= \frac{\left|\{i \in [1,n-m] \mid x_i^{i+m} \in b_1^m A\}\right|}{n-m} \\ &= \frac{\left|\{i \in [1,n-m] \mid x_i^{i+m-1} = b_1^m\}\right|}{n-m} \\ &= p_m(b_1^m|x_1^n)\frac{n-m+1}{n-m} - \frac{z}{n-m} \end{aligned} \tag{11}$$

where $b_1^m A$ is short for $\{a_1^{m+1} \in A^{m+1} \mid a_1^m = b_1^m\}$ and $z = 1$ if $x_{n-m+1}^n = b_1^m$ and $z = 0$ otherwise. Thirdly, a similar argument shows that for $b_1^m \in A^m$ with $m < n$ and $n \in \mathbb{N}$

$$\begin{aligned} \sum_{a \in A} p_{m+1}(a b_1^m|x_1^n) &= \frac{\left|\{i \in [2,n-m] \mid x_i^{i+m-1} = b_1^m\}\right|}{n-m} \\ &= p_m(b_1^m|x_1^n)\frac{n-m+1}{n-m} - \frac{w}{n-m} \end{aligned} \tag{12}$$

where $w = 1$ if $x_1^m = b_1^m$ and $w = 0$ otherwise.

Because $\mathbf{x}$ is stochastic all limiting relative probabilities exists. Taking the limit with respect to $m$ on all three results above (10-12), and writing $p_m(\cdot)$ for $p_m(\cdot\,|\mathbf{x})$, we find the following: Any stochastic source instance $\mathbf{x}$ induces a family $\{p_m\}_{m \in \mathbb{N}}$ of functions $p_m : A^m \to [0,1]$ for which the following statements hold:

$$\sum_{a \in A} p_1(a) = 1 \tag{13}$$

$$\sum_{a \in A} p_{m+1}(b_1^m a) = p_m(b_1^m) \tag{14}$$

These two properties are also known as the *kolmogorov consistence axioms* or simply the *consistence conditions*. Furthermore, the following equality, which is known as the *stationarity condition* holds:

$$\sum_{a \in A} p_{m+1}(a b_1^m) = p_m(b_1^m). \tag{15}$$

§65 **Remark**. We are now ready to introduce the measure-theoretic framework upon which much information theory rests, namely the stationary ergodic measure. Often these measures are also known as *stationary ergodic sources*. The following definitions and theorem should thus be seen in the light of the preceding results about limiting relative probabilities: We believe it is reasonable to think of the concept of stationary ergodic sources as something which fit the concept of stochastic source instances, rather than viewing the measure theoretical machinery as something of inherent truth or relevance to our purpose.

§66 **Definition**. Let $\pi_n : A^\infty \to A^n$ be defined by $\pi_n(\mathbf{x}) \stackrel{\text{def}}{=\!=} x_1^n$. The map $\pi_n$ is known as the *projection* of $\mathbf{x}$ on $A^n$. Sets of the form $\pi_n^{-1}(a_1^n)$ for some $a_1^n \in A^n$ and some $n$ are known as *cylinder sets*. Let $\mathcal{F}$ be the $\sigma$-algebra generated from the cylinder sets of $A^\infty$ and let $\mu : \mathcal{F} \to [0,1]$ be a countably additive, nonnegative measure with total mass 1. The triple $(A^\infty, \mathcal{F}, \mu)$ is then known as a *probability space*. Alternatively we say that $\mu$ is a *probability measure* on $A^\infty$. Let $T$ be the left-shift on $A^\infty$ which is measurable because $T^{-1}(\pi_n^{-1}(b_1^n)) = \cup_{a\in A}\pi_{n+1}^{-1}(ab_1^n)$, then if $\mu(T^{-1}B) = \mu(B)$ for all $B \in \mathcal{F}$, we say that $\mu$ is *stationary*. If any $T$-invariant set $B$, that is, any set such that $T^{-1}B = B$, has trivial measure in the sense that either $\mu(B) = 1$ or $\mu(B) = 0$, we say that $\mu$ is *ergodic*.

§67 **Remark**. For brevity we will, when $\mu$ is a stationary measure, identify the cylinder $\pi_n^{-1}(a_1^n)$ with $a_1^n$ allowing us to write $\mu(a_1^n)$ for $\mu(\pi_n^{-1}(a_1^n))$. The motivation for this is the following observation which captures the essence of what the stationary property means. Consider sets of the form $C = \{\mathbf{x} \in A^\infty \mid x_i^{i+n-1} = a_1^n\}$ for some $a_1^n \in A^n$ and some $i \in \mathbb{N}$. We have

$$C = \bigcup_{b_1^{i-1}\in A^{i-1}} \pi_{i+n-1}^{-1}(b_1^{i-1}a_1^n) = T^{-i}\pi_n^{-1}(a_1^n)$$

and hence $\mu(C) = \mu(\pi_n^{-1}(a_1^n))$.

§68 **Definition**. Let $\mathcal{M}_s(A^\infty)$ denote the set of all stationary probability measures on $A^\infty$. Let $\mathcal{M}_e(A^\infty) \subset \mathcal{M}_s(A^\infty)$ be the set of all stationary and ergodic probability measures. The alternative terms *stationary source* and *ergodic source* are sometimes used for elements of these sets.

§69 **Theorem** (KOLMOGOROV'S EXISTENCE THEOREM). *Let $\{p_n\}_{n\in\mathbb{N}}$ be a family of functions $p_n : A^n \to [0,1]$ satisfying the consistence conditions, observation §64, (13) and (14), then there exists a unique probability measure $\mu$ on $A^\infty$ such that*

$$p_n(a_1^n) = \mu(a_1^n).$$

§70 **Proof**: For a detailed and self-contained proof see [Calude, 1994].

§71 **Observation**. In this observation we show how (using Kolmogorov's existence theorem) that a stationary measure may be constructed from a consistent family of functions. Let the family $\{p_n\}_{n\in\mathbb{N}}$ of functions $p_n : A^n \to [0,1]$ be defined by $p_n(b_1^n) = |A|^{-n}$ for $b_1^n \in A^n$. It is easy to check that the consistence conditions observation §64, (13) and (14) holds. By Kolmogorov's existence theorem we then have a probability measure $\mu$ on $(A^\infty, \mathcal{F})$ such that the measure of a cylinder $\mu(\pi_n^{-1}(b_1^n))$ equals $|A|^{-n}$. In order to check that $\mu$ is stationary, we verify that the stationarity condition (15) hold. This is easily done since $\sum_{a\in A} p_{n+1}(ab_1^n) = |A||A|^{-n-1} = |A|^{-n} = p_n(b_1^n)$. Now let $C = \pi_n^{-1}(b_1^n)$ be a cylinder and observe, that $T^{-1}C = \cup_{a\in A}\pi_{n+1}^{-1}(ab_1^n)$, where the union clearly is disjoint. Since $\mu$ is a probability measure we have $\mu(T^{-1}C) = \sum_{a\in A}|A|^{n+1} = |A|^n$, where the last equality follows from the stationarity condition (15). We conclude that $\mu$ is invariant under $T$ for all cylinder sets. The cylinder sets span all measurable sets, and by a standard result in probability theory (see e.g. [Jacobsen, 1995, Theorem 4.2]) we then find that $\mu$ is invariant not only on the cylinder sets but on all measurable sets; hence it is stationary. Later (in example §118) we will check that $\mu$ is also ergodic.

§72 **Observation**. For any $x \in \mathcal{S}$ we get a family of functions $\{p_n\}_{n\in\mathbb{N}}$ satisfying the consistence conditions (see observation §64, (13) and (14)). The Kolmogorov theorem then provides us with a probability measure which we will denote $\mu_x$. The stationarity condition (see observation §64(15)) ensures that the measure is stationary. In the light of this it is natural to consider a partition of $\mathcal{S}$ into classes of elements for which the induced probability measures are equal. We will return to this idea shortly; first, however, we present a well known and central result of ergodicity.

§73 **Theorem** (ERGODIC THEOREM). *Let $\mu$ be a stationary probability measure on $(A^\infty, \mathcal{F})$. If $f : A^\infty \to \mathbb{R}$ is integrable, then the average $(1/n)\sum_{i=1}^n f(T^{i-1}x)$ converges almost surely and in $L^1$ to an integrable, invariant (with respect to $T$) function $\tilde{f} : A^\infty \to \mathbb{R}$, that is,*

$$\mu\left\{x \in A^\infty \mid \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n f(T^{i-1}x) = \tilde{f}(x)\right\} = 1 \tag{16}$$

*and*

$$\lim_{n\to\infty} \int_{A^\infty} \left| \frac{1}{n}\sum_{i=1}^n f(T^{i-1}x) - \tilde{f}(x)\right| d\mu(x) = 0.$$

*Furthermore, if $\mu$ is ergodic then $\tilde{f}(x) = E_\mu f(x)$ almost surely.*

§74 **Proof (partial)**: The theorem, which is also known as *Birkhoff's pointwise ergodic theorem*, is traditionally proved using the maximal ergodic theorem. For two proofs in this vein, see [Billingsley, 1965, Chapter 1, section 2]. It is also possible to prove the theorem using packing and counting techniques, see [Shields, 1996, Section I.3]. Although we do not give a proof here, we will check that the set of sequences from (16) is indeed measurable. To see this we note that any set of the form

$$B(n,\epsilon) = \left\{x \in A^\infty \mid \frac{1}{n}\sum_{i=1}^n f(T^{i-1}x) - \tilde{f}(x) < \epsilon\right\},$$

for $n \in \mathbb{N}$ and $\epsilon > 0$, is measurable since both $f$ and $\tilde{f}$ are measurable functions from $(A^\infty, \mathcal{F})$ to $\mathbb{R}$. We now argue that

$$\left\{x \in A^\infty \mid \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n f(T^{i-1}x) = \tilde{f}(x)\right\} = \bigcap_{N=1}^\infty \bigcup_{M=1}^\infty \bigcap_{n\geq M} B(n, 1/N). \tag{17}$$

We immediately note that if (17) holds, then the left hand side is a measurable set because $B(n,1/M) \in \mathcal{F}$ and $\mathcal{F}$ is a $\sigma$-algebra. To see that (17) is true, observe that the set $\cup_{M=1}^\infty \cap_{n\geq M} B(n,\epsilon)$ contains the sequences from $A^\infty$ for which the value $(1/n)\sum_{i=1}^n f(T^{i-1}x) - \tilde{f}(x)$ is eventually (as $n \to \infty$) less than $\epsilon$. The equation (17) now follows.

§75 **Observation**. The ergodic theorem is a powerful tool, not unlike the law of large numbers for i.i.d. measures. We may apply it to the relative probabilities by writing

$$p_m(a_1^m | x_1^n) = \frac{1}{n-m+1}\sum_{i=0}^{n-m} 1_{a_1^m}(T^i x)$$

where $1_{a_1^m} : A^\infty \to \{0,1\}$ is the indicator function for the cylinder $\pi_m^{-1}(a_1^m)$. The ergodic theorem now tells that if $\mu$ is an ergodic probability measure, the limiting relative probability, $p_m(a_1^m|\mathbf{x})$, is almost surely defined and equal to the measure of the cylinder $a_1^m$, that is,

$$p_m(a_1^m|\mathbf{x}) = \int_{A^\infty} 1_{a_1^m}(\mathbf{x}) d\mu(x) = \mu(a_1^m)$$

almost surely.

§76 **Definition**. For $\mathbf{x} \in \mathcal{S}$ let $\mu_\mathbf{x}$ denote the stationary probability measure induced by $\mathbf{x}$. Now define

$$[\mathbf{x}] \stackrel{\text{def}}{=\!=} \{\mathbf{y} \in \mathcal{S} \mid \mu_\mathbf{y} = \mu_\mathbf{x}\}.$$

that is, the set of stochastic source instances whose limiting relative probabilities agree with those of $\mathbf{x}$. The set $[\mathbf{x}]$ is also known as the set of sequences that are *frequency equivalent* with $\mathbf{x}$. For a stationary measure $\mu$ we now define

$$\mathcal{T}(\mu) \stackrel{\text{def}}{=\!=} \{\mathbf{x} \in \mathcal{S} \mid \mu_\mathbf{x} = \mu\}.$$

The set $\mathcal{T}(\mu)$ is known as the set of *frequency typical sequences* of $\mu$, and $\mathcal{T}(\mu)$ is measurable as seen by the following argument. By the ergodic theorem the set for which $p_n(a_1^n|\mathbf{x}) = \mu(a_1^n)$ holds is measurable (and has measure 1). We may write $\mathcal{T}(\mu) = \cap_{n=1}^\infty \cap_{a_1^n \in A^n} \{\mathbf{x} \in A^\infty \mid p_n(a_1^n|\mathbf{x}) = \mu(a_1^n)\}$, showing that $\mathcal{T}(\mu)$ is measurable.

§77 **Observation**. A natural question to raise at this point is if we, given a stationary measure $\mu$ can find a source instance $\mathbf{x} \in \mathcal{S}$ such that $\mu$ is induced by $\mathbf{x}$, that is, so that $\mu = \mu_\mathbf{x}$. In case $\mu$ is also *ergodic*, the answer turns out affirmative.

§78 **Theorem**. *Let $\mu$ be a stationary probability measure. Then $\mu$ is ergodic if and only if $\mu(\mathcal{T}(\mu)) = 1$.*

§79 **Observation**. Note, that by the definition of typical sequences we have $\mathcal{T}(\mu) = [\mathbf{x}]$ for any $\mathbf{x} \in \mathcal{T}(\mu)$. Furthermore, if $\mu$ is ergodic we see that the set of typical sequences is indeed non-empty and thus $\mu = \mu_\mathbf{x}$ for any $\mathbf{x} \in \mathcal{T}(\mu)$.

§80 **Proof (of theorem §78)**: Assume $\mu$ is ergodic. For any $n \in \mathbb{N}$ and any $a_1^n \in A^n$ we have by the ergodic theorem (see observation §75),

$$p_n(a_1^n|\mathbf{x}) = \mu(a_1^n), \tag{18}$$

almost surely, that is, (18) holds for for all $\mathbf{x}$ in some set $G(a_1^n) \subset A^\infty$ for which $\mu(G(a_1^n)) = 1$. This means, that the set $B(a_1^n) \stackrel{\text{def}}{=\!=} A^\infty \backslash G(a_1^n)$ has measure 0. We now construct $B$ as follows

$$B \stackrel{\text{def}}{=\!=} \bigcup_{a_1^n \in A^n,\, n=1,2,\ldots} B(a_1^n)$$

and note that $\mu(B) = 0$. The set $B$ has the property that for any $\mathbf{x} \in A^\infty \backslash B$ the following hold

$$\forall n \in \mathbb{N}\ \forall a_1^n \in A^n : p_n(a_1^n|\mathbf{x}) = \mu(a_1^n).$$

This proves that $\mu(\mathcal{T}(\mu)) = 1$.
Now assume conversely that $\mu(\mathcal{T}(\mu)) = 1$. For any block length $m \in \mathbb{N}$, any block $a_1^m \in A^m$ and any $x \in \mathcal{T}(\mu)$ we then have

$$\lim_{n\to\infty} \frac{|\{i \in [1, n-m+1] \mid x_i^{i+m-1} = a_1^m\}|}{n} = \lim_{n\to\infty} \frac{\sum_{i=0}^{n-m} w(a_1^m|T^i x)}{n} = \mu(a_1^m)$$

where $w(a_1^n|x) = 1$ if $a_1^n = x_1^n$ and $w(a_1^n) = 0$ otherwise. For any $l \in \mathbb{N}$ and any $b_1^l \in A^l$ we have

$$\lim_{n\to\infty} \frac{\sum_{i=0}^{n-m} w(a_1^m|T^i x) w(b_1^l|x)}{n} = \mu(a_1^m) w(b_1^l|x)$$

Integration of $w(a_1^m|x)$ yields $\mu(\pi_m^{-1}(a_1^n))$ because of the ergodic theorem, so integrating the whole equation leads to

$$\lim_{n\to\infty} \frac{\sum_{i=0}^{n-m} \mu(T^{-i}\pi_m^{-1}(a_1^m) \cap \pi_l^{-1}(b_1^l))}{n} = \mu(a_1^m)\mu(b_1^l).$$

Because sets of the form $\pi_m^{-1}(a_1^m)$ generates $\mathcal{F}$ we see that in fact

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-m} \mu(T^{-i}B \cap C) = \mu(B)\mu(C)$$

for any sets $B, C \in \mathcal{F}$. Hence $T^{-1}B = B$ implies $\mu(B) = \mu(B)\mu(B)$ which shows that the measure of $B$ is trivial and thus $\mu$ is ergodic.

§81 **Observation**. In order to better understand the structure of the stationary measures on $A^\infty$, we present the ergodic decomposition theorem. Let $\mathcal{E} \subset \mathcal{S}$ be the set of stochastic sequences for which the induced measure is ergodic, that is, $\mathcal{E} \stackrel{\text{def}}{=\!=} \{x \in \mathcal{S} \mid \mu_x \text{ is ergodic}\}$. Consider also the set of frequency equivalence classes $C = \{[x] \mid x \in A^\infty\}$ and the map $\iota$ which maps sequences into the frequency equivalence classes, $\iota(x) = [x]$. The set $C$ can be made into a measurable space $(C, \mathcal{G})$ by equipping it with the $\sigma$-algebra, $\mathcal{G}$, spanned by the atoms of $C$. This makes the projection map $\iota$ measurable (since $\iota^{-1}([x]) = [x] \in \mathcal{F}$.) For any stationary measure $\mu$ on $(A^\infty, \mathcal{F})$ the projection $\iota$ now yields a measure on $(C, \mathcal{G})$, namely $\omega = \mu \circ \iota^{-1}$. These are the ingredients in the ergodic decomposition theorem.

§82 **Theorem** (ERGODIC DECOMPOSITION). *Any stationary measure* $\mu$ *has* $\mu(\mathcal{S}) = 1$ *and even* $\mu(\mathcal{E}) = 1$. *Indeed, for a measurable set* $B \in \mathcal{F}$*, we have*

$$\mu(B) = \int_{[x]\in C} \mu_{[x]}(B) d\omega([x])$$

*where* $\mu_{[x]} = \mu_x$.

§83 **Proof**: See e.g. [Shields, 1996].

§84 **Example**. In the simplest case of theorem §82 we have the non-ergodic, stationary measure $\mu$ concentrated in a finite number of frequency equivalence classes of $\mathcal{S}$: Let $\nu$ and $\eta$ be ergodic measures. Then $\mu = 1/2\nu + 1/2\eta$ is a stationary measure, and in a sense it is a canonical example of one, because the ergodic decomposition theorem states that *all* stationary measures are a mix of ergodic measures.

§85 **Observation**. Let $\mu$ be an ergodic measure and consider the value $p_\infty(\mathbf{x}) \stackrel{\text{def}}{=\!=} \lim_{n\to\infty} \mu(x_1^n)$ which exists for all $\mathbf{x}$ because $\mu(x_1^n)$ is nonnegative and non-increasing in $n$. We have $\mu(x_1^n) \leq \mu(x_2^n)$ by the stationarity condition and it follows that $p_\infty(\mathbf{x}) \leq p_\infty(T\mathbf{x})$. As the sub-invariance lemma (below) shows, this yields $p_\infty(\mathbf{x}) = p_\infty$ almost surely, for a constant $p_\infty$. Intuitively we expect $p_\infty = 0$ and indeed this is the case in all but some very special cases. To see this, assume $p_\infty > 0$ and define the set $B = \{\mathbf{x} \in A^\infty \mid p_\infty(\mathbf{x}) = p_\infty\}$ of measure $\mu(B) = 1$. For any $n \in \mathbb{N}$ let $B_n$ be the set of $n$-prefixes of the elements of $B$, that is, $B_n = \pi_n^{-1}(B) \subseteq A^n$. We observe that $|B_n| \leq |B|$ for any value of $n$ and thus

$$1 = \mu(B) \geq \sum_{a_1^n \in B_n} \mu(a_1^n) \geq |B_n| p_\infty \geq |B| p_\infty$$

for any value of $n$, where we used that $\mu(a_1^n) \geq p_\infty$. It follows that $\mu$ is concentrated on a finite subset of $A^\infty$.

§86 **Lemma** (SUBINVARIANCE LEMMA). *Let $f : A^\infty \to [0,1]$ be a measurable function and $\mu$ an ergodic measure. If $f$ is sub-invariant, that is, if $f(T\mathbf{x}) \leq f(\mathbf{x})$, then $f(\mathbf{x})$ is constant almost surely.*

§87 **Proof**: Set $B_\alpha \stackrel{\text{def}}{=\!=} \{\mathbf{x} \in A^\infty \mid f(\mathbf{x}) \leq \alpha\}$ for any $\alpha \in [0,1]$. From this definition and the subinvariantness of $f$, it is clear that

$$B_\alpha \subseteq T^{-1} B_\alpha. \qquad (\dagger)$$

Let $C \stackrel{\text{def}}{=\!=} \cup_{i\geq 0} T^{-i} B_\alpha$. We now argue that $C$ is an invariant set. Firstly, to see that $C \subseteq T^{-1}C$ let $\mathbf{x} \in C$. By definition of $C$ we have $\mathbf{x} \in T^{-i}B_\alpha$ for some $i \geq 0$ and by $(\dagger)$ we find $\mathbf{x} \in T^{-i}T^{-1}B_\alpha \subseteq C$. Secondly, note that $C \supseteq T^{-1}C$ follows immediately from the definition of $C$. Because of $(\dagger)$ we have a growing sequence $B_\alpha \subseteq T^{-1}B_\alpha \subseteq T^{-2}B_\alpha \subseteq \cdots$ of sets and the limit of this sequence is $C$. It follows that $\mu(C) = \lim_{i\to\infty} \mu(T^{-i}B_\alpha)$ and by the stationarity of the measure we find $\mu(C) = \mu(B_\alpha)$. Since $C$ is invariant and by the ergodicity of $\mu$ the measure of $B_\alpha$ must be trivial so that $\mu(B_\alpha)$ is either 0 or 1. We note that $B_\alpha \subseteq B_\beta$ if $\alpha \leq \beta$ and introduce the constant $f \stackrel{\text{def}}{=\!=} \inf\{\alpha \in [0,1] \mid \mu(B_\alpha) = 1\}$. For any $\epsilon > 0$ we have $\mu\{\mathbf{x} \mid f(\mathbf{x}) \leq f - \epsilon\} = 0$ and $\mu\{\mathbf{x} \mid f(\mathbf{x}) \leq f + \epsilon\} = 1$ leading us to the conclusion that $\mu\{\mathbf{x} \mid f(\mathbf{x}) = f\} = 1$ which concludes the proof.

§88 **Summary**. In this section we have progressed from the simple notion of stochastic source instances to the more complex machinery of measure theory. In doing so we have gained insight into the structure of the infinite sequences, $A^\infty$, from the point of view of limiting relative probabilities. The natural classification of source instances into frequency equivalence classes $[\mathbf{x}]$ leads to the ergodic measures as being exactly those measures which are concentrated in a single such class. Thus the set of stochastic source instances, $\mathcal{S}$ is the largest set where the probabilistic models are useful: every stationary ergodic measure has its support inside $\mathcal{S}$. On the other hand, every element of $\mathcal{S}$ is in the support for some stationary ergodic measure.

## The Entropy-rate theorem

§89 **Observation**. In this section we consider one of the most important results in ergodic theory as seen from the point of view of an information theorist. Previously (see observation §85) we touched upon the fact that for an ergodic measure the value of $\mu(x_1^n)$ decreases towards zero as

$n$ grows. This is no surprise, especially if one (as we do) consider measures induced by limiting relative probabilities: the longer a sequence $a_1^n$ we consider, the fewer times we expect to see it as a subsequence of some $\mathbf{x}$. As an example we may consider an infinite sequence $\mathbf{x}$ over the binary alphabet such that $p_n(a_1^n|\mathbf{x}) = 2^{-n}$ for any binary word $a_1^n$. If $\mu$ is the i.i.d. measure with $\mu(0) = \mu(1) = 1/2$ we see that not only does $\mu(a_1^n) = p_n(a_1^n|\mathbf{x})$ converge to zero as $n$ grows, but it does so with an exponential *rate* of 1. The entropy-rate theorem (see theorem §108) tells that any ergodic measure exhibits this behavior for all its typical sequences. The rate, which depends on the measure, is known as the *entropy*, hence the name of the theorem.

§90 **Remark**. The proof of the entropy-rate theorem presented here is based on a combinatorial method known as *packing and counting* which was developed in [Ornstein and Weiss, 1983]. The proof is based on a study of the structure of the typical sequences for an ergodic measure. For this task it is useful to keep in mind the image of "windows" on sequences. For example, the notion of a *sliding window* of length $w$ on a sequence $x_1^n \in A^n, n > w$ is well known. By this we mean the procedure of studying $x_1^n$ in pieces of length $w$ starting with $x_1^w$, then $x_2^{w+1}$ and all the way through $x_{n-w+1}^n$. The name of the procedure is due to the imaginative "window" which is placed on $x_1^n$ and through which we see a part of $x_1^n$ of length $w$. In place of a single window being slided along $x_1^n$ we may think of it as a whole sequence of $n - w + 1$ windows, $[1, w], [2, w + 1], \ldots, [n - w + 1, n]$, where each window is represented as an integer interval. Thus, $[1, w]$ is the window which, if placed on a sequence $x_1^n$, would reveal $x_1^w$. The sliding window procedure is just one way to study $x_1^n$, however. A more general approach is to allow all possible windows, that is, $[i, j]$ where $1 \le i \le j \le n$. To facilitate this we introduce in a more formal way these ideas.

§91 **Definition**. Let $[1, n]$ denote the integer interval, that is $[1, n] \stackrel{\text{def}}{=\!=} \{1, 2, \ldots, n\}$. Let $\mathcal{C}$ be a set of subintervals of $[1, n]$; the elements of $\mathcal{C}$ are sometimes called *windows*. If $\mathcal{C}$ consists of disjoint intervals such that at least $(1-\delta)n$ of the integers in $[1, n]$ are covered, we say that $\mathcal{C}$ is a $(1-\delta)$-*packing* of $[1, n]$. If at least $(1 - \delta)n$ of the integers in $[1, n]$ are starting points for a subinterval in $\mathcal{C}$, we say that $\mathcal{C}$ is a *strong* $(1 - \delta)$-*cover* of $[1, n]$. A cover is said to be $L$-bounded, if each member of the cover has length at most $L$.

§92 **Definition**. Let $B \subseteq A^*$, let $x_1^n \in A^n$ be a sequence and assume that we may write $x = b_1 \cdots b_j$ for some $b_i \in A^*$. If $b_i \in B$, we say that $x_1^n$ is *built from* $B$. The $b_i$'s are known as the *building blocks* of $x_1^n$. If $\sum_{\{i|b_i \in B\}} |b_i| \ge (1 - \delta)n$ we say that $x_1^n$ is $(1 - \delta)$-built from $B$.

§93 **Example**. Let $x_1^n$ be a sequence. Sliding a windows of length $w$ across $x_1^n$ would produce a collection $\mathcal{C}$ of subintervals of $[1, n]$ (or windows on $x_1^n$) as described above. If $w/n \le \delta$ we see that at least $(1-\delta)n$ of the indexes in $[1, n]$ (namely $1, \ldots, n-w+1$) are starting points for one of the windows, and thus $\mathcal{C}$ is a $(1 - \delta)$-cover. In fact, because all the windows have equal and thus limited length, $w$, the cover is $w$-bounded. Turning our attention to what one see through the windows, that is the set $B = \{x_s^t \mid [s, t] \in \mathcal{C}\}$, we find that we can write $x_1^n = x_1^w x_{w+1}^{2w} \cdots x_{(k-1)w-1}^{kw}$ provided, of course, that $n = kw$ for an integer $k$. In this case $x_1^n$ is built from $B$. If on the other hand $n = kw + r$ we do still know that $r < w \le \delta n$ and so we have $x_1^n = x_1^w \cdots x_{(k-1)w-1}^{kw} x_{kw+1}^n$ where only $x_{kw+1}^n$ is not a member of $B$, so $x_1^n$ is $(1 - \delta)$-built from $B$.

§94 **Observation**. Consider a sequence $x_1^n$ and a $(1 - \delta)$-cover $\mathcal{C}$. Loosely speaking, $\mathcal{C}$ is a generalization and approximation of the sliding window procedure. It is a generalization because the window size is not fixed and an approximation because the "slide" may be not quite smooth.

Similarly, if $\mathcal{C}$ is a $(1-\delta)$-packing, then it implies that the sequences $x_1^n$ is approximately revealed by the windows of $\mathcal{C}$ in the sense that we may write $x_1^n$ as a concatenation like this $x_1^n = b_1 \cdots b_j$ such that $\sum |b_i| = (1-\delta)n$ where the sum is taken over all such $b_i$ that shows up in one of the windows of $\mathcal{C}$. To make this last statement precise: The $b_i$'s, for which there is a window $[s,t] \in \mathcal{C}$ such that $x_s^t = b_i$, provide at least $(1-\delta)n$ out of the total of $n$ symbols that make up $x_1^n = b_1 \cdots b_j$.

§95 **Lemma** (PACKING LEMMA). *If $n \geq L/\delta$, then any $L$-bounded strong $(1-\delta)$-cover $\mathcal{C}$ of $[1,n]$ contains a $(1-\delta)$-packing.*

§96 **Observation**. Let us try and express the Packing lemma in terms of windows. The $(1-\delta)$-cover $\mathcal{C}$ plays a role similar to that of a sliding window. Each window is limited (by $L$) in size, and at almost all possible starting points $i \in [1,n]$, a window starts. By studying $x_1^n$ through the windows of $\mathcal{C}$ we see a number ($|\mathcal{C}|$, in fact) of subsequences of $x_1^n$. The Packing lemma tells, that $x_1^n$ can be written as a concatenation made almost entirely of these subsequences. This seems intuitively plausible, and indeed the proof is easy.

§97 **Proof (of packing lemma)**: Let $i_1$ be the smallest number in $[1,n]$ such that there is an interval in the cover which has $i_1$ as left endpoint. Pick any one of the intervals of $\mathcal{C}$ which has $s_1$ as left endpoint and let $[s_1,t_1]$ be the chosen interval. Let $s_2$ be the smallest number in $[t_1+1,n]$ such that an interval in the cover has $s_2$ as left endpoint. Proceed in a similar way until the situation occur where there is no interval from $\mathcal{S}$ that begins in the interval $[s_j+1,n]$. The selected intervals from $\mathcal{C}$ are now seen to be an $(1-\delta)$-packing, by the following argument: Any integer $i \in [1,n]$ not covered by the selected intervals can not be a left-endpoint of an interval in $\mathcal{C}$. There can be only $\delta n$ of this kind of integers which proves the theorem.

§98 **Definition**. Let $M \in \mathbb{N}$ and $B \subseteq \cup_{m \geq M} A^m$ and $\delta > 0$ and $n \in \mathbb{N}$. Then define

$$G(B,\delta,n) \stackrel{\text{def}}{=\!=} \left\{ x_1^n \in A^n \;\middle|\; x_1^n = b_1 \cdots b_j,\ b_i \in A^* \text{ and } \sum_{\{i \mid b_i \in B\}} |b_i| \geq n(1-\delta) \right\}.$$

We say that $G(B,\delta,n)$ is the set of sequences of length $n$ that can be $(1-\delta)$-*build* from sequences from $B$.

§99 **Observation**. This definition should be seen in the light of the packing lemma. While the packing lemma gives a condition for how we may get a set $B$ of building blocks (viz. by looking at the sequence through windows from a cover) big enough to ensure that we can have the sequence $(1-\delta)$-built from $B$, the set $G$ captures the opposite idea: Given a set $B$ of building blocks, what sequences can we construct that are $(1-\delta)$-built from $B$.

§100 **Observation**. In the proof of the ergodic theorem it is an important step to upper-bound the cardinality of $G(B,\delta,n)$. In the following we make preparations for a result in this respect. Let $G = G(B,\delta,n)$ be given as above, and let $x_1^n \in G$. By the definition of $G$ there is a set of sequences $b_1,\ldots,b_j \in A^*$ so that $x_1^n = b_1 \ldots b_j$. We may picture $x_1^n$ as a concatenation of blocks $b_i$, and the picture is clearly dominated[5] by those $b_i$'s that are members in $B$. Focusing on the *structure* of this concatenation leads to the set $S = \{[s,t] \mid x_s^t = b_i \in B \text{ for some } i\}$. Note that $S$ is a collection of disjoint subintervals of $[1,n]$, that each subinterval is of length at least $M$ and

[5] In the sense that $\sum_{\{i \mid b_i \in B\}} |b_i| = (1-\delta)n$.

that $S$ is an $(1-\delta)$-*packing* of $[1,n]$. Also note, that there might be many ways to write $x_1^n$ as a concatenation. Some of these will have the same structure (i.e. the same set $S$), others will not. In our efforts to estimate the size of $G(B,\delta,n)$, the number of different structures as well as the number of concatenations possible for any given structure are estimated separately.

§101 **Definition**. Let $M$ and $n > M$ be integers. A *skeleton in* $[1,n]$ is a collection $S = \{[s_i,t_i]\}$ of disjoint, integer subintervals of $[1,n]$ such that the union of the subintervals covers at least $(1-\delta)n$ of the interval $[1,n]$ and such that $t_i - s_i + 1 \geq M$. Let a set $B \subseteq A^*$ and a skeleton $S$ be given. The sequence $x_1^n$ is then said to be *compatible with* $S$ if $x_s^t \in B$ exactly when $[s,t] \in S$.

§102 **Lemma** (COUNTING LEMMA). *Let $h$ and $\epsilon$ be positive numbers. Pick a number $M$ large enough so that $H_b(2/M) \leq \epsilon/2$ and a number $\delta > 0$ so that $\delta \log|A| \leq \epsilon/2$. For each $m \geq M$ let there be given a set $B_m \subseteq A^m$ with $|B_m| \leq 2^{m(h+\epsilon)}$ and set $B = \cup B_m$. Then,*

$$|G(B,\delta,n)| \leq 2^{n(h+2\epsilon)}.$$

*The function $H_b$ is the binary entropy defined as $H_b(p) \stackrel{\text{def}}{=\!=} -p\log p - (1-p)\log(1-p)$ for any value of $p \in [0,1]$.*

§103 **Observation**. The counting lemma translates bounds on $|B_m|$ into a bound on $|G(B,\delta,N)|$. In other words, if for every length the number of building blocks of that length is bounded, then so is the total number of sequences that can be $(1-\delta)$-build from the blocks. Consider the following simplified case: Let $\tilde{G}$ be the set of sequences of length $n$ which can be build as concatenations of sequences $b \in B_M$ of length $|b| = M$, where $kM = n$. Assume we have the bound $|B_M| \leq 2^{Mh}$ for some number $h$. We now find $|\tilde{G}| = (|B_M|)^k \leq (2^{Mh})^k = 2^{nh}$. This trivial result, while similar in spirit to the counting lemma, is simpler in two fundamental ways. Firstly, we require the sequences in $\tilde{G}$ to be completely built from sequences from $B_M$. The lemma allows for $(1-\delta)$-built sequences, that is, sequences of which small parts are foreign (i.e. not from $B$). Secondly, the example allows only subsequences of equal length, because $|b| = M$ for any $b \in B_M$. These two simplification gives one and only one possible skeleton over which the sequences of $\tilde{G}$ can be built, namely the skeleton $S = \{[1,M],[M+1,2M],\ldots,[(k-1)M+1,kM]\}$.

§104 **Proof (of the counting lemma)**: Let $S$ be a skeleton in $[1,n]$. Write $G$ as short for $G(B,\delta,n)$ and let $G_S$ denote the set of sequences from $G$ which are compatible with $S$. Because any sequence $x_1^n \in G$ is $(1-\delta)$-built from $B$, there is a skeleton $S$ such that $x_1^n \in G_S$, and thus $G = \cup_S G_S$. We now estimate the number of elements in $G_S$. For any interval $[s,t] \in S$ of length $m = t-s+1$ the corresponding subsequence $x_s^t$ must be a member of $B_m$, and the size of this set is at most $2^{m(h+\epsilon)}$. The intervals of $S$ is non-overlapping, so $\sum_{I\in S}|I| \leq n$, and so the total number of combinations for filling out the skeleton is

$$\prod_{I\in S} 2^{|I|(h+\epsilon} \leq 2^{n(h+\epsilon)}.$$

And finally, the number of indices $i \in [1,n]$ which are not members of any interval in $S$, are less than $n\delta$. The symbols $x_i$ for such $i$'s can be freely chosen from $A$. In all we have shown

$$|G_S| \leq 2^{n(h+\epsilon)}|A|^{n\delta}.$$

We now estimate the number of skeletons. Any interval in $S$ must be at least $M$ long, so there can be no more than $n/M$ of them. This implies $2n/M$ start or endpoints for the intervals that can be chosen from the range $[1, n]$. The total number of skeletons is thus upper bounded by

$$\sum_{j \le 2n/M} \binom{n}{j}.$$

Another way of looking at the value of this sum is to consider the number of binary sequences $b_1^n$ of length $n$ where every $b_i$ is 1 in the cases where $i$ is either an left endpoint or an right endpoint of an interval and $b_i$ is 0 in all other cases. The sequence $b_1^n$ will have at most $2n/M$ ones. Thus it is reasonable to think we can upper bound the sum by $2^{nH_b(2/M)}$.

Writing $p = 2/M$ and making the reasonable assumption that $p \le 1/2$, we proceed to make a number of calculations. Let $q$ be a number such that $0 \le q \le p$. Then

$$H_b(p) = -p \log p - (1-p)\log(1-p) \ge -q \log p - (1-q)\log(1-p)$$

since when $p \le 1/2$ we have $-\log p \ge -\log(1-p)$. We may then write

$$\begin{aligned} 2^{-nH_b(p)} &= 2^{-n(-p\log p-(1-p)\log(1-p))} \\ &\le 2^{-n(-q\log p-(1-q)\log(1-p))}, && \text{for any } q \text{ where } 0 \le q \le p \\ &= p^{nq}(1-p)^{n(1-q)}, && \text{for any } nq \text{ where } 0 \le nq \le np \\ &= p^k(1-p)^{n-k}, && \text{for any } k \text{ where } 0 \le k \le np. \end{aligned}$$

For any $k$ so that $0 \le k \le np = 2n/M$ we thus have $2^{-nH_b(p)} \le p^k(1-p)^{n-k}$. This can be used to complete the upper bound on the number of skeletons,

$$2^{-nH_b(p)} \sum_{j \le 2n/M} \binom{n}{j} \le \sum_{j \le 2n/M} \binom{n}{j} p^j (1-p)^{n-j} \le \sum_{j=0}^{n} \binom{n}{j} p^j (1-p)^{n-j} = (p+1-p)^n = 1$$

yielding the desired result,

$$\left|\{S \mid S \text{ is a skeleton in } [1,n]\}\right| \le 2^{nH_b(2/M)}.$$

Combining the bound on the number of skeletons and the bound on the number of strings in $G_S$ for each skeleton, provides the final result

$$|G| = \sum_{S \text{ skeleton in } [1,n]} |G_S| \le 2^{nH_b(2/M)} 2^{n(h+\epsilon)} |A|^{n\delta} \le 2^{n\epsilon/2 + n(h+\epsilon) + n\epsilon/2} = 2^{n(h+2\epsilon)}$$

which completes the proof.

§105 **Lemma** (SLIDING WINDOW LEMMA). *Let $\delta > 0$ and let $U_m \in A^m$ be a set for which $\mu(U_m) \ge 1 - \delta/2$ where $\mu$ is a stationary ergodic measure. Then the following holds almost surely: There is an integer $N = N(x)$ so that for every $n > N$ we have*

$$x_i^{i+m-1} \in U_m$$

*for at least $(1-\delta)n$ indices $i \in [1, n-m+1]$.*

§106 **Proof**: Let $x \in \mathcal{T}(\mu)$. Then we have

$$\lim_{n\to\infty} p_m(U_m|x_1^n) = 1 - \frac{\delta}{2}$$

and thus there is an integer N such that for any $n > N$ we find $p_m(U_m|x_1^n) \geq 1-\delta$. This means that for at least $(1-\delta)n$ of the values of $i \in [1, n-m+1]$ the subsequence $x_i^{i+m-1} \in U_m$. Because $\mu(\mathcal{T}(\mu)) = 1$ the result holds almost surely as required.

§107 **Observation**. One possible interpretation of this lemma is, that for any set $U_m \subseteq A^m$ of high probability and any $\mu$-typical sequence $x \in A^\infty$ sliding a window of size $m$ along a sufficient long prefix of $x$, the window will in most cases show an element of $U_m$. Another way to put this, it that the set $\mathcal{C}$ of windows $[s,t]$ where $x_s^t \in B_m$ is a $(1-\delta)$-cover of $[1,n]$.

§108 **Theorem** (ENTROPY-RATE THEOREM). *Let $\mu$ be an stationary ergodic probability measure on $(A^\infty, \mathcal{F})$. Then there exists a constant $h \geq 0$, such that*

$$\lim_{n\to\infty} -\frac{1}{n}\log\mu(x_1^n) = h, \quad \textit{almost surely.}$$

§109 **Proof**: Let us define

$$h(x) \stackrel{\text{def}}{=\!=} \liminf_{n\to\infty} -\frac{1}{n}\log\mu(x_1^n),$$

for any $x \in A^\infty$ and we have,

$$\begin{aligned} h(Tx) &= \liminf_{n\to\infty} -\frac{1}{n}\log\mu(x_2^{n+1}) \\ &\leq \liminf_{n\to\infty} -\frac{1}{n}\log\mu(x_1^{n+1}) \\ &= h(x), \end{aligned}$$

where the inequality follows from the fact that for each $n > 1$, we have $\mu(x_2^n) \geq \mu(x_1^n)$. Thus $h(Tx) \leq h(x)$ and by the subinvariance lemma §86 we have that $h(x) = h$ almost surely for a constant $h$. We have now established that

$$\lim_{n\to\infty} -\frac{1}{n}\log\mu(x_1^n)$$

is lower bounded almost surely by a constant $h$. We now want to show that $h$ is also an upper bound almost surely.

**The plan:** We now proceed in three steps. First we show that long sequences can be $(1-\delta)$ built from subsequences each of which has measure $\mu(x_1^j) \geq 2^{-j(h+\epsilon)}$. We then use this fact to show that there are only $2^{-n(h+\epsilon)}$ sequences of length $n$ which can be constructed in this way. Finally, we show that this upper bound in cardinality translates into an lower bound in probability, that is, for these sequences of length $n$ we essentially have $\mu(x_1^n) \geq 2^{-n(h+\epsilon)}$, which in turn implies that $h$ is an upper bound on $-(1/n)\log\mu(x_1^n)$.

Let $\epsilon$ be an arbitrarily small positive number.

**First step:** Note, that by the definition of $h(x)$ we have $h(x) = \sup_m \inf_{n>m} -(1/n)\log\mu(x_1^n)$, or in words: for all $m \in \mathbb{N}$ there exists and $n > m$ such that $-(1/n)\log\mu(x_1^n) \leq h(x)+\epsilon$. This leads

us to the fact, that $(-1/n)\log\mu(x_1^n) \le h+\epsilon$ infinitely often, almost surely, or correspondingly, $\mu(x_1^n) \ge 2^{-n(h+\epsilon)}$ infinitely often, almost surely.

Let $M$ be an integer which will be the minimum size of the building blocks. Let the set of building blocks be $B_n \stackrel{\text{def}}{=\!=} \{a_1^n \in A^n \mid \mu(a_1^n) \ge 2^{-n(h+\epsilon)}\}$ and define $n(\mathbf{x})$ as the least integer $n \ge M$ such that $x_1^n \in B_n$ (if there is no such integer, we may set $n(\mathbf{x}) = \infty$ for completeness). As we have just seen, $n(\mathbf{x})$ is almost surely finite, so for any $\delta > 0$ we can pick an integer $L \ge M$ such that $\mu(\{\mathbf{x} \mid n(\mathbf{x}) \le L\}) \ge 1-\delta/4$. Expressed differently, we have $\mu(U_L) \ge 1-\delta/4$ for $U_L \stackrel{\text{def}}{=\!=} \{x_1^L \mid x_1^i \in B_i \text{ for some } i \in [M,L]\} \subseteq A^L$.

Applying the sliding window lemma §105 to the set $U_L$ shows, that eventually (as $n \to \infty$), almost surely, at least $(1-\delta/2)n$ of the indices $i \in [1, n-L+1]$ are starting points for a subsequence from $U_L$, that is, $x_i^{i+L-1} \in U_L$. This in turn implies that for at least $(1-\delta/2)n$ of the indices $i$, there exists an interval $[i,j]$ such that $x_i^j \in B_{j-i+1}$ and $M \le j-i+1 \le L$. The set of such intervals is a strong $(1-\delta/2)$-cover of $[1,n]$, so we may (provided $n$ is large enough) extract a $(1-\delta)$-packing of $[1,n]$ from it. Thus we have shown that with $B \stackrel{\text{def}}{=\!=} \cup_{M\le i\le L} B_i$

$$x_1^n \in G(B,\delta,n), \text{ eventually (as } n\to\infty), \text{ almost surely.}$$

This completes the first step in the plan.

**Second step:** In the preceding section we allowed for any integer $M$ and any number $\delta > 0$. We now fix $M$ so that $H_b(2/M) \le \epsilon/2$, and $\delta > 0$ such that $\delta \log|A| \le \epsilon/2$. The counting lemma §102 then provides an upper bound,

$$|G(B,\delta,n)| \le 2^{n(h+2\epsilon)},$$

on the number of sequences of length $n$ that are $(1-\delta)$-built from $B$.

**Third step:** To execute the final part of the plan, let $V_n \stackrel{\text{def}}{=\!=} \{x_1^n \in G(B,\delta,n) \mid \mu(x_1^n) \le 2^{-n(h+3\epsilon)}\}$. Clearly,

$$\mu(V_n) \le |G(B,\delta,n)|2^{-n(h+3\epsilon)} \le 2^{nh+2n\epsilon-nh-3n\epsilon} = 2^{-n\epsilon}$$

and so by the Borel-Cantelli principle, $x_1^n \notin V_n$ eventually (as $n\to\infty$), almost surely. We do know, however, that $x_1^n \in G(B,\delta,n)$ eventually, almost surely, so we get an lower bound on the probability, $\mu(x_1^n) \ge 2^{-n(h+3\epsilon)}$, eventually, almost surely. Taking limes superior yields

$$\limsup_{n\to\infty} -\frac{1}{n}\log\mu(x_1^n) \le h+3\epsilon$$

and this completes the proof since then

$$h = \liminf_{N\to\infty} -\frac{1}{n}\log\mu(x_1^n) \le \limsup_{n\to\infty} -\frac{1}{n}\log\mu(x_1^n) \le h+3\epsilon$$

for all positive values of $\epsilon$.

§110 **Summary**. Using the technique of packing and counting we have established a standard result of information theory: the entropy-rate theorem. It is noteworthy that the ergodic theorem, which is used in the proof of the entropy-rate theorem, may also be derived using the packing

and counting method, and consequently that these results can be constructed without using any of the machinery of traditional measure theory. Compared to the traditional approach we find that the packing and counting technique provides a fresh insight and a distinctive intuitive appeal. The possibility of proving important classical results combined with this intuitive appeal speaks, we feel, in favor of the notion of source instances as the building blocks of a source concept.

## A coding theorem for the probabilistic case

§111 **Remark**. The entropy-rate theorem provides us with one possible definition of entropy for an ergodic measure, namely by defining it as the (uniquely identified) constant $h$, such that

$$h \stackrel{\text{def}}{=\!=} \lim_{n\to\infty} -\frac{1}{n}\log \mu(x_1^n), \text{ almost surely.}$$

Another, and probably more traditional, way of introducing the entropy is outlined in the following. Let $\mu$ be an stationary measure and write $\{\mu_n\}_{n\in\mathbb{N}}$ for its collection of consistent probability measures on $A^n$ (i.e. $\mu_n = \mu \circ \pi_n^{-1}$). We set

$$H_n(\mu_n) \stackrel{\text{def}}{=\!=} -\sum_{a_1^n \in A^n} \mu_n(a_1^n)\log \mu_n(a_1^n) \tag{19}$$

for every $n$. The entropy of $\mu$ is now defined as the following limit

$$H(\mu) \stackrel{\text{def}}{=\!=} \lim_{n\to\infty} \frac{H_n(\mu_n)}{n}, \tag{20}$$

when it exists. The following observation and lemma shows that for a stationary measure, $\mu$, the entropy $H(\mu)$ is always defined.

§112 **Observation**. It is easy to see that $H_{i+j}(\mu_{i+j}) \le H_i(\mu_i) + H_j(\mu_j)$. Let $n = i+j$, and note that the map defined by $a_1^n \mapsto \mu_i(a_1^i)\mu_j(a_{i+1}^n)$ is a probability measure on $A^n$. We now have

$$\begin{aligned} H_n(\mu_n) &= -\sum_{a_1^n} \mu_n(a_1^n)\log \mu_n(a_1^n) \\ &\le -\sum_{a_1^n} \mu_n(a_1^n)\log\Big(\mu_i(a_1^i)\mu_j(a_{i+1}^n)\Big) \\ &= H_i(\mu_i) + H_j(\mu_j). \end{aligned}$$

where the inequality follows from the log-sum inequality (see lemma §276, in Appendix A). If $\mu_i(a_1^i)\mu_j(a_{i+1}^n) = \mu_n(a_1^n)$ for any $a_1^n \in A^n$ and for any values of $i$ and $j$ (and hence also $n$), we say that $\mu$ is independent, and we see that $H(\mu_n) = H(\mu_i) + H(\mu_j)$ in this case. In fact, by the log-sum inequality we know that $\mu$ is independent if and only if $H(\mu_n) = H(\mu_i) + H(\mu_j)$.

§113 **Lemma**. *Let $(b_n)_{n\in\mathbb{N}}$ be a sequence of nonnegative numbers such that $b_{n+m} \le b_n + b_m$ for any pair $n, m \in \mathbb{N}$. Then the limit $\lim_{n\to\infty} b_n/n$ exists.*

§114 **Proof**: Let $b = \inf_{n\in\mathbb{N}} b_n/n$. Let $\epsilon > 0$ and pick $n$ accordingly so that $b_n/n \le b + \epsilon$. Now

assume $m \geq n$ and set $m = np + r$ for suitable integers $p$ and $0 \leq r < n$. By the assumptions, we now have

$$\frac{b_m}{m} \leq \frac{b_{np} + b_r}{m} \leq \frac{p\, b_n}{m} + \frac{b_r}{m}.$$

The fraction $b_r/m$ clearly vanishes as $m \to \infty$, while the fraction $p\, b_n/m \to b_n/n$ as $m \to \infty$. This shows that $\limsup b_m/m \leq b + \epsilon$ and since $\liminf b_m/m \geq b$ the lemma has been proved.

§115 **Theorem**. *For an ergodic measure $\mu$ the entropy rate $h$ and the entropy $H(\mu)$ are equal.*

§116 **Proof**: Assume $h > 0$ (the case $h = 0$ is handled later), let $\epsilon > 0$ be given and define the "good set"

$$G_n = \{x_1^n \mid 2^{-n(h+\epsilon)} \leq \mu(x_1^n) \leq 2^{-n(h-\epsilon)}\} \tag{21}$$

as well as the "bad set" $B_n = A^n \backslash G_n$. We may now write the process entropy $H$ as follows:

$$\begin{aligned} H_n &= -\sum_{x_1^n \in A^n} \mu(x_1^n) \log \mu(x_1^n) \\ &= -\sum_{x_1^n \in G_n} \mu(x_1^n) \log \mu(x_1^n) - \sum_{x_1^n \in B_n} \mu(x_1^n) \log \mu(x_1^n). \end{aligned}$$

We proceed by making an estimate of the sum over $B_n$. Note first, that writing $\mu(x_1^n \mid B_n) = \mu(x_1^n)/\mu(B_n)$ we have $\sum_{x_1^n \in B_n} \mu(x_1^n \mid B_n) = 1/\mu(B_n) = \sum_{x_1^n \in B_n} 1/(\mu(B_n) \log |B_n|)$. Hence by the log-sum inequality (see lemma §276) we obtain

$$\sum_{x_1^n \in B_n} \mu(x_1^n \mid B_n) \log \mu(x_1^n \mid B_n) \geq \sum_{x_1^n \in B_n} \mu(x_1^n \mid B_n) \frac{1}{\mu(B_n) \log |B_n|} = -\log |B_n| \tag{22}$$

which be useful in a moment. Moving on, we write

$$\begin{aligned} -\sum_{x_1^n \in B_n} \mu(x_1^n) \log \mu(x_1^n) &= -\sum_{x_1^n \in B_n} \mu(x_1^n \mid B_n) \mu(B_n) \big( \log \mu(x_1^n \mid B_n) + \log \mu(B_n) \big) \\ &= -\mu(B_n) \sum_{x_1^n \in B_n} \mu(x_1^n \mid B_n) \log \mu(x_1^n \mid B_n) - \sum_{x_1^n \in B_n} \mu(x_1^n) \log \mu(B_n) \\ &\leq \mu(B_n) \log |B_n| - \mu(B_n) \log \mu(B_n) \end{aligned}$$

where the inequality follows from (22). Because $B_n \subset A^n$, we find that $\log |B_n| \leq \log |A^n| = n \log |A|$ and hence we have the bound

$$-\sum_{x_1^n \in B_n} \mu(x_1^n) \log \mu(x_1^n) \leq n \log |A| \mu(B_n) - \mu(B_n) \log \mu(B_n). \tag{23}$$

The right-hand is $o(n)$ as $n \to \infty$ since, by the entropy theorem, $\mu(G_n) \to 1$ as $n \to \infty$ and hence $\mu(B_n)$ vanish as $n \to \infty$.
We now estimate the sum over $G_n$. By (21) we have that

$$n(h - \epsilon) \leq -\log \mu(x_1^n) \leq n(h + \epsilon)$$

holds true for any $x_1^n \in G_n$, and hence so does the the sum

$$n(h - \epsilon) \sum_{G_n} \frac{\mu(x_1^n)}{n\mu(G_n)} \leq -\sum_{G_n} \frac{\mu(x_1^n)}{n\mu(G_n)} \log \mu(x_1^n) \leq \sum_{G_n} \frac{\mu(x_1^n)}{n\mu(G_n)} n(h + \epsilon)$$

leading to

$$h - \epsilon \leq -\frac{1}{n\mu(G_n)} \sum_{G_n} \mu(x_1^n) \log \mu(x_1^n) \leq h + \epsilon.$$

By the entropy rate theorem $\mu(G_n) \to 1$ as $n \to \infty$ and it follows that $-1/n \sum_{G_n} \mu(x_1^n) \log \mu(x_1^n)$ approaches $H(\mu)$, so the above inequality may in fact read

$$h - \epsilon \leq H(\mu) \leq h + \epsilon$$

which completes the proof of the case $h > 0$.

For the case $h = 0$ define $G_n = \{x_1^n \mid \mu(x_1^n) \leq 2^{-n(h+\epsilon)}\}$ and observe that the upper bounds in the proof still holds.

§117 **Example** (ENTROPY OF I.I.D. PART I). Let $\mu$ be an i.i.d. measure on $(A^\infty, \mathcal{F})$, that is the measure is defined as the product measure based on $\mu_1$ like this:

$$\mu_n(a_1^n) \stackrel{\text{def}}{=\!=} \mu_1(a_1)\mu_1(a_2)\cdots\mu_1(a_n).$$

By definition $\mu$ is independent and thus we have by observation §112 additivity of $H$. We find that

$$H_n(\mu_n) = H_1(\mu_1) + H_1(\mu_1) + \cdots + H_1(\mu_1) = nH_1(\mu_1)$$

and from that

$$H(\mu) = \lim_{n\to\infty} \frac{1}{n} nH_1(\mu_1) = -\sum_{a\in A} \mu(a) \log \mu(a).$$

§118 **Example** (ENTROPY OF I.I.D. PART II). Let $\mu$ be an i.i.d. measure. For any $a_1^n, b_1^m \in A^*$ we have $\mu\left(T^{-N}(\pi_n^{-1}(a_1^n)) \cap \pi_n^{-1}(b_1^m)\right) = \mu(\pi_n^{-1}(a_1^n))\mu(\pi_n^{-1}(b_1^m))$ for sufficient large values of $N$ (for example $N > m$). Because sets of the form $\pi_n^{-1}(a_1^n)$ generates $\mathcal{F}$ we have the property

$$\lim_{n\to\infty} \mu(T^{-n}A \cap B) = \mu(A)\mu(B), \quad \forall A, B \in \mathcal{F}$$

which is known as the *mixing condition*. This implies ergodicity, for if $T^{-1}A = A$ then we find $\mu(A \cap A) = \mu(A)\mu(A)$ which yields $\mu(A) = 1$ or $\mu(A) = 0$.
Now, for any sequence $\mathbf{x} \in \mathcal{T}(\mu)$ we may calculate the entropy $h$ as follows:

$$\begin{aligned} h = \lim_{n\to\infty} -\frac{1}{n} \log \mu(x_1^n) &= \lim_{n\to\infty} -\frac{\sum \log \mu(x_i)}{n} \\ &= \lim_{n\to\infty} -\sum_{a\in A} p_1(a|x_1^n) \log \mu(a) \\ &= -\sum_{a\in A} \mu(a) \log \mu(a) \end{aligned}$$

The equality in the second line is just a regrouping of the sum and the last equality is by the fact that we have $\lim_{n\to\infty} p_1(a|x_1^n) = p_1(a|\mathbf{x}) = \mu(a)$. Since $\mu(\mathcal{T}(\mu)) = 1$ we may say that the result holds almost surely.

§119 **Remark**. A theorem, which states a lower bound on the length of codewords and proves that the bound is attainable, is known as a *coding theorem*. Shannons theorem which we present now is

such a theorem. As seen from the proof (or simply by looking at the expression $\mathrm{E}_\mu(|\varphi_n(x_1^n)|)/n$ for a Shannon code $\varphi_n$) the celebrated theorem by Shannon is closely related to the definition of $\mathrm{H}$. A similar result to Shannons theorem which is connected to the entropy-rate, $\mathfrak{h}$, is presented further below.

§120 **Theorem** (SHANNON). *Let $\mu$ be a stationary measure on $A^\infty$, then*

$$\frac{\mathrm{E}_\mu(|\varphi_n(x_1^n)|)}{n} \geq \mathrm{H}(\mu), \tag{24}$$

*for any prefix code sequence $\{\varphi_n\}_{n\in\mathbb{N}}$ and any $n \in \mathbb{N}$. There exists a prefix code sequence such that*

$$\limsup_{n\to\infty} \frac{\mathrm{E}_\mu(|\varphi_n(x_1^n)|)}{n} = \mathrm{H}(\mu). \tag{25}$$

*For any prefix code sequence $\{\varphi_n\}_{n\in\mathbb{N}}$ with $|\varphi_n(x_1^n)| = \lceil -\log \mu(x_1^n)\rceil$ the equation (25) holds.*

§121 **Proof**: Note that the proof is first made for the case of prefix codes using Kraft's inequality, and then corollary §33 is employed to yield the result for all codes. Let $n \in \mathbb{N}$ and assume $\varphi_n$ is a prefix code. We have

$$\begin{aligned}
\mathrm{E}_\mu(|\varphi_n(x_1^n)|) &= \sum_{a_1^n\in A^n} |\varphi_n(a_1^n)|\mu(a_1^n) \\
&= \sum_{a_1^n\in A^n} \log \frac{2^{|\varphi_n(a_1^n)|}\mu(a_1^n)}{\mu(a_1^n)}\mu(a_1^n) \\
&= \sum_{a_1^n\in A^n} \mu(a_1^n)\log\frac{\mu(a_1^n)}{2^{-|\varphi_n(a_1^n)|}} - \sum_{a_1^n\in A^n}\mu(a_1^n)\log\mu(a_1^n) \\
&\geq \mathrm{H}_n(\mu)
\end{aligned}$$

with equality if and only if $\mu(a_1^n) = 2^{-|\varphi_n(a_1^n)|}$ for any $a_1^n \in A^n$. This proves the bound for prefix codes. By corollary §33 we may extend the result to any code sequence $\{\varphi_n\}_{n\in\mathbb{N}}$. For the last part of the theorem assume, according to Kraft's theorem (see theorem §25), that $|\varphi_n(x_1^n)| = \lceil -\log\mu(x_1^n)\rceil$. It follows that

$$\mathrm{E}_\mu(|\varphi_n(x_1^n)|) \leq \mathrm{E}_\mu(-\log\mu(x_1^n)+1) = \mathrm{H}_n(\mu)+1.$$

§122 **Theorem** (BARRONS LEMMA). *Let $\mu$ be a stationary, ergodic measure on $A^\infty$, then*

$$\liminf_{n\to\infty}\frac{|\varphi_n(x_1^n)|}{n} \geq \mathrm{H}(\mu)$$

*almost surely, for any prefix code sequence $\{\varphi_n\}_{n\in\mathbb{N}}$.*

§123 **Proof**: Define the set $B_n \overset{\text{def}}{=\!=} \{x_1^n \mid |\varphi_n(x_1^n)| \leq -\log\mu(x_1^n) - \epsilon_n\}$, where $\epsilon_n = 2\log n$. We see that $x_1^n$ is a member of $B_n$ if and only if $2^{-|\varphi_n(x_1^n)|}2^{-\epsilon_n} \geq \mu(x_1^n)$. The measure of $B_n$ is now easy to calculate:

$$\mu(B_n) = \sum_{x_1^n\in B_n}\mu(x_1^n) \leq 2^{-\log n^2}\sum_{x_1^n} 2^{-|\varphi_n(x_1^n)|} \leq \frac{1}{n^2}$$

where the last inequality is due to Kraft's inequality. By the Borel-Cantelli lemma we have $x_1^n \notin B_n$ eventually, almost surely. Thus the following inequality holds

$$|\varphi_n(x_1^n)| + \epsilon_n \geq -\log \mu(x_1^n),$$

eventually, almost surely. This leads to the desired result

$$\liminf_{n\to\infty} \frac{|\varphi_n(x_1^n)|}{n} \geq \liminf_{n\to\infty} \frac{-\log \mu(x_1^n)}{n} = H(\mu),$$

which holds almost surely. The last identity is due to theorem §108 and theorem §115.

§124 **Observation**. A code of the form introduced in theorem §120, that is, a code with $|\varphi_n(x_1^n)| = \lceil -\log \mu(x_1^n) \rceil$ is known as a *Shannon code* for $\mu$. It is almost trivial to see that the Shannon code for $\mu$ attains the entropy $h$ in the limit:

$$\limsup_{n\to\infty} \frac{|\varphi(x_1^n)|}{n} \leq \limsup_{n\to\infty} \frac{-\log \mu(x_1^n) + 1}{n} = h$$

The equality follows from the entropy-rate theorem (theorem §108).

§125 **Summary**. A link to the more traditional definition of entropy was provided by proving equivalence between the entropy-rate (the constant $h$ of the entropy-rate theorem) and the process-entropy (denoted $H(\mu)$. The theorem of Shannon is, of course, of immense importance and we will continue the general theme of coding theorems in the following section where we consider a generalization to Shannon's theorem.

## A coding theorem for the non-probabilistic case

§126 **Remark**. In the previous couple of sections we have worked with the probabilistic assumptions (beginning with the introduction of the set $\mathcal{S}$ of stochastic source instances) and seen a number of nice results which hold in this case. Yet our notion of source instances allow us to consider a much broader class of sequences, than just $\mathcal{S}$.

§127 **Definition** (HAUSDORFF DIMENSION). Let $S \subseteq A^\infty$ be a combinatorial source. We say that a finite or countable prefix set $C \subset A^*$ is a $n$-*cover* of $S$ if each element $c$ of $C$ has length at least $n$ and if for any $s \in S$ there exists a $c \in C$ such that $c$ is a prefix of $s$. Set $l_\alpha(S, n) = \inf_C \sum_{c\in C} |A|^{-|c|\alpha}$ where the infimum is over all $n$-covers of $S$ and let $l_\alpha(S) = \lim_{n\to\infty} l_\alpha(S, n)$. We then define

$$\dim_H(S) \stackrel{\text{def}}{=\!=} \sup\{\alpha \mid l_\alpha(S) = \infty\} = \inf\{\alpha \mid l_\alpha(S) = 0\}.$$

§128 **Observation**. We first note, that the limit involved in the definition of $l_\alpha(S)$ will always exists (within $[0, \infty]$) because of the fact that a $(n+1)$-cover of $S$ is also a $n$-cover, and hence the limit is over a non-decreasing, non-negative sequence. If $S \subseteq S'$ we note that any cover of $S'$ is also a cover of $S$ and hence $l_\alpha(S) \leq l_\alpha(S')$. One may check, that $l_\alpha(\cup S_i) \leq \sum l_\alpha(S_i)$ for any at most countable collection of sets $S_i$. These properties makes $l_\alpha$ an outer measure. For further details, see Appendix D. Also, if $\beta > \alpha$ we see that $\sum |A|^{-|c|\beta} \leq |A|^{-|n|(\beta-\alpha)} \sum |A|^{-|c|\alpha}$ and hence

$l_\beta(S,n) \leq |A|^{-|n|(\beta-\alpha)} l_\alpha(S,n)$ from which easily follows that $l_\alpha(S) < \infty$ then $l_\beta(S) = 0$ for any $\beta > \alpha$. The value of $\dim_H(S)$ as introduced is thus well defined. We note, that the definition of Hausdorff dimension given above is not identical to the classical definition. The definition given here is, however, compatible with the traditional one, in that it yields the same dimension. Again, further details can be found in Appendix D.

§129 **Example** (CANTORS SET). Let $A = \{0,1,2\}$ and define $S = \{x \in A^\infty \mid x \in \{0,2\}^\infty\}$. We now want to calculate the Hausdorff dimension of $S$. First we note that the set $C = \{x_1^n \mid x \in S\}$ is a $n$-cover of $S$ and hence, since the definition of $l_\alpha(S,n)$ involved the infimum over all coverings, $l_\alpha(S,n) \leq \sum_{c\in C} 3^{-n\alpha} = 2^n 3^{-n\alpha}$. It is also clear, that the value of $2^n/3^{n\alpha}$ will either tend to infinity (if $\alpha < 1/\log 3$) or to zero (if $\alpha > 1/\log 3$) as $n$ grows towards infinity. Thus we have shown that $\dim_H(S) \leq 1/\log 3$. We now define $d = 1/\log 3$ and set out to prove that indeed $\dim_H(S) = d$. We will do that by demonstrating that $l_d(S) = 1$, which implies that $l_\alpha(S) = \infty$ for any $\alpha < d$.
Let $n \in \mathbb{N}$ and let $C$ be any $n$-covering of $S$. We then have $\sum_{c\in C} |A|^{-|c|d} = \sum_{c\in C} 2^{-|c|}$. As we are interested in the infimum over all covers, we may assume any $c \in C$ is a member of $\{0,2\}^*$, since if it is not, we may construct the set $C\backslash\{c\}$ which will still be a cover, because no sequence in $S$ contains the symbol 1. Because $C$ is a prefix set over $A$ it is in particular a prefix set over $\{0,2\}$ and hence the inequality of Kraft yields $\sum_{c\in C} 2^{-|c|} \leq 1$. But $C$ is also a complete prefix set, for if not it would be possible to extend $C$ with a sequence $c \in \{0,2\}^*$ without breaking the prefix property. But that would imply that any sequence from $\{0,2\}^\infty$ which has $c$ as prefix cannot be in $S$, which is a contradiction. Hence $\sum_{c\in C} 2^{-|c|} = 1$ leading to $l_d(S) = 1$ and we are done.

§130 **Observation**. Assume for a moment that the Hausdorff dimension was defined using exactly-$n$-covers, that is, covers $C_n$ where each element of $C_n$ had length exactly $n$. In this case we see that $l_\alpha(S,n) = \inf_{C_n} |C_n||A|^{-n\alpha}$. It is easy to check, that the only parameter which affects the value of $|C_n||A|^{-n\alpha}$ is the *size* of the cover $C_n$ and that the infimum is attained using $C_n = \{x_1^n \mid x \in S\}$. The value of $l_\alpha(S,n)$ may now be given the interpretation as the number of possible prefixes of length $n$ that may arise from the combinatorial source $S$ (that number is $|C_n|$), divided by the number of possible words that can be made using $n\alpha$ letters (that number is of course $|A|^{n\alpha}$). Now let us consider this in terms of coding, more specifically, in terms of trying to represent each possible sequence from $S$ of length $n$ by using $n\alpha$ letters from $A$. If the value of $l_\alpha(S,n)$ is less than or equal to 1, it means that there is enough different sequences of $n\alpha$ letters from $A$ to represent all $n$-prefixes from $S$. The corresponding "dimension" is now seen to be the exact value of $\alpha$ for which $n\alpha$ letters is sufficient to encode the first $n$ letters of a sequence from the source $S$. Of course, in order for this to be true we must argue that the $l_\alpha(S)$ we have defined using only exactly-$n$-covers, behaves just like the real one (where the infimum is over *any* $n$-cover). That is, however, not true in general, but for now we are satisfied that the connection between coding and Hausdorff dimensions is not something taken out of thin air. It is certainly not a coincidence that we employ the Kraft inequality which speaks about the tug-of-war between codeword-length and number of codewords. This idea is also present in dimension theory where the tug-of-war, loosely speaking, is between not having too small cover-(balls/cubes/etc.) yet capturing enough details about the geometric figure one is trying to cover.
As an aside we note, that by tweaking things a bit one can still get a usable dimension using limited freedom in selecting covers, as we did above. The dimensions one can get by doing this will unfortunately not be defined for any subsets of $A^\infty$ (like the Hausdorff dimension is), but for those subsets where it is defined one can show that it has relations to the Hausdorff

dimension. The dimension, which will emerge by the above construction, is known as the *box-counting dimension*. In general this dimension is larger than the Hausdorff dimension, but for reasonable regular (smooth) sets the box-counting dimension and the Hausdorff dimension will be equal. For further details we refer to the litterature (e.g. [Falconer, 1990]).

§131 **Theorem**. *Let $S \subseteq A^\infty$ be a combinatorial source over an alphabet $A$ and let $B = \{0,1\}$, then for any code $\varphi_\infty : A^\infty \to B^\infty$, the following hold*

$$c(\varphi_\infty, S) \geq \dim_H(S) \log |A| \tag{26}$$

*with equality for at least one code.*

§132 **Proof**: We first construct a code, $\varphi_\infty$, for which (26) holds with equality. The idea in this part of the proof is to grow[6] a prefix set into an increasingly more detailed covering of the set $S$. Let $\alpha_i \stackrel{\text{def}}{=\!=} \dim_H(S) + 2^{-i}$ for all $i \in \mathbb{N}$. Let $C_1$ be a 1-cover of $S$, chosen so that

$$\sum_{c \in C_1} |A|^{-|c|\alpha_1} < 1. \tag{27}$$

The existence of $C_1$ is ensured by the definition of the Hausdorff dimension since $\alpha_1 > \dim_H(S)$. We now describe the construction of $C_{i+1}$ based on $C_i$. For each $c \in C_i$ let $S(c)$ be the sequences from the source $S$ having $c$ as prefix. That is, $S(c) \stackrel{\text{def}}{=\!=} \{x \mid x_1^{|c|} = c\}$. We then pick a $(|c|(i+1)^2)$-cover $C(c)$ of $S(c)$. Since $\dim_H S(c) \leq \dim_H S < \alpha_{i+1}$ we may choose $C(c)$ in such a way that

$$\sum_{d \in C(c)} |A|^{-|d|\alpha_{i+1}} < 1 \tag{28}$$

holds. Setting $C_{i+1} \stackrel{\text{def}}{=\!=} \cup_{c \in C_i} C(c)$ yields a cover of $S$. The sequence $C_1, C_2, \ldots$ now induces a parsing of any $x \in S$, namely by $x = c_1 c_2 \cdots$ where $c_1 \in C_1$, $c_1 c_2 \in C_2$ and so on. By the construction we have

$$|c_i| \geq |c_1 \cdots c_{i-1}| i^2. \tag{29}$$

For any $i$ we may rewrite (28) to get $\sum_{d \in C(c)} 2^{-|d|\alpha_{i+1} \log |A|} < 1$ holding for any $c \in C_{i-1}$. It the follows (by Kraft's result, see §25) that there is a prefix code $\varphi_{i,c} : C_i(c) \to B^*$ which maps the words of $C_i$ having $c$ as prefix into binary sequences. Indeed, the codeword lengths of this code is given by

$$|\varphi_{i,c}(d)| = \lceil |d|\alpha_i \log |A| \rceil. \tag{30}$$

Note, that even though the code is defined only for sequences of $C_i$ having $c$ as prefix, it is the length of the *whole* sequence $d \in C_i$ which is used in the expression of the length. We write $\varphi_1$ for the code on $C_1$ induced in a similar way by (27). We now construct the code $\varphi_\infty : A^\infty \to B^\infty$ in the following way: Let $x \in A^\infty$ and write $x = c_1 c_2 \cdots$ as described above. Then define

$$\varphi_\infty(x) \stackrel{\text{def}}{=\!=} \varphi_1(c_1)\varphi_{2,c_1}(c_1c_2)\varphi_{3,c_1c_2}(c_1c_2c_3)\cdots$$

and $n_i \stackrel{\text{def}}{=\!=} |c_1 \cdots c_i|$. It is clear, that

$$c(\varphi_\infty, x, n_i) \leq |\varphi_1(c_1) \cdots \varphi_{i,c_1 \cdots c_{i-1}}(c_1 \cdots c_i)|$$

[6] Those readers who use *trees* as mental pictures for prefix sets, will find the term *grow* makes a lot of sense. Others will just have to study the proof!

We can now make an estimation of the performance of the code:

$$\begin{aligned}
c(\varphi_\infty, x) &= \liminf_{n\to\infty} n^{-1} c(\varphi_\infty, x, n) \\
&\le \liminf_{i\to\infty} n_i^{-1} c(\varphi_\infty, x, |c_1 \cdots c_i|) \\
&\overset{\text{by (30)}}{\le} \liminf_{i\to\infty} \log|A| \frac{|c_1|\alpha_1 + |c_1c_2|\alpha_2 + \cdots + |c_1\cdots c_i|\alpha_i + i}{n_i} \\
&\le \liminf_{i\to\infty} \left( \log|A| \frac{|c_1\cdots c_i|\alpha_i}{n_i} + \log|A|\alpha_1 \frac{|c_1| + \cdots + |c_1\cdots c_{i-1}| + i}{n_i} \right) \\
&\le \liminf_{i\to\infty} \left( \log|A|\alpha_i + \log|A|\alpha_1 \frac{i|c_1\cdots c_{i-1}|}{|c_1\cdots c_i|} \right) \\
&\overset{\text{by (29)}}{\le} \liminf_{i\to\infty} \left( \log|A|\alpha_i + \log|A|\alpha_1 i \frac{1}{i^2} \right) \\
&= \liminf_{i\to\infty} \log|A|\alpha_i = \dim_H(S) \log|A|
\end{aligned}$$

This concludes the first part of the proof. Conversely, let $\varphi_\infty$ be any code on $S$ and let $\delta > 0$ and $n \in \mathbb{N}$ be given. By the definition of the cost (see (6) in definition §43), we may choose, for any $x \in S$, an integer $k(x) \ge n$ so that

$$c(\varphi_\infty, x, k(x))/k(x) \le c(\varphi_\infty, S) + \delta$$

Define $L \overset{\text{def}}{=\!=} \{x_1^{k(x)} \mid x \in S\}$ and make it into a prefix set by the following procedure. For each $m \in \mathbb{N}$ go through all $l \in L$ with $|l| = m$ and remove $l$ from $L$ if it is a prefix of another element of $L$. By this procedure a prefix set $\tilde{L} \subseteq L$ is created, and $\tilde{L}$ is a cover of $S$. Since $L$ is clearly a $n$-cover of $S$ and we have only removed elements which are prefixes of other elements from $L$ we conclude, that $\tilde{L}$ is also a $n$-cover of $S$. Now, for any $l \in \tilde{L}$, pick $x$ so that $l$ is a prefix of $x$ and set

$$\psi(l) \overset{\text{def}}{=\!=} \varphi_\infty(x)_1^{c(\varphi_\infty, x, k(x))}$$

that is, $\psi(l)$ is the prefix of $\varphi_\infty(x)$ needed in order to be able to decode the first $k(x)$ symbols of $x$. We now have

$$|\psi(l)|/|l| \le c(\varphi_\infty, S) + \delta \tag{31}$$

In fact $\psi$ is a prefix code, for if $l_1, l_2 \in \tilde{L}$ and $\psi(l_1)$ is a prefix of $\psi(l_2)$ then $\varphi_\infty^{-1}(\psi(l_1)) \supseteq \varphi_\infty^{-1}(\psi(l_2))$. We have $\varphi_\infty^{-1}(\psi(l_1)) = l_1 A^\infty \cap S$ and similarly $\varphi_\infty^{-1}(\psi(l_2) = l_2 A^\infty \cap S$. Because $\tilde{L}$ is a prefix set, we conclude that $l_1 = l_2$. We may now estimate

$$\sum_{l\in\tilde{L}} |A|^{-|l|(c(\varphi_\infty,S)+\delta)/\log|A|} \le \sum_{l\in\tilde{L}} |A|^{-|\psi(l)|/\log|A|} = \sum_{l\in\tilde{L}} 2^{-|\psi(l)|} \le 1$$

where the first inequality is due to (31) and the second is by the prefix property of $\psi$ and the Kraft inequality. We conclude, that for any $n \in \mathbb{N}$ we have constructed a $n$-cover, namely $\tilde{L}$, of $S$ such that $\sum_{l\in\tilde{L}} |A|^{-|l|(c(\varphi_\infty,S)+\delta)/\log|A|} \le 1$. Setting $\alpha = (c(\varphi_\infty, S) + \delta)/\log|A|$ we have, $\inf_C \sum_{c\in C} |A|^{-|c|\alpha} \le 1$ which in turn implies that $\dim_H(S) \le \alpha$ for all values of $\delta$ leading to the desired result,

$$\dim_H(S) \log|A| \le c(\varphi_\infty, S).$$

This concludes the proof.

§133 **Theorem**. *For any stationary ergodic measure $\mu$ over $A^\infty$ the following holds:*

$$\log|A|\dim_H(\mathcal{T}(\mu)) = H(\mu).$$

§134 **Proof**: We calculate the Hausdorff dimension of the set $\mathcal{T}(\mu)$. Write $h = H(\mu)$ and observe, that by the entropy-rate theorem we have $\lim_{n\to\infty} -\log\mu(x_1^n)/n = h$ for any $x \in \mathcal{T}(\mu)$. Let $\epsilon > 0$ and define

$$B(N) \stackrel{\text{def}}{=\!=} \{x \in \mathcal{T}(\mu) \mid 2^{-n(h+\epsilon)} \le \mu(x_1^n) \le 2^{-n(h-\epsilon)} \text{ for all } n \ge N\}$$

As seen directly by this definition $B(N) \subseteq B(N+1)$ and for any $x \in \mathcal{T}(\mu)$ there is a $N \in \mathbb{N}$ such that $x \in B(N)$. Thus $\mathcal{T}(\mu) = \cup_{N\in\mathbb{N}} B(N)$. Fix $N \in \mathbb{N}$ and observe, that $C_n \stackrel{\text{def}}{=\!=} \{x_1^n \mid x \in B(N)\}$ is a $n$-cover of $B(N)$. Also, for $n \ge N$ we have for any $c \in C_n$ the bound $\mu(c) \ge 2^{-n(h+\epsilon)}$. Therefore the size of $C_n$ is at most $2^{n(h+\epsilon)}$ and we have

$$\sum_{c\in C_n} |A|^{-|c|(h+\epsilon)/\log|A|} = \sum_{c\in C_n} 2^{-|c|(h+\epsilon)} \le 2^{n(h+\epsilon)} 2^{-n(h+\epsilon)} = 1$$

so we obtain $l_{(h+\epsilon)/\log|A|}(B(N)) < \infty$ which in turn implies that $\dim_H B(N) \le h$. Since this holds independently of our choice of $N$, we conclude, using the rule $\dim_H \mathcal{T}(\mu) = \sup_{N\in\mathbb{N}} \dim_H B(N)$ (see lemma §288) that $\log|A|\dim_H \mathcal{T}(\mu) \le h$.
On the other hand, for each $N \in \mathbb{N}$ and each $n$-cover $C$ of $B(N)$ where $n \ge N$ we have $\mu(c) \le 2^{-|c|(h-\epsilon)}$. Hence

$$\mu(C) = \sum_{c\in C} \mu(c) \le \sum_{c\in C} 2^{-|c|(h-\epsilon)} = \sum_{c\in C} |A|^{-|c|(h-\epsilon)/\log|A|}$$

so that taking infimum over all covers yields $l_{(h-\epsilon)/\log|A|}(B(N), n) \ge \inf_C \mu(C) = \mu(B(N)) > 0$. In the limit this leads to $\log|A|\dim_H B(N) \ge h$ which implies $\log|A|\dim_H \mathcal{T}(\mu) \ge h$.

§135 **Observation**. Let $\mu$ be an ergodic measure and let $\{\varphi_m\}_{m\in\mathbb{N}}$ be a corresponding Shannon code. We define, for each $m \in \mathbb{N}$, the following code on $A^\infty$:

$$\varphi_{\infty,m} \stackrel{\text{def}}{=\!=} \varphi_m(x_1^m)\varphi_m(x_{m+1}^{2m})\cdots .$$

It is now possible to give an upper bound on the cost of $\varphi_{\infty,m}$. Let $p$ be the smallest number so that $pm \ge n$, then $c(\varphi_{\infty,m}, x, n) \le |\varphi_m(x_1^m)| + \cdots + |\varphi_m(x_{mp-m+1}^{mp})|$, giving us

$$c(\varphi_{\infty,m}, x) \le \lim_{i\to\infty} \frac{1}{im} \sum_{j=0}^{i-1} |\varphi_m(\pi_m(T^{jm}x))|.$$

Here we may use a true limit in place of a lower limit since the code is of bounded delay. We conclude, by the ergodic theorem, that $c(\varphi_{\infty,m}, x) \le E_\mu|\varphi_m(x_1^m)|/m$ almost surely. By Shannon's theorem, §120, we find $c(\varphi_{\infty,m}, x) \le H(\mu) + \epsilon_m$ almost surely, where $\epsilon_m \to 0$ as $m \to \infty$.

§136 **Observation**. We now check that the map $c(\varphi_\infty, \cdot) : (A^\infty, \mathcal{F}) \to \mathbb{R}$ is $\mu$-measurable under reasonable constraints on the code $\varphi_\infty$. Recall, that the definition of the cost function (see §43) is made by defining

$$c(\varphi_\infty, x, n) = \min\{k \mid \varphi_\infty^{-1}(\varphi_\infty(x)_1^k B^\infty) \subseteq x_1^n A^\infty\}$$

and then setting

$$c(\varphi_\infty, \mathbf{x}) = \liminf_{n\to\infty} \frac{c(\varphi_\infty, \mathbf{x}, n)}{n}.$$

We first consider $c(\varphi_\infty, \cdot, n) : (A^\infty, \mathcal{F}) \to (\mathbb{R}, \mathbf{B})$, and check that this map is measurable. It is enough to check that any interval of the form $]-\infty, a] \subset \mathbb{R}$ (these intervals spans $\mathbf{B}$) is carried back into a $\mu$-measurable set by the map $c^{-1}(\varphi_\infty, \cdot, n)$. Let $]-\infty, a] \subset \mathbb{R}$ be given. The set in question is now given as

$$c^{-1}(\varphi_\infty, \cdot, n)(]-\infty, a]) = \left\{ \mathbf{x} \in A^\infty \mid \min\{k : \varphi_\infty^{-1}(\varphi_\infty(\mathbf{x})_1^k B^\infty) \subseteq x_1^n A^\infty\} \leq a \right\}.$$

The above set is the infinite sequences which lets us decode the $n$ first symbols by looking at no more than $\lfloor a \rfloor$ symbols of the encoded message. For *many* (some would say all sane) codes this will only depend on the prefix of the sequences $\mathbf{x}$. For example, assume we have a simple code $\varphi_1(a) = 0$, $\varphi_1(b) = 10$ and $\varphi_1(c) = 11$ and form $\varphi_\infty(\mathbf{x}) = \varphi_1(x_1)\varphi_1(x_2)\varphi_1(x_3)\cdots$. It is now pretty clear, that if we want to decode two message symbols by looking at no more than three code symbols, the (infinite) message must have either `a`, `ba` or `ca` as prefix. Thus for this particular code $\varphi_\infty$ we have

$$c^{-1}(\varphi_\infty, \cdot, 2)(]-\infty, 3]) = \pi_1^{-1}(a) \cup \pi_2^{-1}(ba) \cup \pi_2^{-1}(ca)$$

In general, for any value of $n$ we have $c^{-1}(\varphi_\infty, \cdot, n)([-\infty, a]) = \emptyset$ if $a < 1$ and likewise we have $c^{-1}(\varphi_\infty, \cdot, n)([-\infty, a]) = A^\infty$ if $a > N$ for a suitable $N = N(n)$ (more precisely, $N = n \times \max\{\varphi_1(a) \mid a \in A\} = 2n$.) Guided by this example we arrive at the following conclusion: For any code $\varphi_\infty$ on $A^\infty$ which is created as a concatenation of a block-code, that is, $\varphi_\infty(\mathbf{x}) = \varphi_m(x_1^m)\varphi_m(x_{m+1}^{2m})\cdots$, the map $c(\varphi_\infty, \cdot, n)$ is $\mu$-measurable for any value of $n$. Hence so is also $c(\varphi_\infty, \cdot)$, since composition of measurable maps produces a measurable map and the limit of measurable maps is again a measurable map.

§137 **Example**. A natural question to ask is if there are any codes $\varphi_\infty$ for which the cost function $c(\varphi_\infty, \cdot)$ is *not* measurable. Let $A = B = \{0, 1\}$ and write $\overline{0} = 1$ and $\overline{1} = 0$. Let $C \subset A^\infty$ such that $C \notin \mathcal{F}$. Define $\varphi_\infty$ as follows:

$$\varphi_\infty(\mathbf{x}) = \begin{cases} 0\mathbf{x} & \text{if } \mathbf{x} \in C \\ 1\mathbf{x} & \text{if } \mathbf{x} \notin C \text{ and } \liminf_{n\to\infty} f(0|x_1^n)/n > 0.5 \\ 1\overline{\mathbf{x}} & \text{otherwise} \end{cases}$$

where $f(0|x_1^n)$ is the frequency count of 0's in the sequence $x_1^n$. It is now clear, that $c(\varphi_\infty, \mathbf{x}, n) = \infty$ for any $\mathbf{x} \notin C$ and hence $c^{-1}(\varphi_\infty, \cdot, n)(]-\infty, a])$ for $a > n$ equals the set $C$ which is not measurable.

§138 **Theorem**. *Let $\mu$ be an ergodic measure on $A^\infty$, then*

$$\inf_{\varphi_\infty} \int_{A^\infty} c(\varphi_\infty, \mathbf{x}) d\mu(\mathbf{x}) = H(\mu)$$

*where the infimum is over all codes $\varphi_\infty : A^\infty \to B^\infty$ such that $c(\varphi_\infty, \cdot) : A^\infty \to \mathbb{R}$ is $\mu$-measurable.*

§139 **Proof**: Based on the result of observation §135 we may write $\int_{A^\infty} c(\varphi_{\infty,m}, \mathbf{x}) d\mu(x) \leq H(\mu) + \epsilon_m$

with $\epsilon_m \to 0$ as $m \to \infty$, and hence

$$\inf_{\varphi_\infty} \int_{A^\infty} c(\varphi_\infty, x) d\mu(x) \leq H(\mu).$$

On the other hand, assume

$$\inf_{\varphi_\infty} \int_{A^\infty} c(\varphi_\infty, x) d\mu(x) < H(\mu)$$

then there exists a code $\varphi_\infty$ so that $\int_{A^\infty} c(\varphi_\infty, x) d\mu(x) < H(\mu)$. And thus for some subset $U \subseteq \mathcal{T}(\mu)$ with $\mu(U) > 0$ we have $\sup_{x \in U} c(\varphi_\infty, x) < H(\mu)$ but this is impossible by theorem §131 since $\log |A| \dim_H(U) = H(\mu)$.

§140 **Summary**. We have generalized the coding theorem of Shannon by proving that the Hausdorff dimension provides an attainable lower bound on the compression that we can achieve, and that the Hausdorff dimension agrees with the entropy of stationary ergodic sources. The use of Hausdorff dimensions in information theory is quite new, and we believe that it would be interesting to pursue more parallels between the field of information theory and the field of geometric measure theory.

## The source concept revisited

§141 **Remark**. As stated in the introduction, the goal of this thesis is not to engage in a philosophical debate on the foundations of source coding, but rather to present a number of results selected by their relation to the foundational aspects of source coding. Hopefully the reader has, at this point, seen results and methods showing the source concept from new angles. In this section we take a moment to reflect on the source (coding) concept, before we move on to the second half of the thesis.

§142 **Observation** (SOURCE CONCEPTS). During this chapter we have considered two different views on the concept of sources, namely the *combinatorial source* and the *stationary ergodic source*. Somewhat unconventionally both views was constructed in a two-step fashion: The first step was to simply focus on a particular infinite sequence, known as the *source instances*. The second step was to group together a variety of source instances that we considered similar in some sense, either by specifying directly a subset of $A^\infty$ (leading to the *combinatorial source*) or by using a stationary ergodic measure to describe the limiting statistical properties of the source instances (leading to the *stationary ergodic source*.) The stationary ergodic point of view is by far the most commonly encountered (indeed it contains a multitude of important subclasses such as the i.i.d. measure, the Markov measures and so forth), however, as we have seen, a stationary ergodic measure is essentially just a nice way of specifying a class of source instances. Thus any stationary ergodic measure $\mu$ may be thought of as the combinatorial source consisting of $\mathcal{T}(\mu)$, the frequency typical sequences for $\mu$.

§143 **Example**. This example illustrates the idea that stochastic constraints to a large degree are transferable to combinatorial sources. Consider the well known class of finite memory Markov measures, that is, probability measures for which there is a number $n$, known as the *memory-size*, such that

$$\frac{\mu(a_1^m b)}{\mu(a_1^m)} = \frac{\mu(a_{m-n+1}^m b)}{\mu(a_{m-n+1}^m)}$$

for any $m \geq n$, any $a_1^m \in A^m$ and any $b \in A$. Clearly, in the case $n = 0$ we have $\mu(a_1^m b) = \mu(b)\mu(a_1^m)/\mu(\lambda) = \mu(b)\mu(a_1^m)$ that is, the measure is i.i.d. As we saw in example §118 any i.i.d. measure is ergodic, and one may check that the same hold for any Markov measure of finite memory. We can easily translate this concept into a set $\mathcal{S}_{M(n)} \subseteq \mathcal{S}$. A sequence $\mathbf{x}$ is a member of this set if and only if the limiting relative probabilities (definition §62) behave like this: $p_{m+1}(a_1^m b|\mathbf{x})/p_m(a_1^m|\mathbf{x}) = p_{n+1}(a_{m-n+1}^m b|\mathbf{x})/p_n(a_{m-n+1}^m|\mathbf{x})$ for all cases of $m \geq n$, $a_1^m \in A^m$ and $b \in A$.

§144 **Remark**. Although not the common case, in the litterature we do meet the source instance or infinite sequence as a starting point for studies. In ergodic theory the term *sample path* is used for a infinite sequence of outcomes chosen under the law defined by some ergodic measure $\mu$. Other works study source instances chosen freely from $A^\infty$. Since no code exists which can encode such sequences (or, in the finite case, the set of sequences $A^n$) into a shorter representation, a different approach must be taken to ensure an interesting problem. One such method is to put constraints on the *codes* instead of on the source instances. That is, to any source instance, $\mathbf{x} \in A^\infty$, we consider the best code chosen from some limited class of codes. This way of posing the problem is usually known as the *individual setup*, and in [Merhav and Feder, 1998] a study of the individual setup as compared to the stochastic setup is conducted.

§145 **Observation**. As example §143 shows, there is nothing profound about the idea that we may consider both combinatorial sources and stationary ergodic sources as based on source instances. Still, it is beneficial to have the various source concepts in a common framework. In the table below, we summarize the source concepts we have been looking at so far, as well as a few key point regarding how the sources are defined. In addition we mention how the—for lack of a better word—"complexity" of the source may be defined.

| | Combinatorial setup | Stochastic setup | Complexity measure |
|---|---|---|---|
| Arbitrary | $S \subseteq A^\infty$ | — | $\dim_H(S)$ |
| Stationary | — | $\mu \in \mathcal{M}_s(A^\infty)$, $\mu = \int \mu_e$ | $H(\mu) \neq \dim_H(\mathcal{T}(\mu))$ |
| Ergodic | $[\mathbf{x}] \subseteq \mathcal{S}$ | $\mu \in \mathcal{M}_e(A^\infty)$, $\mathcal{T}(\mu_\mathbf{x}) = [\mathbf{x}]$ | $\dim_H(\mathcal{T}(\mu)) = H(\mu) = h$ |

We will comment on each line in the table.
*Arbitrary:* When no assumptions about the source instances are made, there is no possible stochastic interpretation. The relevant complexity measure is Hausdorff dimension, yet we could also consider for example the Lempel-Ziv complexity (see [Ziv and Lempel, 1978] as well as the next chapter), and the Kolmogorov complexity (see [Li and Vitanyi, 1993]) of a sequence.
*Stationary:* A possible combinatorial interpretation of a stationary, but not ergodic, measure in the individual setup would be the set $\cup[\mathbf{x}]$ where the union is taken over those ergodic classes we get from the ergodic decomposition theorem. However, we would also need a *distribution* on the set of ergodic frequency equivalence classes of $A^\infty$. In short, the stationary situation is not easily captured in a combinatorial setup. The notation $\mu = \int \mu_e$ is supposed to be a reminder about the ergodic decomposition theorem §82. The entropy $H$ is defined (see remark §111) for any stationary measure, but the Hausdorff dimension of the typical sequences is just the supremum over the dimensions of the ergodic classes. Thus the dimension is unable to "see" the underlying distribution of the ergodic components.
*Ergodic:* In the ergodic case, the combinatorial interpretation is a subset of source instances

of equal statistical properties. These subsets all live within the set $\mathcal{S}$ which, as we have seen, in a sense contains the complete stochastic setup. It is interesting to note, that if we identify an ergodic measure $\mu$ with its class of typical sequences, $\mathcal{T}(\mu) = [\mathbf{x}]$, where $\mathbf{x}$ is any $\mu$-typical sequence, we find that the entropy $H$ can be considered as defined on subsets (of the form $[\mathbf{x}]$) of $A^\infty$. That is, the entropy is a quantity attached to a subset of sequences of $A^\infty$ rather than to a stationary ergodic measure. Another way of measuring the size or complexity of a subset of $A^\infty$ is to use the Hausdorff dimension, and indeed in the case of sets of the type $[\mathbf{x}]$ the Hausdorff dimension and the entropy agrees (we have omitted the scaling factor $\log|A|$.) The quantity $h$ is of course the entropy as defined (in almost surely sense) by the entropy-rate theorem.

## Source coding

§146 **Remark**. Having established coding-theorems (theorem §120 and theorem §131) for both the probabilistic and non-probabilistic case, our work might seem finished. However, the task of actually putting these theorems to work on practical problems is not trivial. In this section we survey some of the work done in this area which is known as *source coding*, leading up to the next chapter where we present in detail some of these practical codes.

§147 **Observation** (SHANNON AND HUFFMANN CODES). From a theoretical point of view the coding problem for stationary ergodic sources may seem solved: The theorem of Shannon (theorem §120) instructs us in how to choose the lengths of the codewords representing the blocks of data $x_1^n$, leading to the Shannon code, $\varphi_n^S$, for which $|\varphi_n^S(x_1^n)| = \lceil -\log \mu(x_1^n)\rceil$. (By Kraft's theorem (theorem §25) we know that a prefix code with these code lengths does indeed exist.) We get the following bound on the code lengths,

$$H_n(\mu) \leq E_\mu|\varphi_n(x_1^n)| \leq H_n(\mu) + 1 \tag{32}$$

where the first inequality is by Shannons theorem and the second is by straightforward calculation (see proof §121). However, we still need to devise an algorithm to construct the code inside a computer. This was done in [Huffmann, 1952] where an algorithm for constructing the Huffmann code was described. This code is not precisely the same as the Shannon code; in fact it might be considered slightly better: it achieves the minimum possible expected code length, that is, letting $\varphi_n^H$ denote the Huffmann code on $A^n$ we have $E_\mu(\varphi_n^H(x_1^n)) = \inf_{\varphi_n} E_\mu(\varphi_n(x_1^n))$ where the infimum is taken over any $n$-code on $A^n$. For example, if our alphabet is $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and we consider the i.i.d. measure where $\mu(\mathtt{a}) = \mu(\mathtt{b}) = \mu(\mathtt{c}) = 1/3$, the Shannon code would pick codewords of length $\lceil -\log 1/3\rceil = 2$ for all symbols. In the same situation the Huffmann algorithm would pick two codewords of length 2 and one codeword of length 1. Still the Huffmann code is subject to the same performance bound, (32), as the Shannon code.

§148 **Observation**. It should be noted, however, that both the Shannon code and the Huffmann code are quite impractical because we need to somehow keep track (or even store) the $|A|^n$ different codewords which is difficult for large values of $n$. If $n = 256$ and $|A| = 2$, our codebook would contain a huge number of words, namely $2^{256}$. Encoding messages of of length 256 or longer messages using a block-size of 256 is not at all unrealistic. Storing $2^{256}$ codewords, however, is definitely unrealistic. The number is truly astronomical, in the sense that it is close to the number of atoms in the visible universe! On the other hand, using a small $n$ is not attractive

either. One reason for this is that the constraint that a codeword need to have integer length becomes more visible then. As an extreme we may consider encoding output from an binary, i.i.d. source with the measure $\mu(0) = 1 - \epsilon$ for some small $\epsilon > 0$ using the Shannon code $\varphi_1$. We will have $|\varphi_1(0)| = |\varphi_1(1)| = 1$ and hence no compression at all even though the source has very low entropy. This also shows that the upper bound in (32) is sharp. Another reason that we want $n$ to have a large value is that we need to capture conditionals in the measure, for example if the measure is a Markov measure of order $m$. Both of these reasons remind us that we must remember that $E_\mu|\varphi_m(x_1^m)|/m$ is only *asymptotically* close to $H(\mu)$.

§149 **Observation** (ARITHMETIC CODING). In order to overcome the problems of Huffmann coding and Shannon coding as described above, the arithmetic code was created by Rissanen and others (see §158 for historical facts). We will not describe the details of the arithmetic code, but we will point out a number of fundamental differences when compared to the Huffmann code.
*Huffmann:* When employing the Huffmann algorithm to encode a message $x_1^n$, we must supply the corresponding measures $\mu_n = \mu \circ \pi_n^{-1}$ in the form of $|A|^n$ positive numbers which sum to 1. As a result we get a map, $\varphi_n^H : A^n \to B^*$, whose representation in memory is quite big; its size is exponential in $n$. We then map the message $x_1^n$ directly into the codeword $\varphi_n^H(x_1^n)$ simply by looking up the codeword for $x_1^n$ in the codebook.
*Arithmetic:* When using the arithmetic code we feed the source message symbolwise to the encoder. In addition, together with each symbol, we must supply a probability vector over $A$. If the source is i.i.d. we would use $(\mu(a))_{a\in A}$; if the structure of $\mu$ is more complicated we would use $(\mu(x_1^i a)/\mu(x_1^i))_{a\in A}$ when encoding the $i+1$'th symbol. In general any assignment of positive values to the elements of $A$ such that the numbers sum to 1 will work, hence we use the term *codespace assignment* for the process of selecting the values assigned to the symbols. The arithmetic code will then produce the codeword using memory linear in $n$.
These black-box presentations of the codes highlights some important differences between the two codes. First of all, the Huffmann code create an immense number of codewords and even though only one is used (namely $\varphi_n^H(x_1^n)$) all of them have to be calculated and stored in memory. The storage needed for this is exponential in the block-size $n$. The arithmetic code only calculates the one codeword that is needed and hence uses only space proportional to $n$. Secondly, by the nature of the Huffmann algorithm, the measure $\mu$ upon which the code is constructed must be known beforehand. In many situations (as we will shortly see) we have no knowledge of $\mu$, but need to estimate or approximate it. We would like to be able to do this as we are doing the actual coding and thus avoid two-pass methods. The arithmetic code enables exactly this: we need only supply the conditionals of $\mu$ and thus the method lends itself very well to adaptive coding. It is also important to note that the arithmetic code is sequential, that is, the codeword may be constructed sequentially by taking into account a growing part of the message[7], see also observation §49. The performance of the arithmetic code may be expressed in the following bound

$$|\varphi_n^A(x_1^n)| \leq -\log \mu(x_1^n) + c$$

where $c$ is a constant depending on the implementation of the arithmetic code. From this follows (by Shannons theorem, §120) that $\lim_{n\to\infty} E_\mu|\varphi_n^A(x_1^n)|/n = H(\mu)$.

§150 **Remark** (UNIVERSAL CODING). In universal coding we ask for a code such that not only does $E_\mu|\varphi_n(x_1^n)|/n$ approach the entropy in the limit, but furthermore we want $\varphi_n$ to be independent

[7] ... or, to put it short, "the codeword of a prefix is a prefix of a the codeword."

of the particular source. This approach was forwarded in [Davisson, 1973] where the concepts of *weak universality* and *strong universality* were introduced.

§151 **Strong Universal Coding**. Loosely speaking strong universal coding is attained when the difference between the expected code length and the entropy converges to 0 *uniformly* on some class $M \subset \mathcal{M}_e$. That is,

$$\lim_{n\to\infty} \sup_{\mu\in M} \{\frac{E_\mu|\varphi_n(x_1^n)| - H_n(\mu_n)}{n}\} = 0.$$

The difference between expected code length and entropy is also known as *redundancy* and we write $R_n(\mu, \varphi_n) \stackrel{\text{def}}{=\!=} (E_\mu|\varphi_n(x_1^n)| - H_n(\mu))/n$. The two most important results about strong universal coding is firstly, the redundancy-capacity theorem (see e.g. [Haussler, 1997] for the most general treatment) which links the existence of strong universal codes to the asymptotic behavior of the capacity of a discrete memoryless channel and secondly, the so called converse theorems in [Rissanen, 1984], [Merhav and Feder, 1995] and [Feder and Merhav, 1996] which state that in a sense almost all sources give rise to equally high redundancy thus defending the rather pessimistic use of $\sup_{\mu\in M}$ in the strong universal setup. Of interest is also the study of fastest possible convergence to zero of the upper bound on redundancy. As an example, in [Davisson, 1983] is shown that for the class of Markov sources of memory no more than $m$, there is a code $\{\varphi_n\}_{n\in\mathbb{N}}$ so that $\sup_{\mu\in M} R_n(\mu, \varphi_n) \simeq |A|^m(|A| - 1)(\log n/2n) + c$. Finally, we should note that there are classes of sources for which there are *no* strong universal codes. For instance, this is the case for the stationary ergodic sources, $\mathcal{M}_e(A^\infty)$.

§152 **Weak Universal Coding**. Correspondingly, a weak universal code is one for which the difference between the expected code length and the entropy converges to 0 *pointwise* on some class $M \subseteq \mathcal{M}_e$. That is,

$$\lim_{n\to\infty} \frac{E_\mu|\varphi_n(x_1^n)| - H_n(\mu_n)}{n} = 0, \quad \text{for any } \mu \in M.$$

Interesting results on weak universal coding includes several proofs of the existence of such codes on $\mathcal{M}_e(A^\infty)$, e.g. [Davisson, 1973] and [Ziv and Lempel, 1978]. Noteworthy is also the second order analysis of the convergence of the redundancy: The definition of weak universal coding might be expressed as $R_n(\mu, \varphi_n) = o(1)$ for any $\mu \in \mathcal{M}_e$. Checking this for some code is known as *first order analysis*. In [Shields, 1993] it is shown however, that we cannot in general (for $\mu \in \mathcal{M}_e$), say anything about the speed of the convergence[8]. For certain subclasses $M \subset \mathcal{M}_e(A^\infty)$, however, we do have knowledge of the convergence speed. For example the Lempel-Ziv performs like this, $R_n(\mu, \varphi_n^{LZ}) = O(\log\log n/\log n)$ on the class of finite memory Markov sources (see [Plotnik et al., 1992].) These kind of results are known as *second order analysis*.

§153 **Summary**. In this section we have directed our focus onto the applied aspects of source coding. The most important idea is the notion of universal codes, which is a active research area. Aside from implementational details (of which there admittedly are a few) the coding of messages from a known, stationary ergodic source is completely solved by the arithmetic code. As the following chapter shows, creating functional, universal codes is a much trickier business.

[8] That is, there does *not* exists a sequence $c_n = o(n)$ such that $R_n(\mu, \varphi_n) = O(c_n)$.

## Notes and references

§154 **References**. In developing the ideas of a framework based on source instances, [Billingsley, 1965], [Feder et al., 1992], [Rissanen, 1989], [Hojo et al., 1998] and [Ryabko et al., 1999] has been major inspirations. The accounts in [Shields, 1996] and [Shields, 1998] shaped much of the first part of this chapter, in particular the proof of the entropy-rate theorem. The notions of codes on $A^\infty$ using the cost function originates from [Ryabko, 1986], but the works in [Rissanen and Langdon, 1981] and [Kieffer and Yang, 1996] has been influential as well. The coding theorem for combinatorial sources linking coding and Hausdorff dimension beautifully together was pointed out to the author by Boris Ryabko, who also supplied an english translation of [Ryabko, 1986]. For an introduction to Hausdorff dimension we relied on [Billingsley, 1965] and [Falconer, 1990, 1997].

§155 **Notes on sequential codes**. The concept of *sequential codes* has been introduced on several occasions. In [Rissanen and Langdon, 1981] the notion of *alphabet extensions* used is quite similar to our sequential code. The focus, however, is on the implicit parsing (see observation §49) introduced by a sequential code. It is shown, that the parsing is unnecessary since it is possible to determine the corresponding codespace on a symbol by symbol basis. We will return to this idea later. Our treatment of the performance of codes on infinite sequences, using the quantity $c(\varphi_\infty, x)$, is based on [Ryabko, 1986]. In [Kieffer and Yang, 1996] a class of sequential codes is formalized based on the explicit parsings of the codes.

§156 **Codes on** $\mathbb{N}$. In theorem §30 we considered the code by Elias, $\mathcal{E} : \mathbb{N} \to \{0,1\}^*$, on the positive integers such that $|\mathcal{E}(n)| \leq \log n + o(\log n)$. In fact the construction in [Elias, 1975] is a bit more powerful, yielding a stronger result, namely,

$$\log^*(n) < |\mathcal{E}'(n)| \leq \log^*(n) + i_{MAX} + c$$

where $i_{MAX}$ is the largest integer such that $\log^i n \geq 0$ and $\log^*(n) \stackrel{\text{def}}{=\!=} \sum_{i=1}^{i_{MAX}} \log^i n$. Recently, [Yamamoto, 2000] presented a new code whose codewords are shorter then the previous lower bound $\log^*$ infinitely many times.

§157 **The Huffmann code**. The Huffmann code mentioned in observation §147 has been studied extensively. Since the Huffmann code is not unique we are free to consider other features within a possibly large set of codes without any cost in expected code length. One such feature is the *self-synchronizing* property, that is studied in the very readable articles [Ferguson and Rabinowitz, 1984] and [Escott and Perkins, 1998]. An adaptive version of the Huffmann construction is proposed in [Knuth, 1985], and Huffmann codes for countable infinite alphabets are considered in [Gyorfi et al., 1994], [Kato and Nagaoka, 1996] and [Linder et al., 1997].

§158 **The Arithmetic code**. The arithmetic code was first presented in an implementable version in [Rissanen, 1976]. Many improvements and variants have been forwarded since then, see e.g. [Pasco, 1979], [Rissanen and Langdon, 1979], [Langdon, 1984], [Witten et al., 1987], [Bell et al., 1990]. Interesting complexity improvement was made in [Moffat, 1990], while [Pisinger et al., 1998] suggest a method for minimizing the probability of the so called bit-stuffing in the arithmetic code using knapsack-like techniques. Recently [Ryabko and Fionov, 1999] continued improving the complexity of the algorithm. As can be seen the arithmetic code is still a very popular research area.

§159 **Twice universal codes**. Another interesting class of weak universal codes are the *twice universal codes*. Such a code yields pointwise (i.e. weak) universality on the class $\mathcal{M}_e$, but on certain subclasses $\mathsf{M} \subset \mathcal{M}_e$ for which strong universality is attainable, the code does indeed perform as good as a code tuned just for $\mathsf{M}$. These remarkable codes has been treated extensively in recent years, see e.g. [Ryabko, 1984, 1988], [Rissanen, 1983, 1986], [Willems et al., 1995] and [Feder and Merhav, 1996].

*The Cat seemed to think that there was enough of it now in sight, and no more of it appeared.*

LEWIS CARROLL

CHAPTER III

# The Lempel-Ziv codes

*I know how to spell 'banana', but I don't know when to stop.*

A LITTLE GIRL

## Introduction

The Lempel-Ziv codes are a remarkable class of codes. It is probably the most widely studied of the weak universal codes, and although there are many variants, the codes share enough key features for us to think of them as a family. The codes have shown themselves to be of significant practical interest in data compression, but they have also attracted attention as objects worthy—in their own right—of theoretical study. From the fruits of this study we present in this chapter a small sample.

For our purposes the most important feature of the Lempel-Ziv codes is that they are solutions to the well posed optimization problem of weak universal codes. It is (at least to the author) surprising that a coding procedure based on a quite simple algorithm turns out to be universal on the class of stationary ergodic sources, and judging from the relative involved nature of the proofs, this result is rather deep. The interest which the Lempel-Ziv code has attracted from fields like ergodic theory tend to support this view.

The fact that the Lempel-Ziv codes are weak universal codes, provides the link to the probabilistic framework presented in chapter II. Additionally, as we will see in chapter IV, it is possible to prove results about the Lempel-Ziv codes in the non-probabilistic setting of combinatorial sources that was also considered in chapter II. Consequently, the Lempel-Ziv code is a perfect specimen of a real-life coding procedure with enough interesting aspects to exercise our theoretical framework to its limits.

## Description of the Lempel-Ziv code

§160 **Remark**. The two best known variations of Lempel-Ziv codes, are the 1977 variant (LZ77) and the 1978 variant (LZ78), named by their emergence in [Ziv and Lempel, 1977] and [Ziv and Lempel, 1978] respectively. In the present work we will focus primarily on one of these codes, namely LZ78, with LZ77 being our variant-example; we shall not treat any of the other variants.

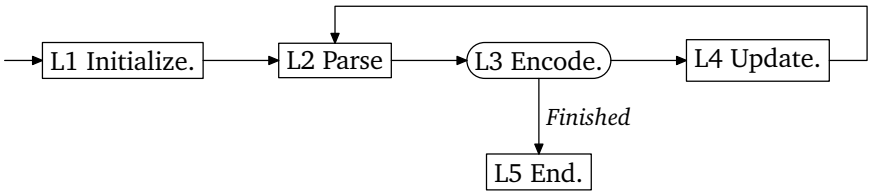



Figure 1: Flowchart for the Lempel-Ziv 1978 algorithm described in §163

§161 **Observation**. A note on the usage of the term "code" is appropriate. The Lempel-Ziv codes, while routinely referred to by the term "code", are in fact defined by algorithms, that is, procedures for transforming input symbols (the message) into a codeword, and back again. As was seen in Chapter II (see e.g. example §47) such algorithms define an injective map $\varphi_\infty : A^\infty \to B^\infty$, and the cost function enables us to treat this as a code on $A^\infty$. For the purposes in this chapter, though, it is easier to work with the finite code forms and fortunately it is easy to describe the Lempel-Ziv algorithms in more conventional terms, namely as mappings from $A^*$ to $B^*$ (a global code) or as families of maps from $A^n$ to $B^*$ (a code sequence). Our first task is therefore to describe the algorithm of LZ78 and ensure, that we understand how this algorithm in fact *defines* a code in one of the finite forms.

§162 **Definition**. A *list* $D$ is a vector $D = (d_1, \dots, d_k)$. For the empty list we write $(\,)$. Let us write $d \in D$ if $d$ is found in $D$, and if $d = d_i$, we say that $i$ is the *index* of $d$ in $D$. By *adding* an element $d$ to the list $D$ we mean setting $D = (d_1, \dots, d_k, d)$. For any $d$ we define $D(d)$ to be the index of $d$ in $D$ if $d \in D$ and $D(d) = 0$ otherwise.

§163 **Algorithm** (LZ78). Given a finite alphabet $A$ and a sequence $x_1^n \in A^n$ of length $n$ to be encoded. The list, $D$, employed by the algorithm is known as the *dictionary* in reference to step L2 of the algorithm, where we *look up* prefixes of the unparsed message in the dictionary. A flowchart of the algorithm is shown in figure 1 and a Perl implementation can be found in appendix C.

**Input:** The sequence $x_1^n$ to be encoded.

**Output:** A finite sequence of pairs of the form $(i, a) \in \mathbb{N}_0 \times A \cup \{\lambda\}$.

**L1** [Initialize.] Set the dictionary $D \leftarrow (\,)$; so that the dictionary is empty. Set $p \leftarrow 1$.

**L2** [Parse.] Let $i$ be the smallest non-negative integer so that $x_p^{p+i} \notin D$ and $p + i \leq n$, or, if no such integer exist, let $i \leftarrow n - p$. (The subsequence $x_p^{p+i}$ is known as a *phrase*.)

**L3** [Encode.] If $i = 0$, the output is $(0, x_p)$; if $x_p^{p+i} \notin D$ then the output is $(D(x_p^{p+i-1}), x_{p+i})$; if $x_p^{p+i} \in D$, the output is $(D(x_p^{p+i}), \lambda)$. If $p + i = n$ then go to step L5.

**L4** [Update.] Add $x_p^{p+i}$ to $D$. Set $p \leftarrow p + i + 1$. Go to step L2.

**L5** [End.] The algorithm ends.

Each pair $(i, a)$ of the output should be thought of as the encoding of the phrase defined in step L2. The phrase encoded into the pair $(i, a)$ can be recovered by forming the concatenation of the sequence from $D$ with index number $i$ and the symbol $a$, that is, $(i, a)$ represents $d_i a$.

§164 **Example**. Let $A = \{a, b, c\}$ and let $x_1^{14} = \texttt{abbabcbaababcc}$ be the message to be encoded. According to step L1 of the algorithm, initially the dictionary $D$ is empty and $p$ has the value 1. Advancing to step L2, we observe that $i$ is given the value 0, since clearly $x_1^{1+0}$ is not a member of the empty list $D$. Thus the first *phrase* is $x_1^1 = a$. Moving on to step L3, the output is $(0, a)$ since $i = 0$. In step L4 we set $p$ to the value 2 and add the phrase $a$ to $D$ yielding $D = (a)$. Going back to step L2 we find the next phrase to be $b$ and following the algorithm through, we see that the phrases defined by step L2 are

$$a, b, ba, bc, baa, bab, c, c.$$

The corresponding output generated in step L3 is

$$(0, a)(0, b)(2, a)(2, c)(3, a)(3, b)(0, c)(7, \lambda). \tag{1}$$

§165 **Observation** (DECODING LZ78). While we have not stated the decoding algorithm explicitly, it is seen that the original sequence may be recovered from the output, by the following procedure: Begin with an empty dictionary, $D = (\,)$. When decoding a pair $(i, a)$, the output is the concatenation of the $i$'th word in the dictionary with the letter $a$ (the 0'th word of the dictionary is defined to be the empty word $\lambda$.) Add the output to the dictionary $D$ and repeat until $n$ symbols has been decoded. The decoding procedure for the message encoded in example §164, is presented in the table below, where each row represents one step in the decoding as just explained.

| *Dictionary* | *Input* | *Output* |
|---|---|---|
| $D = (\,)$ | $(0, a)$ | $a$ |
| $D = (a)$ | $(0, b)$ | $b$ |
| $D = (a, b)$ | $(2, a)$ | $ba$ |
| $D = (a, b, ba)$ | $(2, c)$ | $bc$ |
| $D = (a, b, ba, bc)$ | $(3, a)$ | $baa$ |
| $D = (a, b, ba, bc, baa)$ | $(3, b)$ | $bab$ |
| $D = (a, b, ba, bc, baa, bab)$ | $(0, c)$ | $c$ |
| $D = (a, b, ba, bc, baa, bab, c)$ | $(7, \lambda)$ | $c$ |

Concatenating the outputs produces the original message: $\texttt{abbabcbaababcc}$.

§166 **Lemma**. *Let $x_1^n$ be a sequence of symbols from $A$. Applying the LZ78 algorithm to $x_1^n$ yields a parsing into $t$ phrases, $x_1^n = w_1 \dots w_t$, with the following properties:*

*(a) The phrases $w_1, \dots, w_{t-1}$ are distinct.*

*(b) For any $1 \le i < t$ with $|w_i| > 1$ there is a number $1 \le j < i$ and a symbol $a \in A$ so that $w_i = w_j a$.*

§167 **Proof**: In order to prove the lemma, we observe first that the only way to avoid step L4 in the algorithm is in the case where $p + i = n$, that is, when $x_p^{p+i}$ is the final phrase. Thus all phrases but the last one gets added to $D$. We now set out to prove the following: When step L4 is completed the elements of $D$ are distinct. It is clear that (a) follows if we prove this. As step L4 is the only place where elements gets added to $D$ and since $D$ is initially empty the first time step L4 is completed we have only one element in $D$. Each time we enter step L4 we know from

step L3 that $p + i \neq n$ and hence by step L2 we have $x_p^{p+i} \notin D$. Thus, when we leave step L4 the elements of $D$ are still distinct. This completes the proof of (a). Turning to the proof of (b) we use again the fact that all but the last phrase will end up as members of $D$. From step L2 and step L3 we know that when entering step L4 $i$ is the smallest non-negative integer so that $x_p^{p+i} \notin D$ and $p + i \leq n$. Assume now $i > 0$. Then $x_p^{p+i-1} \in D$ which proves that any phrase which goes into $D$ and which has length strictly greater than 1 may be written as $wa$ where $w$ is some earlier phrase and $a$ is a letter of $A$. This completes the proof of the lemma.

§168 **Remark**. In the description of the LZ78 encoding given above we have chosen to let the algorithm output integer and symbol pairs. This puts an emphasis on the key concept of the LZ78 method which is to identify and exploit recurring subsequences of the message. However, for both theoretical and practical purposes one is more likely to want the codeword to be a sequence of symbols from some code alphabet. More precisely, we wish to work with a code sequence, that is, a collection of injective maps $\{\varphi_n\}_{n\in\mathbb{N}}$ from $A^n$ to $B^*$ such that each map has uniquely concatenable codewords. In the following we present the steps needed to define such a code sequence based on the LZ78 encoding. For simplicity, we have chosen to deal only with the binary alphabet, $B = \{0, 1\}$, as code alphabet.

§169 **Observation**. For a given message length $n$ we seek an injective map from $A^n$ to $B^*$ based on the LZ78 algorithm. We may without loss of generality assume that $A = \{0, \ldots, |A|-1\}$ and then proceed to define the map $\psi : \mathbb{N} \times A \to \mathbb{N}$ in the following manner,

$$\psi(i, a) \stackrel{\text{def}}{=\!=} i|A| + a, \tag{2}$$

since a symbol $a \in A$ is also a number between 0 and $|A| - 1$. (As the reader may notice, $\psi$ is not defined for $a = \lambda$, even though the LZ78 algorithm produces this symbol in the last pair. However, since the length of the message is known, we can do fine without $\lambda$, for example by letting the LZ78 algorithm output the symbol 0 instead. In the case of example §164 the final pair would then be $(7, 0)$ in place of $(7, \lambda)$.) We note that if the pair $(i, a)$ corresponds to $w_j$ (so that $w_j = w_i a$), we have

$$\psi(i, a) = i|A| + a \leq (j - 1)|A| + |A| - 1 \leq j|A| - 1,$$

by the obvious inequality $i < j$. Thus the integer $\psi(i, a)$ representing $w_j$, may be encoded using $\lceil \log |A|j \rceil$ bits. If we make this bound into a rule stating that when encoding the j'th phrase (represented by $(i, a)$) the length of the code word is $\lceil \log |A|j \rceil$ bits, then it is possible to decode the message even after concatenating the individual encodings of each pair.

§170 **Example**. In the example above (see example §164) we studied the message `abbabcbaababcc` and we saw (see (1) of the example) that the LZ78 algorithm generated as output the following sequence of pairs:

$$(0, \mathtt{a})(0, \mathtt{b})(2, \mathtt{a})(2, \mathtt{c})(3, \mathtt{a})(3, \mathtt{b})(0, \mathtt{c})(7, \lambda).$$

The encoding described in the previous observation (§169) would then look like this:

| *The ψ mapping* | *The codeword length* | *Output* |
|---|---|---|
| $\psi(0, \mathtt{a}) = 0 + 0 = 0$ | $\lceil \log 3 \rceil = 2$ | 00 |
| $\psi(0, \mathtt{b}) = 0 + 1 = 1$ | $\lceil \log 6 \rceil = 3$ | 001 |
| $\psi(2, \mathtt{a}) = 2 \cdot 3 + 0 = 6$ | $\lceil \log 9 \rceil = 4$ | 0110 |
| $\psi(2, \mathtt{c}) = 2 \cdot 3 + 2 = 8$ | $\lceil \log 12 \rceil = 4$ | 1000 |
| $\psi(3, \mathtt{a}) = 3 \cdot 3 + 0 = 9$ | $\lceil \log 15 \rceil = 4$ | 1001 |
| $\psi(3, \mathtt{b}) = 3 \cdot 3 + 1 = 10$ | $\lceil \log 18 \rceil = 5$ | 01010 |
| $\psi(0, \mathtt{c}) = 0 + 2 = 2$ | $\lceil \log 21 \rceil = 5$ | 00010 |
| $\psi(7, \lambda) = 7 \cdot 3 + 0 = 21$ | $\lceil \log 24 \rceil = 5$ | 10101 |

producing as a codeword the sequence 000010110100010010101000010101010101. Note, that even though we have to each $n \in \mathbb{N}$ a map from $A^n$ to $B^*$ we cannot use it as a global code. (See observation §38.) For example, the LZ78 block-code (of block-length 4) will encode abba into 000010110 which is a prefix of the codeword above.

§171 **Definition**. Let $n \in \mathbb{N}$. The map from $A^n$ to $B^*$ defined by the LZ78 algorithm (via observation §169) is known as the *Lempel-Ziv-78 block-code* and we write $\varphi_{LZ78} : A^n \to B^*$.

§172 **Observation** (PARSE RULES). Let $x_1^n \in A^n$ be a message. As we have seen, a *parsing* of $x_1^n$ is a subdivision of $x_1^n$ into phrases $w_1, \ldots, w_t$ so that $x_1^n = w_1 \cdots w_t$. A parsing may conveniently be defined incrementally by specifying how the next phrase should be selected. For example, by defining the rule "the next phrase is the next symbol", we get the trivial parsing $x_1^n = w_1 \cdots w_n$ with $|w_i| = 1$. The parsing induced by the 1978 Lempel-Ziv algorithm can be described in this fashion by the rule

**LZ78:** *The next phrase is the shortest new block that has not been seen as a phrase previously.*

A couple of things should be pointed out: A *new* phrase is a sequence which has not already been defined as a phrase. The first phrase is the first symbol, since no phrases has been defined yet and hence any (non-empty) phrase is new. Finally, the last phrase is either a new phrase or simply the rest of the sequence to be parsed in case there are not enough symbols left to yield a new phrase.

§173 **Observation** (THE LZ77 VARIATION). The variation known as Lempel-Ziv '77 differs from the LZ78 code primarily in the parse rule. The parse rule of the LZ77 code can be stated as

**LZ77:** *The next phrase is the shortest new block that does not start somewhere in the past.*

As an example, consider the sequence bananananana. Comparing the two parsing methods,

LZ78: b, a, n, an, ana, na
LZ77: b, a, n, ananana

we find that the LZ77 method produces longer phrases. Aside from the different parsing rule, the LZ77 needs a slightly modified encoding of the phrases, because each phrase is represented not by a pair, but by a triple taking into account that the recurring part of the phrase needs to be identified not only by its starting point in the past but also by its length.

§174 **Summary**. We have introduced the LZ78 algorithm as a two-stage method consisting of a parsing stage and an encoding stage. Some properties of the phrases of the LZ78 parsing was then identified. The encoding of each phrase was described in details leading to the definition of a prefix code sequence. The idea of parse rules allowed for an easy explanation of the variation known as LZ77. Finally, it was implied that at least part of the LZ77 concept has been independently discovered by an unknown little girl!

## First order analysis

§175 **Observation**. Let $x_1^n \in A^n$ and let $x_1^n = w_1 \cdots w_t$ be the corresponding LZ78 parsing. We now derive a bound on the code length of the LZ78 code, expressed in $t$. According to observation §169, each phrase $w_i$ is encoded using $\lceil \log i|A| \rceil \leq \log i|A| + 1$ bits. From this we conclude that there is the following (rather crude) upper bound on the total codelength,

$$\begin{aligned} |\varphi_{LZ78}(x_1^n)| &\leq \sum_{i=1}^{t} \big( \log i|A| + 1 \big) \\ &\leq t \log t + t \log |A| + t \\ &= t \log t + o(t \log t), \end{aligned} \tag{3}$$

as $n \to \infty$. (It is clear that $t$ grows to infinity when $n$ does, since the phrases are distinct.)

§176 **Remark**. In the light of the preceding observation we find that a good understanding of the number of phrases, $t(x_1^n)$, produced by the LZ78 algorithm, is an important key in finding theorems about the performance of the code. Consequently this section contains lemmas and theorems which increases our knowledge of parsings in general, and the Lempel-Ziv parsing in particular.

§177 **Observation**. As we will see shortly, the (finite) sequences one gets from concatenating all possible words of lengths 1, followed by all words of length 2 and so forth up to some finite number, $m$, is of importance when analyzing LZ78. For example, for the case $A = \{a, b, c\}$ and $m = 2$, one of these sequences (there are many due to the possibility of different orderings of the words) is

$$a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc,$$

where the commas are included for readability only. The length of this sequence is 21, and it is clear that all the other sequences corresponding to $|A| = 3$ and $m = 2$, have the same length. In [Pierce II and Shields, 1998] the term *Champernowne sequence* is proposed for this type of sequences, due to the similarity with the well known Champernowne number $0.1234567891011121314\cdots$, which appeared in [Champernowne, 1933] where it was proved to be normal[1] in base 10. We may calculate the length of a Champernowne sequence from the size of $A$ and the value of $m$ using the formula

$$\sum_{i=1}^{m} i|A|^i.$$

[1] A number $\alpha \in [0,1]$ is said to be *normal* in base $N$ if the base-$N$ expansion of $\alpha$, that is, $\alpha = 0.a_1a_2\cdots$ with $a_i \in \{0, \ldots, N-1\}$, contains the different digits $0, \ldots, N-1$ in equal proportions. In other words, if $f(a|a_1 \cdots a_n)$ is the number of times $a$ occur in the first $n$ digits, then $\lim_{n\to\infty}(1/n)f(a|a_1 \cdots a_n) = 1/N$ for all $a \in \{0, \ldots, N-1\}$.

We will encounter this sum a number of times in the following pages.

§178 **Lemma** (COMPACTNESS). *Let* $x_1^n \in A^n$ *and let* $t(x_1^n)$ *be the number of phrases in the LZ78 parsing of* $x_1^n$*. Then*

$$t(x_1^n) \leq \frac{n(1+\delta_n)}{\log n} \log |A|$$

*where* $\delta_n \to 0$ *as* $n \to \infty$*.*

§179 **Proof**: In order to get an upper bound on $t(x_1^n)$ we use the fact that among all but the last phrase, any phrase is distinct from all other phrases. The proof is based on the (trivial) fact, that the maximal number of distinct phrases is achieved when each of the $|A|^i$ possible words of length $i$ occurs as a phrase (see observation §177). In order to express the idea of the proof precisely, we introduce to each integer $m \in \mathbb{N}$ two numbers $n_m$ and $t_m$. One should think of $m$ as the maximal length of the phrases in the distinct parsing. Then $n_m$ is the length of the longest sequence that can be made by concatenating distinct phrases of length no more than $m$, and $t_m$ is the number of phrases in such a sequences. In the following, we let $k = |A|$. It is clear that

$$t_m = \sum_{i=1}^{m} k^i, \tag{4}$$

since this is the number of phrases we get by using each phrase once. As we saw in observation §177, the corresponding number, $n_m$, should be defined as follows,

$$n_m = \sum_{i=1}^{m} i k^i. \tag{5}$$

Fortunately, both of the above sums, have exact formulas allowing for easy calculation of their values as well as derivation of bounds. The sum (4) is well known and may be calculated as

$$t_m = (k^{m+1} - k)/(k-1). \tag{6}$$

The sum (5) may be calculated using lemma §278 as follows,

$$n_m = \frac{mk^{m+2} - (m+1)k^{m+1} + k}{(k-1)^2}, \tag{7}$$

and from observation §280 we get the following bound:

$$n_m \leq \frac{mk^{m+1}}{k-1}. \tag{8}$$

Returning to the LZ78 parsing we find that if $n = n_m$ for some $m$, then the number of distinct phrases in $x_1^n$ is upper bounded by $t_m$. However, if $x_1^n$ does indeed yield $t_m$ distinct phrases, then these will have total length $n_m = n$ hence the number of all phrases is upper bounded by $t_m$. The bound is tight because $t(x_1^n)$ does indeed equal $t_m$ for a Champernowne sequence $x_1^n$. We are now ready to treat the general case: Let $m$ be chosen as the largest integer such that $n_m \leq n$. By (5) we have $n_{m+1} - n_m = (m+1)k^{m+1}$ so that we may write $n = n_m + \eta$ with $0 \leq \eta < (m+1)k^{m+1}$. This enables the estimation,

$$t(x_1^n) \leq t_m + \frac{\eta}{m+1} + 1, \tag{9}$$

which follows since $t_m$ is the number of distinct phrases of length at most $m$ and their length is $n_m$, so that the number of phrases not counted by $t_m$ is at most $(n-n_m)/(m+1) = \eta/(m+1)$. The additional one phrase reflects the possibility of a non-unique last phrase in the parsing. We now have

$$t_m + 1 = \frac{k^{m+1}-k}{k-1} + 1 \leq \frac{k^{m+1}}{k-1} \leq \frac{k^{m+1}}{k-1} \cdot \frac{m(k-1)+1/k^m-1}{m(k-1)+1-k}$$

where the equality follows directly from (6) and the last inequality holds because we multiply by a fraction strictly greater than 1. Evaluating the slightly "magic" expression on the right hand side of the inequality and comparing with (7) we get

$$t_m + 1 \leq \frac{n_m}{m-1}.$$

Combining with (9), we have

$$t(x_1^n) \leq \frac{n_m}{m-1} + \frac{\eta}{m+1} \leq \frac{n_m+\eta}{m-1} = \frac{n}{m-1}. \tag{10}$$

The next task is to estimate $m-1$ in terms of $n$. We find that

$$\log n \leq \log n_{m+1} \leq \log(m+1) + (m+2)\log k - \log(k-1) \leq (m+2)\log k + \log(m+1) \tag{11}$$

where the inequalities follows from the choice of $m$, inequality (8) and a trivial estimation respectively. On the other hand,

$$\log n \geq \log n_m = \log(mk^m + n_{m-1}) \geq m \log k,$$

where the first inequality is by the choice of $m$. Combining with (11) we get

$$m-1 \geq \frac{\log n - \log(m+1)}{\log k} - 3 \geq \frac{\log n - \log(1+\log n/\log k)}{\log k} - 3,$$

and for suitable $\epsilon_n$ with $\lim_{n\to\infty} \epsilon_n = 0$ we may write

$$m-1 \geq \frac{\log n}{\log k}(1-\epsilon_n).$$

Finally, by combining with (10) we conclude that the following bound holds:

$$t(x_1^n) \leq \frac{n \log k}{\log n(1-\epsilon_n)}.$$

Since $1/(1-\epsilon_n) = 1 + \epsilon_n/(1-\epsilon_n)$ setting $\delta_n = \epsilon_n/(1-\epsilon_n)$ yields the desired result:

$$t(x_1^n) \leq \frac{n(1+\delta_n)}{\log n} \log |A|.$$

§180 **Corollary**. *Let* $x_1^n \in A^n$ *and let* $x_1^n = w_1 \cdots w_t$ *be a distinct parsing. Then* $t \leq n(1+\delta_n)\log|A|/\log n$ *with* $\delta_n \to 0$ *as* $n \to \infty$.

§181 **Proof**: The corollary follows since the proof of the compactness lemma, §§179, does not use any specific features of the LZ78 parsing, aside from the fact that the phrases are distinct. Leaving

out the possibility of a non-distinct last phrase, is in fact a strengthening of the conditions, so this only works in favor of the corollary.

§182 **Observation**. The compactness lemma just proven can be interpreted as saying that the LZ78 code will never (asymptotically) expand the message. To see this, note that the length of the code for a long message, $x_1^n \in A^n$, is dominated (see observation §175) by $t \log t$ where $t$ is the number of phrases. By the compactness lemma, and the fact that $t \log t$ is itself bounded by $t \log n$, we conclude that the codelength is essentially bounded by $n \log |A|$ bits, and this is, of course, the best possible bound.

§183 **Theorem** (LZ78 UNIVERSALITY THEOREM). *Let $\mu$ be a stationary, ergodic measure. Then*

$$\limsup_{n\to\infty} \frac{|\varphi_{LZ78}(x_1^n)|}{n} = h$$

*almost surely, when $h$ is the entropy of $\mu$.*

§184 **Remark**. The proof of theorem §183 is carried out in §199 and is based on a the distinct parsing theorem (theorem §197), which is of interest in its own right. Thus the reader might want to read the distinct parsing theorem, theorem §197, and the proof of the LZ78 universality theorem first in order to see how the study of distinct parsings leads to the LZ78 result.

§185 **Lemma** (BUILDING BLOCKS). *Let $\mu$ be a stationary, ergodic measure with entropy $h$ and let $\delta > 0$ be given. Then there exists, for a sufficiently large integer $k$, a set $B_k \subseteq A^k$ with $\mu(B_k) \geq 1 - \delta/2$ and $|B_k| \leq 2^{k(h+\delta)}$.*

§186 **Proof**: Let $\mu$ with entropy $h$ and $\delta > 0$ be given. By the entropy theorem, we have for the set

$$B_n \stackrel{\text{def}}{=\!=} \{a_1^n \mid \mu(a_1^n) \geq 2^{-n(h+\delta)}\}$$

that $\mu(B_n) \to 1$ as $n \to \infty$. It follows that there exist an integer $k$ such that $\mu(B_k) \geq 1 - \delta/2$. By the definition of $B_k$ we have a lower bound on the measure of the elements, and this turns into an upper bound on the size of the set: $|B_k| \leq 2^{k(h+\delta)}$.

§187 **Remark**. The next three lemmas are about properties of distinct parsings of sequences that are built from the building blocks of the building block lemma. To give a brief overview of what to expect, the lemma of rare phrases tells that if a sequence is $(1-\delta)$ covered by a set $B$ then so is most of the phrases in a distinct parsing. The lemma of short phrases shows that if the sequence we are parsing into distinct phrases is long enough, then the phrases shorter than a given length does not cover too much of the whole sequence. Both of these lemmas are almost trivial. The third lemma, the common phrase lemma, is a counting type lemma giving an estimate of phrases that can be made provided the phrase is "common" in the sense of being $(1-\delta)$ built up from a set $B$. The general idea of the proof of the distinct parsing theorem is to use the first two lemmas to show that we need not worry about short phrases nor "rare" phrases (phrases that are not $(1-\delta)$ built from a set $B$), and then use the bound from the common phrase lemma to argue that the common phrases must be long (exactly how long depends on the entropy of the stationary ergodic measure $\mu$.)

§188 **Definition**. Let $x_1^n \in A^n$ and let $B \subseteq A^*$. Let $\mathcal{C}$ be the collection of integer intervals $[s,t]$ such

that $x_s^t \in B$. If $\mathcal{C}$ is a strong $(1-\delta)$-cover (see definition §91), we say that $x_1^n$ *is* $(1-\delta)$*-covered by* $B$.

§189 **Lemma** (LEMMA OF RARE PHRASES). *Let* $B_k \subseteq A^k$ *and assume that* $x_1^n$ *is* $(1-\delta^2/2)$*-covered by* $B_k$. *For any parsing* $x_1^n = w_1 \cdots w_t$, *let* $W_1 \stackrel{\text{def}}{=\!=} \{w_i \mid w_i \text{ not } (1-\delta)\text{-covered by } B_k\}$. *Then the words of* $W_1$ *makes only a small contributes to the total length of* $x_1^n$, *that is,*

$$\sum_{w \in W_1} |w| \leq \delta n/2.$$

§190 **Proof**: If $\sum_{w\in W_1} |w| > \delta n/2$ then we would have at least $\delta \sum_{w\in W_1} |w| \geq \delta^2 n/2$ indexes $i$ in $[1, n-k+1]$ for which $x_i^{i+k-1} \notin B_k$ which is a contradiction according to definition §188. We conclude that $\sum_{w\in W_1} |w| \leq \delta n/2$.

§191 **Lemma** (LEMMA OF SHORT PHRASES). *For any* $\delta > 0$ *and any* $k \in \mathbb{N}$ *there exists a number* $N$ *such that if* $n > N$ *and* $x_1^n = w_1 \cdots w_t$ *is a distinct parsing, then*

$$\sum_{w \in W_2} |w| \leq \delta n$$

*where* $W_2 \stackrel{\text{def}}{=\!=} \{w_i \mid |w_i| < k\}$, *the set of 'short' phrases.*

§192 **Proof**: Since the phrases are distinct, the phrases of length less than $k$ can cover only a fixed number of symbols of $x_1^n$. Let this number be $n_0$ and set $N = n_0/\delta$. We now have, assuming $n > N$, that

$$n = \sum_{|w_i|\geq k} |w_i| + \sum_{|w_i|<k} |w_i| \leq \sum_{|w_i|\geq k} |w_i| + n_0,$$

where the inequality follows by the definition of $n_0$. Shifting our point of view slightly, we get

$$\sum_{|w_i|\geq k} |w_i| \geq n - n_0 = n(1 - n_0/n) \geq n(1-\delta),$$

which completes the proof.

§193 **Observation**. One interpretation of lemma §191 is that when parsing sufficient long sequences into distinct words, the length of the words *must* grow. We also note, that although the exact value of $n_0$ as introduced in the proof is not needed, it does in fact equal $\sum_{i=1}^{k} i|A|^i$; see observation §177.

§194 **Lemma** (THE LEMMA OF COMMON PHRASES). *Let* $\delta > 0$, $\epsilon > 0$ *and let* $B_k \subseteq A^k$ *be a subset with* $|B_k| \leq 2^{k(h+\epsilon)}$ *for some value* $h$. *Let* $G_m$ *be the sequences of length* $m$ *that are* $(1-\delta)$*-built up from* $B_k$. *Then*

$$|G_m| \leq 2^{m(h+2\epsilon)}$$

*for sufficiently small values of* $\delta$.

§195 **Proof**: The proof follows the same idea as the proof of the counting lemma. We begin by noting that any sequence from $G_m$ is $(1-\delta)$-built up and therefore will be compatible with a skeleton in $[1, m]$ made from intervals of length $k$. Let $\mathcal{S}$ be such a skeleton, and let $G_{\mathcal{S}}$ be the sequences

which are compatible with $\mathcal{S}$. We now estimate the size of $G_{\mathcal{S}}$. Denoting by $|\mathcal{S}|$ the number of intervals in $\mathcal{S}$ we have an upper bound on the number of ways, $|B_k|^{|\mathcal{S}|}$, we can fill this skeleton,

$$|B_k|^{|\mathcal{S}|} \leq |B_k|^{m/k}.$$

The remaining parts of $x_1^m$ has a total length of no more than $\delta m$ and can be filled in at most $|A|^{\delta m}$ ways, yielding the following bound,

$$|G_{\mathcal{S}}| \leq |A|^{\delta m}|B_k|^{m/k}.$$

Finally, we consider the number of different skeletons. Each skeleton corresponds to a binary sequence $b_1^m$ where $b_i$ is 0 if $i$ is covered by some interval in the skeleton and $b_i$ is 1 otherwise. For example, if $m = 7$ and the skeleton considered is $\{[2,3],[5,7]\}$ we would have the representation $b_1^7 = 1001000$. It is clear, that the number of 1's in $b_1^m$ must be less than $\delta m$. As was seen in the proof of the counting lemma (see lemma §102), such a bound translates into the bound $2^{mH_b(\delta)}$ on the number of such sequences, and hence skeletons. We conclude,

$$|G_m| \leq 2^{mH_b(\delta)}|A|^{\delta m}2^{m(h+\epsilon)}$$

which is the desired result provided $H_b(\delta) + \log|A|\delta \leq \epsilon$.

§196 **Remark**. The following theorem is the key result in this section. It is important to note, that while the preceding lemmas are expressed in terms of arbitrary, finite sequences $(x_1^n)$, the following theorem deals with elements of $A^\infty$. In particular the (finite) prefixes of such infinite sequences are considered, which is why the lemmas can be brought into play.

§197 **Theorem** (DISTINCT PARSING THEOREM). *Let $\mu$ be a stationary, ergodic measure with entropy $h$. There exists then a set $\mathcal{G} \subset A^\infty$ with $\mu(\mathcal{G}) = 1$ and such that for any $x \in \mathcal{G}$ and any $\epsilon > 0$ there is an integer $n_\epsilon$ so that the following holds: if $n > n_\epsilon$ and $x_1^n = w_1 \cdots w_t$ is a parsing into distinct words, then*

$$\sum_{|w_i| \geq (\log n)/(h+\epsilon)} |w_i| \geq (1-\epsilon)n.$$

§198 **Proof**: In order for the main idea of the proof to be visible, the proof is based heavily on a number of lemmas. Let $\xi > 0$ and $\delta > 0$ be positive numbers. Exactly how small $\xi$ and $\delta$ should be will be specified later. For now we just need to make sure the steps we perform work for arbitrarily small, positive values of $\xi$ and $\delta$. By lemma §185 we may pick a number $k$ and a set $B_k \subseteq A^k$ with $\mu(B_k) \geq 1 - \delta^2/4$ and $|B_k| \leq 2^{k(h+\xi)}$. The elements of this set is going to play the role of *building blocks* in the rest of the proof.

By the sliding window lemma (see lemma §105), we get a set $\mathcal{G} \subseteq A^\infty$ of measure 1 such that for any $x \in \mathcal{G}$ there is an integer $N$ such that for any $n > N$ we have $x_i^{i+k-1} \in B_k$ for at least $(1-\delta^2/2)n$ of the indices $i \in [1, n-k+1]$. In other words, for any $x \in \mathcal{G}$ the prefix $x_1^n$ is eventually (as $n$ grows) strongly $(1-\delta^2/2)$-covered by $B_k$.

We now consider the parsing $x_1^n = w_1 \cdots w_t$ for some element of $\mathcal{G}$ assuming that $n > N$. Let $W = \{w_1, \ldots, w_t\}$ be the set (since the words are distinct) of words (or phrases) that $x_1^n$ is parsed into. In the following we identify some subsets of $W$ of special interest.

Let $W_1 \subset W$ be the words that are *not* strongly $(1-\delta)$-covered by $B_k$. By lemma §189 we see that those words contributes less than $\delta n/2$ to the total length of $x_1^n$, that is,

$$\sum_{w \in W_1} |w| \leq \delta n/2.$$

Let $W_2 \subset W$ be the set $W_2 = \{w \in W \mid |w| \leq k/\delta\}$. By lemma §191 we have a number $N'$ so that for $n > N'$, the words of $W_2$ contributes only a small fraction of the total length $n$, that is,

$$\sum_{w \in W_2} |w| \leq \delta n/2.$$

Let $W_3 = W \backslash (W_1 \cup W_2)$, that is, the words which *are* $(1-\delta)$-covered by $B_k$ and which *are* longer than $k/\delta$. By the packing lemma (see lemma §95) we see that those words, are indeed $(1-\delta)$-built up from $B_k$.

Let $G_m \subseteq A^m$ be the set of all sequences which are $(1-\delta)$-built up from $B_k$, so we have $W_3 \subseteq \cup_{m \geq k/\delta} G_m$. By the lemma of built-up phrases, lemma §194, we find

$$|G_m| \leq 2^{m(h+\xi)}, \tag{12}$$

provided $\delta$ is small enough for the lemma to work. As noted in the beginning of the proof, we are indeed allowed to pick $\delta$ as small as we wish, so we may assume that $\delta \leq \xi/2$ as well.

We now have

$$n = \sum_{w \in W} |w| \leq \sum_{w \in W_1 \cup W_2} |w| + \sum_{w \in W_3} |w| \leq \frac{1}{2}\xi n + \sum_{w \in W_3} |w|,$$

from which we proceed to estimate the last sum. We do this by introducing the number $s = (1-\xi)\log n/(h+\xi)$ as well as the set $W_{3s} \stackrel{\text{def}}{=\!=} \{w \in W_3 \mid |w| \leq s\}$ of 'short' phrases and the set $W_{3l} \stackrel{\text{def}}{=\!=} \{w \in W_3 \mid |w| > s\}$ of 'long' phrases. We may then write

$$\sum_{w \in W_3} |w| = \sum_{w \in W_{3s}} |w| + \sum_{w \in W_{3l}} |w|.$$

On the right hand side, the first sum can be upper bounded by $\sum_{m \leq s} m|G_m|$ since we know the words are distinct. By the bound (12) we have

$$\sum_{w \in W_{3s}} |w| \leq \sum_{m=1}^{\lfloor s \rfloor} m 2^{m(h+\xi)}$$

and by the bound (4) of observation §280 (setting $\alpha = 2^{h+\xi}$) we get

$$\sum_{W_{3s}} |w| \leq \frac{s 2^{h+\xi} 2^{s(h+\xi)}}{2^{h+\xi}-1} = \frac{\frac{(1-\xi)\log n}{h+\xi} 2^{h+\xi} 2^{(1-\xi)\log n}}{2^{h+\xi}-1} = O(n^{1-\xi}\log n)$$

as $n \to \infty$. And so, provided $n$ is large enough, we have

$$n \leq \frac{1}{2}n\xi + \frac{1}{2}n\xi + \sum_{w \in W_{3l}} |w|. \tag{13}$$

We may pick $\xi$ so that

$$\frac{\log n}{h+\epsilon} \leq \frac{(1-2\xi)\log n}{h+\xi}$$

and thus get (for $w \in W_3$),

$$\sum_{|w|>\frac{(1-\xi)\log n}{h+\xi}} |w| \leq \sum_{|w|>\frac{(1-2\xi)\log n}{h+\xi}} |w| \leq \sum_{|w|>\frac{\log n}{h+\epsilon}} |w|, \tag{14}$$

for sufficiently large $n$. The theorem follows by noting that $\xi < \epsilon$ and combining (13) and (14).

§199 **Proof (of theorem §183)**: Recall that $\mu$ is a stationary, ergodic measure of entropy $h$. Let $x$ be a member of the set $\mathcal{G}$ whose existence was proven in the distinct parsing theorem. We first note, that the LZ78 code parses any prefix $x_1^n$ of $x$ into phrases, $x_1^n = w_1 \cdots w_t$, of which all but possibly the last are distinct. Let us assume, that the last phrase $w_t$ does indeed equal one of the previous ones. Thus we have by a rough estimate $|w_t| \leq n/2$ and $|w_1 \cdots w_{t-1}| \geq n/2$. We conclude that the LZ78 parsing of $x_1^n$ consists of a distinct parsing of total length at least $n/2$ plus possibly an additional phrase. Because the set $\mathcal{G}$ has measure 1, it is enough to show that for any $\epsilon > 0$ there is a number $N$ so that $|\varphi_{LZ78}(x_1^n)|/n \leq h + \epsilon$. Thus let $\epsilon > 0$ be given and we have according to the distinct parsing theorem a corresponding number $n_\epsilon$. We then apply the LZ78 algorithm to $x_1^{2n_\epsilon}$. As estimated above, the first half of this sequence is parsed into distinct words, that is, $x_1^n = w_1 \cdots w_{t-1}$ is a distinct parsing with $n \geq n_\epsilon$. By the distinct parsing theorem, it follows that the phrases of length greater than $(\log n)/(h+\epsilon)$ covers most of $x_1^n$. More precisely, let $W = \{w_1, \ldots, w_{t-1}\}$ be the set of phrases, and set $W_l = \{w \in W \mid |w| \geq (\log n)/(h+\epsilon)\}$, then

$$\sum_{w \in W_l} |w| \geq (1-\epsilon)n.$$

This implies, conversely, that phrases shorter than $(\log n)/(h+\epsilon)$ cover at most $\epsilon n$ of $x_1^n$. These "short" phrases, $W_s = W \backslash W_l$, are still distinct, of course, so corollary §180 applies and yields the bound

$$|W_s| \leq \frac{\epsilon n(1+\delta_{\epsilon n})}{\log \epsilon n}$$

where $\delta_{\epsilon n} \to 0$ as $\epsilon n \to \infty$. On the other hand, the number of "long" phrases is upper bounded by the total length of the message divided by the minimal length of the long phrases, that is $n(h+\epsilon)/\log n$, so we have

$$t - 1 = |W_l| + |W_s| \leq \frac{n(h+\epsilon)}{\log n} + \frac{\epsilon n(1+\delta_{\epsilon n})}{\log \epsilon n}.$$

We have then shown, that for any $x \in \mathcal{G}$ and for any $\epsilon > 0$

$$\limsup_{n\to\infty} \frac{t \log n}{n} \leq h + \epsilon.$$

Recalling from observation §175, that $|\varphi_{LZ78}(x_1^n)| = t \log t + o(t \log t)$ we get the upper bound

$$\limsup_{n\to\infty} \frac{|\varphi_{LZ78}(x_1^n)|}{n} \leq h,$$

which holds almost surely since $\mathcal{G}$ has measure 1. The opposite inequality is a consequence of the fact that entropy is an almost sure lower bound on codelength, see theorem §122. This concludes the proof.

§200 **Remark**. We have now established weak universality for the LZ78 code. We would like to point out, however, two other ways in which the LZ78 code may be considered optimal. We do that by stating, without proof, the following theorem.

§201 **Definition**. Let $\mathrm{FS}(s)$ be the set of all finite state encoders of at most $s$ states. Let

$$\rho_{\mathrm{FS}(s)}(x_1^n) = \min_{\varphi \in \mathrm{FS}(s)} \frac{|\varphi(x_1^n)|}{n}$$

be the minimal compression ration possible with finite state encoders of no more than $s$ states. For $\mathbf{x} \in A^\infty$ define

$$\rho(\mathbf{x}) = \lim_{s\to\infty} \limsup_{n\to\infty} \rho_{\mathrm{FS}(s)}(x_1^n)$$

known as the *finite state compressibility* of $\mathbf{x}$.

§202 **Theorem** (FINITE-STATE OPTIMALITY). *Let $\mathbf{x} \in A^\infty$. Then for any $s \in \mathbb{N}$*

$$\frac{|\varphi_{\mathrm{LZ78}}(x_1^n)|}{n} \le \min\left\{ \frac{|\varphi(x_1^n)|}{n} \;\middle|\; \varphi \in \mathrm{FS}(s) \right\} + \delta_s(n)$$

*with $\delta_s(n) \to 0$ as $n \to \infty$. Furthermore,*

$$\limsup_{n\to\infty} \frac{|\varphi_{\mathrm{LZ78}}(x_1^n)|}{n} \le \rho(\mathbf{x}).$$

*where $\rho(\mathbf{x})$ is the finite state compressibility of $\mathbf{x}$.*

§203 **Proof**: The proof can be found in [Ziv and Lempel, 1978, Theorem 2].

§204 **Summary**. Having identified the number of phrases of the LZ78 encoding as a key to understand the performance of the code we proceeded with a detailed analysis of distinct parsings of sequences. This led to the main theorem stating that the LZ78 code is a weak universal code on the class of stationary ergodic measures. The method of proof was based on the packing and counting ideas known from Chapter II. Finally we have seen very briefly how the LZ78 code is also to be considered optimal as a code which performs as well as any given finite state encoder and furthermore as a code which attains the finite state compressibility.

## Symbolwise analysis

§205 **Remark**. From a practical point of view it is quite obvious *how* the Lempel-Ziv algorithms achieve compression: Recurring pieces of the data sequence are identified and encoded by referring to earlier occurrences much in the same way as a book might begin with a list of abbreviations used throughout the text. However, it is difficult to point, in more precise terms, to the exact "reason" as to why the codes works so well. It is often said that this difficulty is due to the

fact that *modeling* and *parameter estimation* are intertwined in the LZ codes. In this section we describe attempts to separate this modeling and parameter estimation.

§206 **Observation** (SEQUENTIAL CODES REVISITED). The concept of sequential codes was introduced in chapter II, observation §49. The LZ algorithm works by alternating between reading finite phrases from the message and writing finite parts of the codeword. Hence the resulting code, $\varphi_{LZ78} : A^\infty \to B^\infty$ is sequential. We even have, for any sequence $\mathbf{x} \in A^\infty$, that $c(\varphi_{LZ78}, \mathbf{x}) \leq 1$ by the compactness lemma (§178). In the following we will, however, focus on the block version of the LZ78 code, that is, $\varphi_{LZ78} : A^n \to B^*$. Of course, the sequential property of the LZ78 code is still present and clearly visible for large $n$, a fact which is almost directly implied by the definition of sequential codes with limited delay (see definition §50).

§207 **Observation**. In [Rissanen and Langdon, 1981, Theorem 1] it was seen that there exists a "symbolwise equivalent" to; or a "symbolwise simulation" of any code from a quite large class of sequential codes, the so called model-based codes. The Lempel-Ziv codes are not (in general) members of this class; nevertheless, the techniques used in the proof of the result of Rissanen and Langdon, can be applied with success to give us a symbolwise simulation of the LZ78 code. Let $x_1^n$ be a message to be encoded. A symbolwise simulation for a block-code $\varphi_n$ is a probability assignment of conditional probabilities, $p(x_i|x_1^{i-1})$ for all $i$ with $1 \leq i \leq n$, so that for any $x_1^n \in A^n$ the induced probability $p(x_1^n) = p(x_1|\lambda)p(x_2|x_1)\cdots p(x_n|x_1^{n-1})$ agrees with the one implicitly defined by the code, that is, $p(x_1^n)$ is equal to $2^{-|\varphi_n(x_1^n)|}$. (The idea that any code implicitly defines a probability is of course based on the well known connection between the probability $p(x_1^n)$ and the ideal codelength $-\log p(x_1^n)$. Reversing this connection leads us to state that the encoding of the sequence $x_1^n$ using $|\varphi_n(x_1^n)|$ code-letters, implicitly assigns a probability of $2^{-|\varphi_n(x_1^n)|}$ to the sequence $x_1^n$.) The correspondence between probabilities and code-lengths is seen clearly if one considers the arithmetic encoding technique (see observation §149.) By this technique the symbolwise, conditional probabilities from above, may be used to encode a message into a codeword of length $-\log p(x_1^n) = |\varphi_n(x_1^n)|$. It is due to this fact, that we may speak of the symbolwise simulation as the decoupling of modeling and coding. Sometimes the term *code space* is used in place of the term probability. The concept of symbolwise simulation is conveniently introduced in probabilistic terms (as was done above), but it is undoubtedly more correct to use the term code space, and we will do so below. A symbolwise simulation corresponding to the LZ78 code will be derived in the following.

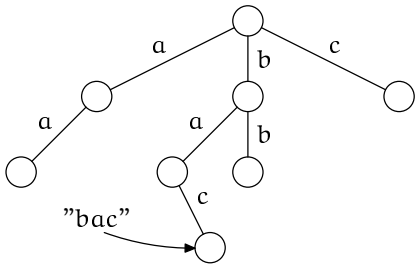
§208 **Observation**. Consider using the LZ78 algorithm on a binary sequence, $x_1^n \in A^n$. Since we have $|A| = 2$, we find by observation §169, that the encoder emits $\lceil \log 2i \rceil = \lceil \log i \rceil + 1$ bits when encoding the phrase $w_i$. Also, by lemma §166(b), we have $w_i = w_j a$ for $j < i$ and $a \in A$. From observation §169(2) (but see also example §170) it is seen that in fact $w_i$ is encoded using $\lceil \log i \rceil$ bits to represent $w_j$ and a single bit to represent $a$. The main idea in the development of a symbolwise equivalent to the LZ78 code, is that whenever the encoder decides to encode a part, $w_j$, of the source using a particular number, $\lceil \log i \rceil$, of bits, in fact it assigns a code space of $2^{-\lceil \log i \rceil}$ to that part. For technical reasons, we drop the ceiling function and say that the LZ78 encoding of the phrase $w_i$ assigns a code space of $2^{-\log i}$ to $w_j$ and $2^{-1}$ to $a$, in total, $\frac{1}{2i}$ to $w_i$.

§209 **Example**. Let the sequence to be encoded be $0110110010110\cdots$. The LZ78 code produces phrases $w_1, w_2, \ldots$ and corresponding pairs as follows:

| Phrases: | 0 | 1 | 10 | 11 | 00 | 101 | $\cdots$ |
|---|---|---|---|---|---|---|---|
| Pairs: | (0,0) | (0,1) | (2,0) | (2,1) | (1,0) | (3,1) | $\cdots$ |

The first phrase, 0, corresponding to the pair $(0,0)$, is encoded using $\lceil \log 1 \rceil + 1 = 1$ bit. Thus implicitly, a code space of $\frac{1}{2}$ is assigned to the phrase $w_1 = 0$. The second phrase, 1, corresponding to the pair $(0,1)$ is encoded using $\lceil \log 2 \rceil + 1 = 2$ bits yielding an assigned code space of $\frac{1}{4}$ for $w_2$. For the next phrase, something interesting happen: The encoding of the pair corresponding to $w_3 = w_2 0$ takes up $\lceil \log 3 \rceil + 1 = 3$ bits. As was seen in the observation above, the first two bits are a contribution from the encoding of (the index of) $w_2$, while the last bit is due to the encoding of 0. However, as stated above, when considering assigned code space, we work with idealized codelength, so we conclude, that $w_2$ was assigned a code space of $2^{-\log 3} = \frac{1}{3}$ while 0 was assigned $\frac{1}{2}$. Jumping to the encoding of $w_6 = w_3 1 = 101$ we see that a total of $\lceil \log 6 \rceil + 1 = 4$ bits will be used, so $w_3$ is assigned $\frac{1}{6}$. The code space assignments seems consistent in a sense, because those phrases which have been assigned a code space more than once, has had the *same* amount of code space assigned. As we will see shortly, this is not a coincidence.

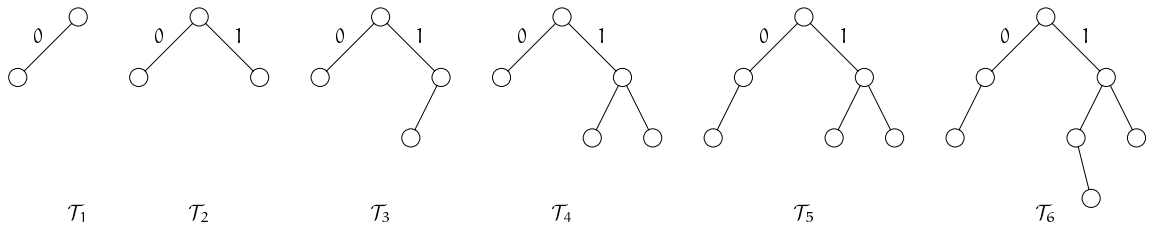
§210 **Observation**. By lemma §166 part (b), we conclude, that the set of phrases of a LZ78 parsing is a *prefix closed set*. That is, if $W$ is the set of phrases, then $uv \in W \Rightarrow u \in W$ where $u$ and $v$ are sequences. It is easy to see, that there is a correspondence between prefix closed sets and $|A|$-ary trees with edges labeled so the edges leaving the same node are labeled with distinct symbols from $A$. For example, let $A = \{a, b, c\}$ and consider the set $W = \{a, b, c, aa, ba, bb, bac\}$. This set is prefix closed and we may draw the following tree:




Each node (except the root) corresponds to an element of $W$ by the path from the root to the node, as indicated by the example "bac".

§211 **Definition** (LZ78 PARSE TREE). Let $x_1^n \in A^n$ and write $x_1^n = w_1 \cdots w_t$ where $w_i$ are the phrases produced by the LZ78 algorithm. For any $1 \leq i < t$ let $\mathcal{T}_i$ be the $|A|$-ary tree corresponding to the (prefix closed) set $\{w_1, \ldots, w_i\}$. The trees $\mathcal{T}_i$ are known as the *LZ78 parsing trees*. Each node in the tree is identified by a phrase, $w_i$, except for the root node which is identified by the empty phrase, $\lambda$.

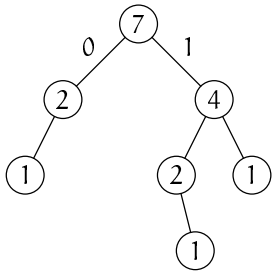
§212 **Example**. Let $x_1^9 = 0, 1, 10, 11, 00, 101$ be a LZ78 parsing. We use this notation when dealing with parsings of a sequence (in this case $x_1^9 = 01101100101$). The commas should not be taken as part of the sequence, but merely as a help to the reader. They are, of course, redundant, because they follows uniquely from the stated parsing method (in this case the LZ78 parsing). More precisely, we have the phrases $w_1 = 0$, $w_2 = 1$, $w_3 = 10$, ..., $w_t = 101$ with $t = 6$ the total number of phrases. The corresponding parsing trees, $\mathcal{T}_1$ through $\mathcal{T}_6$ are drawn below. As indicated by the labels, the left edges corresponds to 0's and the right edges corresponds to 1's.




The first tree ($\mathcal{T}_1$) must, according to the definition, correspond to the set $\{w_1\} = \{0\}$, so it has just a single non-root node. We can think of the sequence of trees as the process of "growing" the final tree[2]. Thus we get $\mathcal{T}_2$ by "adding" the phrase $w_2 = 1$ to the tree $\mathcal{T}_1$. Adding the phrase $w_3 = 10$ to $\mathcal{T}_2$ yields $\mathcal{T}_3$ and so forth. In fact, the parsing trees are a valuable tool in implementing the LZ78 algorithm: Assume that we have just emitted a pair $(i, a)$ corresponding to the phrase $w_i$ and we go to step L2 again; the sequence yet to be encoded is $x_p x_{p+1} \cdots$. We now descent the tree $\mathcal{T}_i$ while at each node selecting the path according to the symbols $x_p x_{p+1} \cdots$. Doing this will bring us to the very leaf corresponding to the *longest* prefix of $x_p x_{p+1} \cdots$ found in $D$, which is exactly what we need in step L2. The next phrase is the sequence of symbols of the path from the root to the node we ended in, plus one additional incoming symbol.

§213 **Definition**. Let $\mathcal{T}$ be a LZ78 parse tree. For any leaf node in $\mathcal{T}$ we assign the value of 1 and for any other node we assign the value of the sum of its children plus 1. The resulting values are known as the *phrase counts* for each node, and the tree with the assigned numbers are known as a *phrase counting tree*. If $d$ is a node in a parse tree, we write $c(d)$ for the phrase count of the node $d$.

§214 **Example**. Consider $\mathcal{T}_6$ from example §212, that is, the sixth parse tree for $0, 1, 10, 11, 00, 101$. Filling in the phrase counts yields the phrase counting tree below:




For the phrase count of the node 10 we write $c(10)$ in this example we have $c(10) = 2$. Note, that the phrase count for the root node in a parse tree equals the number of nodes in the tree, and thus for the $i$'th parse tree, we have $c(\lambda) = i + 1$.

§215 **Algorithm** (SYMBOLWISE SIMULATION OF LZ78). Given a finite alphabet $A = \{0, 1, \ldots, k-1\}$ of $k$ letters and a sequence $x_1^n \in A^n$ of length $n$ to be encoded. We define $x_i^{i-1} = \lambda$. Note, that in the steps of the algorithm, a number of assertions are placed as parenthetical remarks. These assertions are used in the proof of theorem §216.

**Input:** The sequence $x_1^n$ to be encoded.

[2]It is interesting to note, that the last time we encountered the idea of "growing" a tree, was in the proof of theorem §131 (see [Ryabko, 1986]). See observation §241 for further remarks on this.

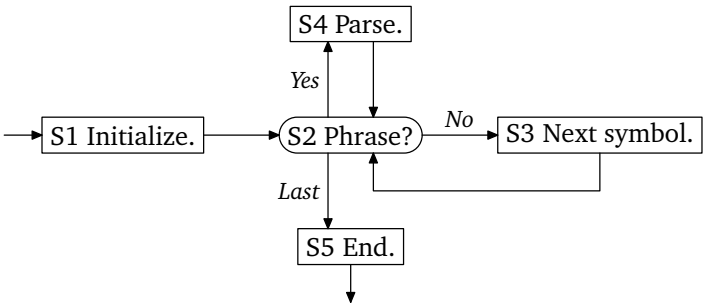



Figure 2: Flowchart for the algorithm for symbolwise simulation of Lempel-Ziv 1978, as presented in §215

**Output:** A sequence of $n$ values, $p(x_i|x_1^{i-1})$, the *symbolwise probabilities*.

**S1:** [Initialize.] Let $\mathcal{T}$ be an empty parse tree, that is, it contains only the root node, namely $\lambda$, having phrase count equal to 1. Set $p \leftarrow 1$ and set $i \leftarrow 1$.

**S2:** [Phrase?] (Now $\mathcal{T}$ contains the LZ78 phrases of $x_1^{p-1}$.) If $i > n$ go to S5. If $x_p^i$ is not a node in $\mathcal{T}$ go to S4.

**S3:** [Next symbol.] Output the value $c(x_p^i)/c(x_p^{i-1})$. Set $i \leftarrow i+1$. (Still, $\mathcal{T}$ contains the LZ78 phrases of $x_1^{p-1}$.) Go to S2.

**S4:** [Parse.] (Now $x_p^i$ is the LZ78 phrase following $x_1^{p-1}$.) Output the value $1/|A|c(x_p^{i-1})$. Add $x_p^i$ to the tree $\mathcal{T}$ and update the phrase counts accordingly. Set $i \leftarrow i+1$. Set $p \leftarrow i$. (Now $\mathcal{T}$ again contains the LZ78 phrases of $x_1^{p-1}$.) Go to S2.

**S5:** [End.] (Here $x_p^n$ is the final LZ78 phrase of $x_1^n$.) Output the value $1/|A|c(x_p^n)$ unless $p > n$. The algorithm terminates.

§216 **Theorem**. *The algorithm §215 produces, in symbolwise fashion, conditional probabilities compatible with the LZ78 code. That is, for any $x_1^n \in A^n$, with the LZ78 parsing $x_1^n = w_1 \cdots w_t$, the following hold:*

$$\prod_{i=1}^{n} p(x_i|x_1^{i-1}) = 2^{-\sum_{j=1}^{t} j \log |A|} \leq 2^{-|\varphi_{LZ78}(x_1^n)|} \tag{15}$$

*where $p(x_i|x_1^{i-1})$ are the output of the algorithm.*

§217 **Proof**: From observation §208 the inequality part of (15) is clear. We now prove the equality of (15). Let $w_j$ be any phrase of the parsing and let $s$ be the position in the message where $w_j$ starts and let $t = s + |w_j| - 1$, that is, $w_j = x_s^t$. We will now prove

$$\prod_{i=s}^{t} p(x_i|x_1^{i-1}) = 2^{-\log j|A|} \tag{16}$$

from which the result follows easily. In order to do this, we must study the relation between the algorithm and the LZ78 parsing. It is quite easy to check the assertions stated (as parenthetical

remarks) in the algorithm; especially if we keep in mind the flowchart of the algorithm shown in figure 2. We note, that the assertion in the beginning of step S4 is true because the only way we can arrive at S4 is from S2, in which case $x_p^i$ is the shortest prefix of $x_p^n$ that is not an already parsed phrase. Returning to the phrase $w_j = x_s^t$, this means that at some point in the execution of the algorithm, we will be in step S2 having $p = s$ and $i = s$. At this point the tree $\mathcal{T}$ will contain all phrases $w_1, \ldots, w_{j-1}$ as nodes. Observe, that the sequence of nodes given by $x_p^i$ for $i$ in the integer interval $[s,t]$, describes a descent in the tree $\mathcal{T}$, at every node $x_p^i$ selecting the next node according to the symbol $x_{i+1}$. The resulting output from the algorithm is a sequence of numbers, $p(x_i|x_1^{i-1})$, for $i \in [s,t]$. Following (16) we multiply these numbers to get

$$\frac{c(x_s^s)}{c(\lambda)} \cdot \frac{c(x_s^{s+1})}{c(x_s^s)} \cdots \frac{c(x_s^{t-1})}{c(x_s^{t-2})} \cdot \frac{1}{c(x_s^{t-1})} \cdot \frac{1}{|A|} = \frac{1}{c(\lambda)|A|},$$

where all of the left hand side factors, except the last two, stems from step S3 in the algorithm, and the last two from step S4. The equality is by the telescopic property of the fractions. By the observation made in example §214 we have

$$\prod_{i=s}^{t} p(x_i|x_1^{i-1}) = \frac{1}{c(\lambda)|A|} = \frac{1}{(j-1+1)|A|}$$

which is the desired result. Finally, we must check that the produced numbers, $p(x_i|x_1^{i-1})$, are indeed legal code spaces, that is, to any $x_1^{i-1} \in A^{i-1}$, we must have $\sum_{a\in A} p(a|x_1^{i-1}) = 1$. Let $w_j$ be any phrase and writing $y_1^k = w_j$, with $k = |w_j|$. We recall that to each $y_i$, $i = 1, \ldots, k$ the output of the algorithm depends on the phrase counts of node $y_1^{i-1}$ and possibly that of $y_1^i$ (if it exists). There are two cases: Firstly, in case the node $y_1^{i-1}$ has all of its $|A|$ possible children, we have by construction of the phrase counting tree

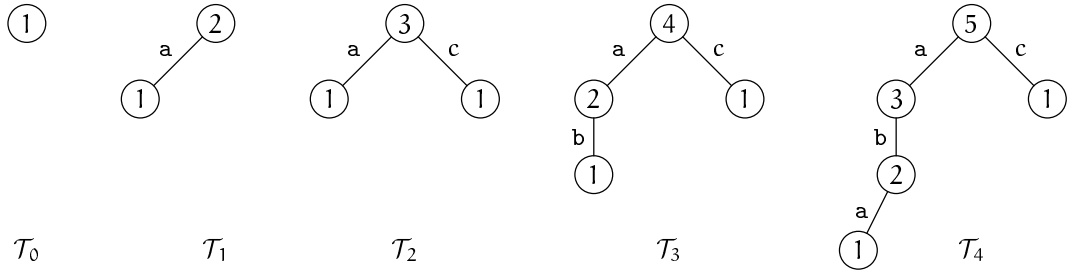
$$\sum_{a\in A} p(a|w_1 \cdots w_{j-1} y_1^{i-1}) = \sum_{a\in A} \frac{c(y_1^{i-1}a)}{c(y_1^{i-1})} = \frac{c(y_1^{i-1}) - 1}{c(y_1^{i-1})} < 1.$$

Secondly, in case the node $y_1^{i-1}$ has less than $|A|$ children (possibly none), we consider the set $B = \{a \in A \mid w_1^{i-1}a \text{ is a node}\}$. We now have

$$\sum_{a\in A} p(a|w_1 \cdots w_{j-1} y_1^{i-1}) = \sum_{a\in B} \frac{c(y_1^{i-1}a)}{c(y_1^{i-1})} + \sum_{a\in A\setminus B} \frac{1}{c(y_1^{i-1})} \cdot \frac{1}{|A|} \leq \frac{c(y_1^{i-1}) - 1}{c(y_1^{i-1})} + \frac{1}{c(y_1^{i-1})} \leq 1.$$

Note, that if $B$ is empty, then $y_1^{i-1}$ is a leaf node and thus $c(y_1^{i-1})$ equals 1 and we have equality. This completes the proof.

§218 **Example**. Let us look at an example of the working of the symbolwise simulation algorithm. Let $A = \{a, b, c\}$ and let acababa be the message to be encoded. The LZ78 parsing is a, c, ab, aba. The sequence of parsing trees is given here:




(17)

With the phrase counting trees worked out beforehand, it is quite easy to follow the algorithm, to get the following output:

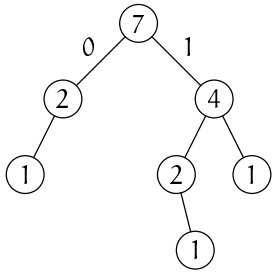
| *Parse tree* | *Step:* $i$ | *Past:* $x_1^{i-1}$ | *Input symbol:* $x_i$ | *Output:* $p(x_i\|x_1^{i-1})$ |
|---|---|---|---|---|
| $\mathcal{T}_0$ | 1 | $\lambda$ | a | $1/c(\lambda)\|A\| = 1/3$ |
| $\mathcal{T}_1$ | 2 | a | c | $1/c(\lambda)\|A\| = 1/6$ |
| $\mathcal{T}_2$ | 3 | ac | a | $c(a)/c(\lambda) = 1/3$ |
| $\mathcal{T}_2$ | 4 | aca | b | $1/c(a)\|A\| = 1/3$ |
| $\mathcal{T}_3$ | 5 | acab | a | $c(a)/c(\lambda) = 1/2$ |
| $\mathcal{T}_3$ | 6 | acaba | b | $c(ab)/c(a) = 1/2$ |
| $\mathcal{T}_3$ | 7 | acabab | a | $1/c(ab)\|A\| = 1/3.$ |

Now let us compare the code space assigned by the symbolwise simulation with the code-lengths for each phrase in the LZ78 encoding.

| *Phrase:* $w_i$ | *LZ78-length:* $\lceil \log i\|A\| \rceil$ | *Probability:* $p(w_j\|w_1 \cdots w_{j-1})$ |
|---|---|---|
| $w_1 = a$ | $\lceil \log 3 \rceil$ | $1/3$ |
| $w_2 = c$ | $\lceil \log 6 \rceil$ | $1/6$ |
| $w_3 = ab$ | $\lceil \log 9 \rceil$ | $\frac{1}{3} \cdot \frac{1}{3} = 1/9$ |
| $w_4 = aba$ | $\lceil \log 12 \rceil$ | $\frac{2}{4} \cdot \frac{1}{2} \cdot \frac{1}{3} = 1/12.$ |

By $p(w_j|w_1 \cdots w_{j-1})$ we mean the product of the symbolwise, conditional code space of $w_j$. Note, that the assigned code space corresponds nicely to that induced by the LZ78-length, prior to rounding up.

§219 **Observation** (LESSON I). Perhaps even more interesting than the existence of a symbolwise simulation of the LZ78 code, is the insight into the workings of LZ78, that such an simulation might give. In this and the following observations, we attempt to derive such insight. Consider the phrase counting tree from example §214, which we repeat here:

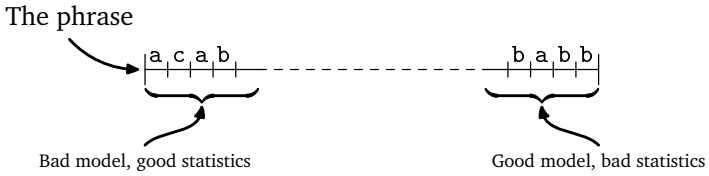



This tree is the result of parsing $0, 1, 10, 11, 00, 101$. The phrase count in the root node is simply the number of phrases (plus one) seen so far. The phrase count in the node "1" (see observation §210 on how the nodes are labeled) is the number of phrases which begins with the symbol 1. There are 4 such phrases, namely 1, 10, 11 and 101, and in the phrase counting tree above, we do indeed find 4 as the phrase count in the node 1. Likewise, the phrase count in the node 11 reflects the number of times we encountered a phrase beginning with 11. In our case we have only one such phrase, namely 11, and correspondingly the phrase count of the node 11 is 1. Thus, in the symbolwise simulation (and hence also implicit in the LZ78 algorithm) the first

symbol of each phrase is encoded using codespace proportional to the number of times we have seen in the past a phrase beginning with this symbol. This means, that when encoding the first symbol of a phrase no use is made of inference from the preceding symbol: the encoding is done memoryless. We can say that the LZ78 algorithm has no inter-phrase inference.

§220 **Observation** (LESSON II). When encoding the last symbol $w_m$ of a phrase, $w_1^m$, there are two different cases. In the first case the node $w_1^{m-1}$ is a leaf node, in which case a codespace of $1/|\mathcal{A}|$ is allotted to the encoding of $w_m$. In the second case, however, when $w_1^{m-1}$ has one or more children, we need to spend $1/|\mathcal{A}|c(w_1^{m-1})$. We may say, that when encoding the symbol following $w_1^{m-1}$ the codespace $1/c(w_1^{m-1})$ is reserved in order to enable us to signify that we move on to a "new symbol", and not one of $w_1^{m-1}$'s children nodes. However, when a node, say $v$, has all of its $k = |\mathcal{A}|$ possible children, say, $u_1, \ldots, u_k$ we still use only $\left(\sum_{i=1}^{k} c(u_i)\right)/c(v) = (c(v) - 1)/c(v)$ to encode which of the children is the next node. Thus we reserve some codespace to an impossible event.

§221 **Observation** (LESSON III). The last lesson which the symbolwise simulation teaches, is that the LZ78 algorithm in essence may collects useless statistics. If encoding output from a memoryless, binary source, it is of no use to have knowledge about, say, the number of times in the past where 0 has followed 1 (this number is $c(10)$) versus the number of times where 0 has followed 0 (this number is $c(00)$). This is a problem of overfitting the model and, as we shall see in the next section, the LZ78 algorithm pays a high price for doing this, when we measure its performance with the memoryless measure. This issue is related to the lack of inter-phrase inference that we considered above in observation §219. There it was seen that the first symbol of each phrase is encoded using no inference. However, it is encoded using very precise statistics because it corresponds to the phrase counts near the root of the parse tree, and here the counts are largest. On the other hand, when encoding the last symbols of a phrase we are close to the leafs in the parse tree and hence use a lot of inference albeit with not so good statistics. We may illustrate the behavior like this:




It is natural to ask if it is possible to identify when there is nothing gained by extending the model, and thus have the encoding algorithm keep the size of the model (the parse tree) bounded and focus on collecting better statistics. These ideas are related to the works of Rissanen on the Context algorithm and the MDL principle, see [Rissanen, 1983, 1986, 1989].

§222 **Observation**. By theorem §216 we see that using the symbolwise code space together with an arithmetic encoder, will yield a codelength strictly smaller than the LZ78 codelength.

§223 **Observation**. Although the symbolwise simulation of LZ78 has shown a number of ways where the code might be improved we must remember, that the LZ78 is a (weakly) universal code on the big class $\mathcal{M}_e(\mathcal{A}^\infty)$; and hence so is the symbolwise equivalent. On the grand scale of all stationary ergodic measures, the problems we have pointed out above is therefore vanishingly small. For any stationary ergodic measure, the implicit estimation of the measure, which is done by the LZ78 algorithm, will therefore *almost surely* converge to the true measure. Seen from this angle, we have in LZ78 a symbolwise entropy estimation method.

§224 **Summary**. In this section we have studied parsing trees for the LZ78 parsing and used them to construct a symbolwise simulation of the LZ78 encoding. While working with this material we have gained better insight into the workings of LZ78, in particular how we may see the LZ78 as a device collecting statistics for a model of dynamically growing complexity. Finally we have identified a number of ways where the LZ78 encoding might be improved.

## Second order analysis

§225 **Remark**. Having acquainted ourselves with the parse trees of the LZ78 algorithm, we may now give a rather easy proof of an important result about the LZ78 code, namely a second order bound on its performance. The term *second order analysis* is, in this context, used to describe the study of how an universal (on the class of stationary ergodic sources) code perform when applied to a much smaller class. More precisely, we will consider the performance of $\varphi_{LZ78}$ on the class of memoryless sources. Alternatively, we may say that we study the behavior of the redundancy of the LZ78 code with respect to i.i.d. measures.

§226 **Observation** (CODEWORD LENGTHS). Recall, (or see observation §169) that encoding of each phrase, $w_j$, in the LZ78 scheme uses $\lceil \log |A| j \rceil$ bits. Let $x_1^n$ be a sequence, and assume it is parsed into $\tilde{t}$ phrases, $x_1^n = w_1 \cdots w_{\tilde{t}}$, by the LZ78 algorithm. Let $t$ be the number of distinct phrases, that is $t = \tilde{t}$ or $t = \tilde{t} - 1$. The codelength of $x_1^n$ is given by

$$|\varphi_{LZ78}(x_1^n)| = \sum_{i=1}^{\tilde{t}} \lceil \log i|A| \rceil$$

A string of calculations (that are a variation on observation §175, (3) but with a focus on the number of distinct phrases) show that

$$\begin{aligned} |\varphi_{LZ78}(x_1^n)| &\leq \tilde{t}((\log \tilde{t}|A|) + 1) \\ &\leq \tilde{t}(1 + \log |A|) + (t+1)\log(t+1) \\ &\leq \tilde{t}(1 + \log |A|) + t \log t + (t+1)\log(1 + \tfrac{1}{t}) + \log t \\ &\leq \tilde{t}(1 + \log |A|) + t \log t + t + 1 \end{aligned}$$

This analysis is far from tight, but for our task it will do. Next is a lemma of rather technical nature.

§227 **Lemma**. *Let* $A = \{0, \ldots, k-1\}$ *be an alphabet. Let* $B \subseteq A^*$ *be a prefix-closed set of* $t$ *elements and partition* $B$ *into* $B_0 \stackrel{\text{def}}{=\!=} \{b \in B \mid \forall a \in A : ba \in B\}$ *and* $B_1 \stackrel{\text{def}}{=\!=} \{b \in B \mid \exists a \in A : ba \notin B\}$. *There exists an injective mapping* $f : B_0 \to B_1$ *such that the following hold*

$$\begin{aligned} &1^\circ \quad f(b) = bx_b, \ x_b \in A^* \quad \textit{for any } b \in B_0. \\ &2^\circ \quad \textstyle\sum_{b \in B_0} |x_b| \leq |B| \end{aligned}$$

§228 **Proof**: It is clear, that the sets $B_0$ and $B_1$ form a disjoint partition of $B$. We now proceed by induction on the size $t = |B|$ of the set $B$. Observe, that if $t \leq k+1$, then $B_0$ is either empty or contains exactly one element. In the first case, the lemma is trivially true. In the latter case, let $b$ be the element of $B_0$. Then the $k$ elements $ba_1, \ldots, ba_k$ belong to $B$ and thus $B$ must be of

the form $\{b, ba_1, \ldots, ba_k\}$ with $|b| = 1$. For if $|b| > 1$ then the prefixes of $b$ would also need to be in $B$, but since $|B| \leq k+1$ that is impossible. The lemma is easily seen to hold in this case for example by setting $f(b) = ba_1$.

Now, for the induction step let $B \subset A^*$ be a prefix-closed set with $|B| = t$ and assume the lemma holds for any set $\tilde{B}$ with $|\tilde{B}| < t$. We must show the lemma to be true for $B$. We may assume $B_0$ to be non-empty. Pick an element $d \in B$ of maximal length (there might be several possibilities for the choice of $d$.) It is possible to write $d = ca$ for some $a \in A$ and $c \in B$ since $B$ is prefix-closed. New let $\tilde{B} \stackrel{\text{def}}{=\!=} B \backslash cA$, where $cA = \{ca \mid a \in A\}$. By the induction hypothesis the lemma holds for $\tilde{B}$. We proceed in three different cases:

Firstly, assume $\tilde{B}_0$ is empty. We will show that $B_0 = \{c\}$. This is seen by the following argument: Let $b \in B_0$ (which is possible because $B_0$ was assumed to be non-empty) then $bA \subseteq B$ by definition. On the other hand, $bA \not\subseteq \tilde{B}$ because $\tilde{B}_0$ is empty. Thus $bA \cap cA \neq \emptyset$ and this yields $b = c$. This proves that $B_0 = \{c\}$, and setting $f(c) = ca$ shows that the lemma holds for $B$.

Secondly, assume $\tilde{B}_0$ is non-empty and that $c \in B_1$. Then $B_0 = \tilde{B}_0$, because the removal of $cA$ from $B$ will have no impact on the $B_0$-set. The induction hypothesis yields a function $\tilde{f} : \tilde{B}_0 \to \tilde{B}_1$ with the desired properties. This function might be extended to $f : B_0 \to B_1$ because $B_0 = \tilde{B}_0$ and $B_1 \subseteq \tilde{B}_1$. Since $f$ is based on $\tilde{f}$ it is easy to check that $f$ satisfies both $1^\circ$ and $2^\circ$.

Thirdly, assume $\tilde{B}_0$ is non-empty and that $c \in B_0$. Then $c \in \tilde{B}_1$, because $cA$ is not a subset of $\tilde{B}$. The induction hypotheses yields a function $\tilde{f} : \tilde{B}_0 \to \tilde{B}_1$ with the desired properties. We consider two subcases: First, assume $c = \tilde{f}(b)$ for some $b \in \tilde{B}_0$. Since $\tilde{f}$ is injective $b$ is the only element of $\tilde{B}_0$ which maps to $c$. Furthermore, no element of $\tilde{B}_0$ can be mapped by $\tilde{f}$ into an element of the form $cx$ because $d$ was chosen to be of maximal length. Thus in order to have the lemma hold for $B$ we merely need to "fix" $\tilde{f}$ for $b$ and extend it to cover $c$ also. Define

$$f(x) \stackrel{\text{def}}{=\!=} \begin{cases} ca' & \text{for } x = b \\ ca & \text{for } x = c \\ \tilde{f}(x) & \text{otherwise} \end{cases}$$

where $a' \in A$ is some letter, $a' \neq a$ (the existence of a letter $a'$ so that $ca' \in B$ is ensured by the facts $c \in B_0$ and $k > 1$). Because of the maximality of $d = ca$, we know that $f$ maps injectively to $B_1$. The property $2^\circ$ holds because of the following

$$\sum_{b \in B_0} |x_b| \leq \sum_{b \in \tilde{B}_0} |\tilde{x}_b| + 2 \leq |\tilde{B}| + 2 \leq |B|$$

where the first inequality is by the definition of $f$, the second is by the induction hypothesis and the last is due to $c \in B_0$ (which means $|\tilde{B}| + k = |B|$). For the second sub-case, where $c \notin f(B_0)$ we set $f(c) = ca$, and $f(x) = \tilde{f}(x)$ when $x \neq c$. We note that

$$\sum_{b \in B_0} |x_b| = \sum_{b \in \tilde{B}_0} |\tilde{x}_b| + |a| \leq |\tilde{B}| + 1 \leq |B|$$

and that $f$ is injective. This completes the proof.

§229 **Theorem**. *Let $A$ be an alphabet with $|A| = k$ and let $\mu$ be an i.i.d. measure over $A^\infty$. Let*

*$x_1^n \in A^n$ and let $t(x_1^n)$ be the number of phrases in the LZ78 parsing. The following hold*

$$\frac{|\varphi_{LZ78}(x_1^n)| + \log \mu(x_1^n)|}{n} \leq \frac{8}{n}\, t(x_1^n) \max_{a\in A}\{-\log \mu(a)\}.$$

§230 **Proof**: Let $B$ be the set of distinct phrases, $B = \{w_1, \ldots, w_t\}$, meaning that either $t = t(x_1^n)$ or $t+1 = t(x_1^n)$. Note that $B$ is prefix-closed. The sets $B_0$ and $B_1$ are defined as in the lemma above. Furthermore, let $p_{\min} \stackrel{\text{def}}{=\!=} \min\{\mu(a) \mid a \in A\}$. The first step is to use the above lemma to get an upper bound on the sum of the measures of the phrases,

$$\sum_{b\in B} \mu(b) = \sum_{b\in B_0} \mu(b) + \sum_{b\in B_1} \mu(b).$$

We start by estimating the sum over $B_1$. By nature of the definition of $B_1$ we may for any $b \in B_1$ pick a letter $a_b$ such that $ba_b \notin B$. The set $C = \{ba_b \mid b \in B_1\}$ is then a prefix-set because if $ba_b, da_d \in C$ and $ba_b$ is a non-trivial prefix of $da_d$ (that is, $ba_b \neq da_d$), we find $ba_b$ to be a prefix of $d$ which is impossible because $d \in B$ (and hence every prefix of $d$ is in $B$ as well.) The measure $\mu$ is consistent, so we have

$$1 \geq \sum_{b\in B_1} \mu(ba_b) = \sum_{b\in B_1} \mu(b)\mu(a_b) \geq p_{\min} \sum_{b\in B_1} \mu(b)$$

and then

$$\sum_{b\in B_1} \mu(b) \leq \frac{1}{p_{\min}}. \tag{18}$$

We then continue to estimate the sum over $B_0$. By the lemma just proved, we have an injective mapping $f : B_0 \to B_1$, that is,

$$\sum_{b\in B_0} \mu(f(b)) \leq \sum_{b\in B_1} \mu(b) \tag{19}$$

Combining the two estimates (18) and (19) from above, we get

$$\sum_{b\in B_0} \mu(f(b)) + \sum_{b\in B_1} \mu(b) = \sum_{b\in B} \mu^*(b) \leq \frac{2}{p_{\min}}$$

where

$$\mu^*(b) = \begin{cases} \mu(f(b)) & \text{if } b \in B_0 \\ \mu(b) & \text{if } b \in B_1 \end{cases}$$

Observe, that $\sum_{b\in B} \mu^*(b) \leq 2/p_{\min} \leq \sum_{b\in B} 2/tp_{\min}$, and apply the log-sum inequality (see lemma §276) to the two sums to get first

$$\sum_{b\in B} \frac{2}{tp_{\min}} \log \mu^*(b) \leq \sum_{b\in B} \frac{2}{tp_{\min}} \log \frac{2}{tp_{\min}}$$

and then

$$0 \leq \sum_{b\in B} \left(\log 2 - \log p_{\min} - \log t - \log \mu^*(b)\right).$$

Setting $l_{max} \stackrel{\text{def}}{=\!=} -\log p_{min}$ and writing $f(b)$ as $bx_b$ (which is perfectly legal due to $1^\circ$ in the lemma), we get

$$-\log \mu^*(b) = \begin{cases} -\log \mu(b) - \log \mu(x_b) & \text{if } b \in B_0 \\ -\log \mu(b) & \text{if } b \in B_1 \end{cases}$$

From this follows easily that

$$t \log t \leq t(l_{max}+1) + \sum_{b\in B_1} -\log \mu(b) + \sum_{b\in B_0} -\log \mu(b) + \sum_{b\in B_0} -\log \mu(x_b)$$

where the last sum is bounded by $tl_{max}$. (This follows from $2^\circ$ in the lemma.) We conclude that the following bound hold

$$t \log t \leq \left( \sum_{b\in B} -\log \mu(b) \right) + t(2l_{max}+1).$$

The observation about the length of the LZ78 code (see observation §226) now yields

$$\begin{aligned} |\varphi_{LZ78}(x_1^n)| &\leq t(x_1^n)(1+\log|A|) + \left( \sum_{b\in B} -\log \mu(b) \right) + t(2l_{max}+1) + t + 1 \\ &\leq t(x_1^n)2l_{max} - \log \mu(x_1^n) + t(x_1^n)2l_{max} + 2t(x_1^n) + 1 \\ &\leq -\log \mu(x_1^n) + 6t(x_1^n)l_{max} + 1 \end{aligned}$$

(where we used that $\log k \leq l_{max}$ and $1 \leq l_{max}$) so that

$$\frac{|\varphi_{LZ78}(x_1^n)| + \log \mu(x_1^n)|}{n} \leq \frac{7}{n} t(x_1^n) l_{max}.$$

This completes the proof.

§231 **Corollary**. *Let $x_1^n \in A^n$ and let $\mu$ be an i.i.d. measure on $A^\infty$. Then*

$$\frac{|\varphi_{LZ78}(x_1^n)| + \log \mu(x_1^n)}{n} = O(\frac{1}{\log n})$$

*as $n \to \infty$.*

§232 **Proof**: By lemma §178 the number of LZ78-phrases of $x_1^n$ is bounded by $O(n/\log n)$ as $n \to \infty$. The corollary follows from the theorem just proved (theorem §229).

§233 **Remark**. Second order results like the one just given, should be compared with similar results for other classes of sources and other codes as well as known theoretical bounds. For a selection of such results we refer to the section on notes and references below. Here, we only note that the best possible convergence rate for the class of i.i.d measures, is $O(n^{-1}\log n)$. The LZ78 code is thus far from optimal when used on the limited class of i.i.d. measures.

§234 **Summary**. We have analyzed the performance of the LZ78 code on the class of i.i.d. measures using simple techniques. We can conclude that LZ78 is far from twice universal (see §159).

## Notes and references

§235 **References**. The basic ideas of the Lempel-Ziv algorithms are, of course, found in the original articles, [Lempel and Ziv, 1976; Ziv and Lempel, 1977, 1978]. We have emphasized the two parts of the algorithm, namely the parsing and the encoding. The analysis of distinct parsings leading to the proof that LZ78 is a weak universal code, is based on [Ornstein and Weiss, 1993], while the interpretation of the LZ78 parsing as a sequential assignment of probability or code space is due to [Langdon, 1983]. For the second order result we relied on [Kieffer and Yang, 1999].

§236 **Notes on the description**. For an additional number of variants and a coverage of the issues of implementing the LZ algorithms, we refer to [Bell et al., 1990]. We also point to Appendix C where an Perl implementation, primarily intended for study purposes, is given.

§237 **Notes on the analysis**. For another proof of the weakly universality of the LZ78 algorithm, see [Cover and Thomas, 1991]. For a result which just proves the existence of such codes see [Davisson, 1973] and [Kieffer, 1978]. Also, [Neuhoff and Shields, 1998] consider the construction of a very simplistic (and not practical implementable) code which is weakly universal. We should also point out the fact that since the LZ77 variant parses the input into distinct phrases (just like the LZ78 does) the distinct parsing theorem (see theorem §197) which was the basis for the universality theorem for LZ78 (see theorem §183), could also be applied to prove weak universality for LZ77.

§238 **Further notes on analysis**. Although a major milestone, the weak universality of LZ78 is not the only way to approach the evaluation of the performance of the code. Unfortunately the different performance criterions are not always easily comparable. In order to show some of the different results, we introduce a bit of notation from [Shields, 1999], which seem to be emerging as a standard naming convention. The LZ78 method we have considered is denoted SLZ (Simple Lempel-Ziv) and we distinguish between two cases: We write $SLZ(\boldsymbol{x}) = \limsup_{n\to\infty} |\varphi_{LZ78}(x_1^n)|/n$ and $SLZ_N(\boldsymbol{x}) = \limsup_{m\to\infty}(1/mN)\sum_{i=0}^{m-1}|\varphi_{LZ78}(x_{iN+1}^{iN+N})|$. Similarly we use ULZ (Unrestricted Lempel-Ziv) to denote the LZ77 variant and write $ULZ(\boldsymbol{x}) = \limsup_{n\to\infty} |\varphi_{LZ77}(x_1^n)|/n$. Finally, we write $SW_N(\boldsymbol{x}) = \limsup_{n\to\infty} |\varphi_{LZ77,N}(x_1^n)|/n$ where the code $\varphi_{LZ77,N}$ is a variant of the LZ77 algorithm where we only look for recurring phrases that are less than $N$ symbols in the past. This variant is known as *sliding window Lempel-Ziv*. The notation $FS(\boldsymbol{x})$ is used here to denote the finite state compressibility (previously written as $\rho(\boldsymbol{x})$.)

We may now list a number of results: In [Ziv and Lempel, 1977] it was seen that $\mu\{\boldsymbol{x} \mid ULZ(\boldsymbol{x}) = H\} = 1$. In [Ziv and Lempel, 1978] it was proved that $SLZ(\boldsymbol{x}) = FS(\boldsymbol{x})$, that $FS(\boldsymbol{x}) = H(\mu)$ almost surely and that $\lim_{N\to\infty} SLZ_N(\boldsymbol{x}) = FS(\boldsymbol{x})$. Surprisingly late, namely in [Wyner and Ziv, 1994] it was seen that $\lim_{N\to\infty} SW_N(\boldsymbol{x}) = H(\mu)$ almost surely. In [Shields, 1999] the different Lempel-Ziv codes are considered when applied to individual sequences. For example it is proved that there is an $\boldsymbol{x} \in \{0,1\}^\infty$ for which $ULZ(\boldsymbol{x}) \le 1/2$ but $SLZ(\boldsymbol{x}) = 1$. This concept of comparing codes on individual sequences has been developed further Boris Ryabko and Joe Suzuki as we will see in the next chapter.

§239 **Notes on the symbolwise simulation**. Even though we have used the term sequential assignment of *code space* it is, of course, entirely possible to think of the assigned values as statistics of the source (see observation §219) or estimated probabilities. These ideas are put to use in

the award winning paper [Feder et al., 1992] where the concept of finite-state predictability is defined and a predictor based on the sequential probability assignment is shown to be optimal in this setting. It should be noted that the problem of universal prediction is not equivalent to the problem of universal coding. Another point worth making is that a symbolwise simulation of the LZ77 is probably not easy to make. At least we cannot make use of the parse trees as the parsing used by LZ77 search for overlapping matches in the past.

§240 **Notes on the second order results**. The second order result of corollary §231 was first published in [Louchard and Szpankowski, 1997]. Before that, the primary second order result was [Plotnik et al., 1992] where it was seen that if we introduce the redundancy

$$R_n(\varphi_{LZ78}) \stackrel{\text{def}}{=\!=} \frac{E_\mu(|\varphi_{LZ78}(x_1^n)|) - H_n(\mu)}{n}$$

we have

$$R_n(\varphi_{LZ78}) = O\left(\frac{\log\log n}{\log n}\right)$$

as $n \to \infty$. This should be compared with the best possible code for finite-order Markov sources. It was shown in [Davisson, 1973] that no code can perform better than

$$R_n(\varphi) = O\left(\frac{\log n}{n}\right)$$

on the Markov sources of finite order. For a balanced view, we should mention that for the class of all stationary, ergodic sources, second order analysis is meaningless, because it was shown in [Shields, 1993] that for any code there are sources for which the redundancy converges to zero at an arbitrarily slow rate.

§241 **Observation** (A LZ78 INSPIRED DIMENSION). During the development of the ideas of symbolwise simulation of the LZ78 encoding, we made a footnote (see example §212) about the connection between the growing of a parsing tree and the proof of theorem §131. An interesting question is if this observation can be used to create a link between the LZ78 parsing and a particular type of covers maybe leading to a LZ78 inspired dimension. These ideas are not new; in [Billingsley, 1965, page 141] a footnote says: *Define a dimension [ like we did with the Hausdorff dimension ] with the infimum restricted to* $\rho$*-coverings by intervals in a given class* $\mathcal{F}$*. Under what conditions on* $\mathcal{F}$ *does this dimension coincide with Hausdorff's? We have proved it does if* $\mathcal{F}$ *is the class of* $r$*-adic intervals, but I know of no general conditions.* Also, in [Shields, 1999, Remark 3] the following question is raised: *Is there some nice entropy concept that corresponds to, say, ULZ compression, as empirical entropy does to optimal finite-state IL compression, or Ziv entropy does to optimal finite-state block-to-block (with vanishing probability of error) compression?*. It is conceivable that a study of Hausdorff like dimensions could help answer these questions.

§242 **History of results**. We close this chapter with a historical overview of the results on Lempel-Ziv algorithms in the form of a time-line of highlights ranging from the invention of the Lempel-Ziv algorithms to the present. As the works in this area is without question far too big to list, we need not apologize the fact, that we have selected a small subset according to this authors personal taste.

1976 The concept of complexity of individual sequences $x \in A^\infty$ is considered in [Lempel and Ziv, 1976].

1977 The LZ77 code emerge in [Ziv and Lempel, 1977].

1978 The LZ78 code is published in [Ziv and Lempel, 1978] and proven to be universal on the class of stationary ergodic sources as well as universal when competing with finite state encoders.

1983 Based on ideas in [Rissanen and Langdon, 1981] it is shown in [Langdon, 1983] that the Lempel-Ziv parsing (LZ78) can be considered as a sequential probability assignment.

1984 The well known LZW variation appears in [Welch, 1984].

1986 The *Context* algorithm is proposed in [Rissanen, 1986, 1983] as an attempt to fix some of the weaknesses in LZ78. It is somewhat twice-universal.

1991-4 New proofs of the universality on the stationary ergodic sources of LZ78/77 emerge in [Cover and Thomas, 1991] (LZ78), [Ornstein and Weiss, 1993] (LZ77 and LZ78), and [Wyner and Ziv, 1994] (LZ77).

1992 Second order results begin to appear. In [Plotnik et al., 1992] the subclass of finite state sources is considered.

1992 In the award winning paper [Feder et al., 1992] the ideas of using Lempel-Ziv parsing for probability assignment is employed as a method for universal prediction.

1997 Finally, [Louchard and Szpankowski, 1997] solves the problem of second order analysis of LZ78 applied to a Bernoulli source! Simultaneously the problem for Markov sources is solved by [Savari, 1997].

1998 Savari continues her work, this time on LZ77 and a number of other variants, in [Savari, 1998].

1999 A comparison of the different variations in the non-probabilistic case is carried out in [Shields, 1999].

2000 In a sequence of articles, [Kieffer and Yang, 2000; Kieffer et al., 2000], Kieffer and Yang explore the new class of Grammar Based Codes, and proposes a new code, known as MPM.

Present Ryabko and Suzuki, (and Shields et al.) compares the performance of codes on infinite sequences and uses Hausdorff dimension to estimate the size of these sets. Still many open problems on second order analysis of MPM.

*In one sense, there are five* 00*'s in* 0000000000*; in another, there are nine.*
BILL GOSPER IN [BEELER ET AL., 1972, ITEM 176]

CHAPTER IV

# A new comparison method

*"Yes, let's consider," said Bruno, putting his thumb into his mouth again, and sitting down upon a dead mouse.*
*"What do you keep that mouse for?" I said. "You should either bury it or else throw it into the brook."*
*"Why, it's to measure with!" cried Bruno. "How ever would you do a garden without one? We make each bed three mouses and a half long, and two mouses wide."*
*I stopped him as he was dragging it off by the tail to show me how it was used.*

LEWIS CARROLL

## Introduction

There are quite a number of different codes all of which are weakly universal on the class of the stationary ergodic sources. Of course, being weakly universal codes, their performance are asymptotically equal: they reach the entropy of the source, the theoretical bound. However, these codes need not be equally well in adapting themselves to subclasses of the stationary ergodic sources. As we saw in the previous chapter, the second order analysis of weakly universal codes are an active research area, and different codes have different performance in this respect. But there are other ways to distinguish between the performance of the codes. Rather than applying the codes to smaller classes of sources, we might try the opposite, namely to apply the codes to classes of sources even larger than the stationary, ergodic sources. From our treatment in chapter II of combinatorial sources, we know that well defined theoretical bounds exists for this immense family of sources. In the present chapter we will put forward some ideas, originally due to Boris Ryabko, Joe Suzuki and others, in this regard.

The form of this chapter is different from the two preceding ones, in that we present an overview of the results rather than detailed presentations of theorems and proofs.

## Coding of individual sequences

§243 **Observation**. The idea of coding of individual sequences is not new. The titles of [Ziv, 1978], *"Coding Theorems for Individual Sequences"*, and [Ziv and Lempel, 1978], *"Compression of Individual Sequences via Variable-Rate Coding"*, are proof of this. What should be understood by the

term individual sequences is quite clear: we assume $x \in A^\infty$ and consider coding of $x$. It is tempting to go ahead and define a coding problem, but we must be careful in order to avoid certain formulations which leads to trivial solutions. For example, to any $x \in A^\infty$ there is a code $\varphi_\infty$ such that observing just one bit of the codeword $\varphi_\infty(x)$ allows us to decode all of $x$. An example of this is the code defined as

$$\varphi_\infty(y) = \begin{cases} 0y & \text{if } y = x \\ 1y & \text{otherwise} \end{cases}$$

Of course this is an uninteresting case; we might even say it is cheating, since the code, to put it in simple terms, learns $x$ by heart and does not even try to make a real compression. A possible remedy is to limit the class of allowable codes. For example, recall (from definition §201) that the definition of the finite state compressibility $\rho(x)$ was based on the $s$-state compressibility, $\rho_{FS(s)}(x)$. The latter was defined like this,

$$\rho_{FS(s)}(x) = \limsup_{n\to\infty} \min_{\varphi_n \in FS(s)} \frac{|\varphi_n(x_1^n)|}{n}.$$

where we note that the minimum is over all finite state encoders that can be made using at most $s$ states. If the minimum was taken over all codes the compressibility would equal zero for all $x \in A^\infty$. Putting limitations on the class of allowable codes resolves the learn-by-heart problem and yields a usable compressibility concept and hence a coding problem. We might even argue that these limitations are reasonable also from a practical point of view, because they reflect real situations where we have limited resources (e.g. limited memory, limited computer time etc.) However, we will not pursue this kind of coding problems any further; this brief description has been included because it illustrates some of the problems associated with the coding of individual sequences.

§244 **Observation**. Another way of getting a meaningful setup based on individual sequences is to study codes that are from the outset known to be optimal in some way, for example weakly universal codes. Since the codes are optimal (and hence of equal performance in one sense,) it becomes interesting to study sets of sequences where the performance of the codes differ. This approach is analogous to what we did in the second order analysis, where we also worked with a weakly universal code and studied its performance in an area where we really could not expect (and did not get) optimal performance. Rather than apply the weakly universal code to a smaller class of sources, we may consider how it performs on sequences that are not even typical for any stationary ergodic measure. This idea was forwarded in [Shields, 1999] (see §238 for some remarks on the notation used in the article as well as below) where the restarted version of the Lempel-Ziv 1978 ($SLZ_N$) and the sliding window version of Lempel-Ziv 1977 ($SW_N$) were studied. We repeat the definitions here,

$$SLZ(x) = \limsup_{n\to\infty} \frac{|\varphi_{LZ78}(x_1^n)|}{n}, \quad SLZ_N(x) = \limsup_{m\to\infty} \frac{\sum_{i=0}^{m-1} |\varphi_{LZ78}(x_{iN+1}^{iN+N})|}{mN}$$

and

$$ULZ(x) = \limsup_{n\to\infty} \frac{|\varphi_{LZ77}(x_1^n)|}{n}, \quad SW_N(x) = \limsup_{n\to\infty} \frac{|\varphi_{LZ77,N}(x_1^n)|}{n},$$

where $\varphi_{LZ77,N}$ denotes the LZ77 encoding method where only N symbols of the past are taken into account when determining the next phrase. The sliding window version of LZ77 and the

restarted version of LZ78 are known as *finite memory* variations, while the ULZ and SLZ (the latter being equal to the LZ78 version studied in the previous chapter) are known as variations of *unbounded* or *infinite memory*.

§245 **Theorem**. *For any* $x \in A^\infty$ *we have*

$$\limsup_{N\to\infty} SLZ_N(x) = \limsup_{N\to\infty} SW_N(x) = FS(x) \tag{1}$$

*where* $FS(x)$ *is the finite state complexity of* $x$. *Furthermore, there exists a sequence* $x \in A^\infty$ *such that*

$$ULZ(x) = 1/2 \quad \textit{and} \quad FS(x) = 1, \tag{2}$$

*where* $ULZ(x) = \limsup_{n\to\infty} |\varphi_{LZ77}(x_1^n)|/n$.

§246 **Proof**: See [Shields, 1999].

§247 **Observation**. We should note, firstly, that the result is purely non-stochastic in that it holds for every infinite sequence. Of course we knew already that for a set of measure 1 the equality (1) holds true, but this does not guarantee that the codes performs equally well on *all* sequences, so the result is really new. Secondly, we must accept that the result does *not* tell us anything about which of the codes (restarted LZ78 or Sliding Window LZ77) performs best in practice, on the contrary, in the words of Shields, *"... [the result (1)] only says that one cannot use limiting arguments to treat such questions."* Thirdly, as equation (2) shows, the infinite memory version of one variation of the Lempel-Ziv code (LZ77) performs significantly better than the restarted versions; but it is only shown for one particular sequence. In the previous chapter we studied the LZ78 variation of unbounded memory, so it is natural to ask how this code performs (on infinite sequences) compared to the infinite memory version of LZ77. This question was raised in [Shields, 1999] and a partial answer was given in [Pierce II and Shields, 1998],[1] where the following theorem was proved.

§248 **Theorem** (PIERCE–SHIELDS). *There is a set* $C \subset A^\infty$ *of sequences such that for* $x \in C$ *we have* $SLZ(x) = 1$ *and* $ULZ(x) = 0$.

§249 **Remark**. We will not prove the theorem rigorously but we give a sketch of the proof below (in proof §255). However, in order to do so, we must introduce the concept of DeBruijn sequences.

§250 **Definition**. Let $A$ be the binary alphabet $\{0,1\}$ and let $n \in \mathbb{N}$. A *DeBruijn sequence* is any sequence $x_1^N \in A^N$ of length $N = 2^n$ such that $f_c(a_1^n|x_1^N) = 1$ for all $a_1^n \in A^n$ where $f_c(a_1^n|x_1^N)$ denotes the *cyclic frequency* of $a_1^n$ relative to $x_1^N$, that is,

$$f_c(a_1^n|x_1^N) = \left|\{i \in [1,N] \mid \overline{x}_i^{i+n-1} = a_1^n\}\right|, \quad \text{with } \overline{x} = xx.$$

§251 **Example**. For $n = 2$ that is, $N = 2^2 = 4$, there is only one (up to rotation) DeBruijn sequence, namely 0011. Note, that the rotations, 0110, 1100 and 1001, are also DeBruijn sequences. We may think of a DeBruijn sequence as one for which sliding a window of length $n$ across the sequence reveals all possible sequences of length $n$ exactly once, i.e.

$$[00]11 \quad 0[01]1 \quad 00[11] \quad 0]01[1$$

where the final window is 'wrapped around' the edges, so to speak.

§252 **Theorem** (DEBRUIJN). *For $n \in \mathbb{N}$ there are (up to rotation) $2^{2^{n-1}-n}$ DeBruijn sequences of length $2^n$.*

§253 **Proof**: See e.g. [Hall, Jr., 1967, Chapter 9].

§254 **Observation**. We observe, that there are $2^{2^{n-1}-n}2^n = 2^{\frac{1}{2}N}$ DeBruijn sequences (including rotations) of length $N = 2^n$. In some sense this means that the number of DeBruijn sequences is comparable to the entropy typical sequences of a source with entropy $\frac{1}{2}$. In other words, it is impossible for any code $\varphi$ to compress DeBruijn sequences such that $|\varphi(x_1^N)|/N < 1/2$.

§255 **Proof (of theorem §248)**: We will prove only the existence of one sequence $\mathbf{x} \in C$ by direct construction. See [Pierce II and Shields, 1998] for further details. We begin by letting $k \in \mathbb{N}$ and then seek to construct a sequence $x_1^{k2^k} \in A^{k2^k}$ such that LZ77 can compress the sequence well but such that LZ78 cannot compress it well. In the example below (example §256) we shows the construction for $k = 3$. The sequence has the general form

$$x_1^{k2^k} = b_1 b_2 \cdots b_k$$

where each block $b_i$ is a DeBruijn sequence of length $2^k$. In fact, each $b_i$ is in essence the *same* DeBruijn sequence, that has been circularly shifted.

Let $b_1$ be a DeBruijn sequence of length $2^k$, the existence of which is ensured by DeBruijn's theorem. Partitioning $b_1$ into blocks of length $k$ yields $\lfloor 2^k/k \rfloor$ distinct (by the DeBruijn property) blocks of length $k$ and possibly one block of length $r$ with $1 \leq r < k$. Let us first treat the case where $2^k/k$ is an integer, and hence no additional block of length $r < k$ is needed. Shifting $b_1$ cyclically one position to the left yields another DeBruijn sequence $b_2$ and the partition of $b_2$ into blocks of length $k$ yields distinct blocks (by the DeBruijn property) and the blocks are distinct from the $k$-blocks of $b_1$ (since $b_2$ is essentially the same DeBruijn sequence as $b_1$). We may continue this up to $b_k$. (Shifting $b_k$ one position would of course give a sequence that if partitioned into $k$-blocks would yield the same blocks as $b_1$, albeit in a different order.) We may then construct the wanted sequence as $x_1^{k2^k} = b_1 \cdots b_k$. For the second case, where $b_1$ is partitioned into $k$-blocks plus an additional $r$-block, we know by the DeBruijn property that by extending the last block cyclically we get a distinct $k$-block, that is, the last $r$ symbols of $b_1$ together with the first $k - r$ symbols of $b_1$ forms a $k$-block distinct from the rest. Hence, we let $b_2$ be a copy of $b_1$ and observe that partitioning $b_1 b_2$ into $k$-blocks will give distinct $k$-blocks plus possibly an additional block of length $1 \leq r < k$. If such an extra block exists, we continue as above by forming $b_1 b_2 b_3$ and so on until $b_1 b_2 \cdots b_i$ is partitioned into $k$-blocks with no remaining block. If $i = k$ we are done. Otherwise shift $b_1$ to the left until the $k$-prefix of it has not been seen as a $k$-block before. (This can be done using less than $k$ shifts because $i < k$.) Let us name the sequence we get by shifting $b_1$ by $b_{i+1}$. We may continue the above procedure until we have selected $b_1 \cdots b_k$ which is the desired sequence $x_1^{k2^k}$. In particular, if we concatenate the sequences $x_1^{k2^k}$ that was generated for each $k$, we get a (infinite) Champernowne sequence, $\mathbf{x}$, and hence $SLZ(\mathbf{x}) = 1$ (see observation §177). On the other hand, note that for any $k$, each of the sequences $b_i$ of length $2^k$ constructed above, share a subsequence of length $2^k - k$. This

[1]The article [Shields, 1999] was submitted for review in February 1997 and the paper [Pierce II and Shields, 1998] was presented at a conference in 1998. Hence there is no causal paradox here.

is because each $b_i$ stems from the same DeBruijn sequence that has been shifted no more than $k$ places. Hence the LZ77 can encode a $(2^k - k)/2^k$ fraction of the sequences $b_i$ for $i > 1$ using $O(\log 2^k)$ bits. In total the sequence $b_1 \cdots b_k$ which has length $k2^k$ is encoded using $2^k + k(k + O(\log 2^k))$ bits so that $\mathcal{ULZ}(\mathbf{x}) = 0$.

§256 **Example**. This example illustrates the construction of the sequence $x_1^{k2^k}$ as described in the proof above. For $k = 3$, we have the following DeBruijn sequence of length $2^k$,

$$00010111,$$

which we may conveniently name $b_1$. Partitioning $b_1$ in 3-blocks yield $000, 101, 11$, that is, two 3-blocks and a remaining block of length 2. According to the procedure given above, we now let $b_2 = b_1$ yielding a sequence $b_1 b_2 = 000, 101, 110, 001, 011, 1$ where the partition ends with a remaining block of length 1. Continuing with $b_3 = b_1$ we are done because $k = 3$ so that we have constructed $x_1^{24} = b_1 b_2 b_3$ as follows,

$$x_1^{24} = b_1 b_2 b_3 = 000, 101, 110, 010, 111, 001, 011, 100,$$

where we have inserted commas for every $k = 3$ symbol. We may say, that based on the DeBruijn sequence $b_1$ (containing by the DeBruijn property $2^k$ different *overlapping* $k$-blocks,) we have constructed the sequence, $x_1^{24}$, which contains $2^k$ different *non-overlapping* $k$-blocks. This makes $x_1^{24}$ difficult to handle for the LZ78 algorithm as was seen in observation §177. Even though the result holds only asymptotically, we can see the effect when considering the LZ78 and LZ77 parsings of $x_1^{k2^k}$. They are as follows,

$$\begin{aligned} &\text{LZ77:} \quad 0, 001, 011, 100, 10111001011100 \\ &\text{LZ78:} \quad 0, 00, 1, 01, 11, 001, 011, 10, 010, 111, 00. \end{aligned}$$

§257 **Observation**. The DeBruijn theorem (see theorem §252) shows that there are many different DeBruijn sequences. The construction made in the proof of theorem §248 will therefore produce many sequences, so that the set $C$ may seem large. However, in [Pierce II and Shields, 1998] it is stated that the set has measure 0 when measured with any stationary ergodic measure.

## The invisible set

§258 **Observation**. The results seen in this chapter so far has been limited to finding single sequences exhibiting interesting behavior, for example by being especially hard to compress. A natural question to ask is how *many* (measured in a reasonable way, for example by a probability measure or the Hausdorff dimension) sequences do there exists having a certain property. A fundamental class of sequences to treat is those sequences that are compressible. We know that if we consider stationary ergodic measures, any measure (of entropy less than 1) has a set of typical sequences that are compressible and has measure 1. Also we know that this set has Hausdorff dimension equal to the entropy of the measure. Thus, a new question is whether or not there exist sequences that are not typical for any stationary ergodic measure (in which case the sequence would of course be compressed to the theoretical limit by any weakly universal code) yet still compressible. It turns out that there are plenty of such sequences as seen in [Hojo

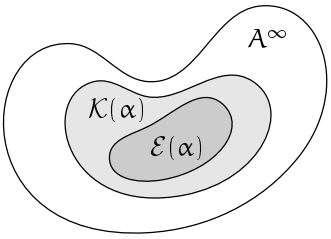



Figure 1: *Theorem §259 shows that the set $\mathcal{E}(\alpha)$ of sequences that are typical for some stationary ergodic measure of entropy at most $\alpha$, is a subset of the set $\mathcal{K}(\alpha)$ of sequences that are compressible (in the sense of Kolmogorov complexity) at the rate $\alpha$ or less. The size of the light gray area, $\mathcal{K}(\alpha)\backslash\mathcal{E}(\alpha)$, is seen to be at least as big as the dark gray area, $\mathcal{E}(\alpha)$, when measured with the Hausdorff dimension.*

et al., 2001], where the following theorem is presented. (It should be noted, that propositions (a) to (c) in the theorem below are well known; the result proved in the paper is proposition (d).) In the formulation of the theorem we use the notion of Kolmogorov complexity of a finite sequence, $K(x_1^n)$. The definition of $K$ can be found in [Li and Vitanyi, 1993].

§259 **Theorem**. *Let $A = \{0,1\}$ be the binary alphabet. Let $\mathcal{E}(\alpha) = \{x \in \mathcal{E} \mid H(\mu_x) \leq \alpha\}$ be the set of sequences that are typical for a stationary ergodic measure of entropy less than $\alpha$. Let $K(\mathbf{x})$ be the Kolmogorov compressibility defined as $K(\mathbf{x}) = \liminf_{n\to\infty} K(x_1^n)/n$ where $K(x_1^n)$ is the Kolmogorov complexity of $x_1^n$, and let $\mathcal{K}(\alpha) = \{\mathbf{x} \in A^\infty \mid K(\mathbf{x}) \leq \alpha\}$. The following hold:*

*(a)* $\mathcal{E}(\alpha) \subset \mathcal{K}(\alpha)$.

*(b)* $\dim_H(\mathcal{E}(\alpha)) = \alpha$.

*(c)* $\dim_H(\mathcal{K}(\alpha)) = \alpha$.

*(d)* $\dim_H(\mathcal{K}(\alpha)\backslash\mathcal{E}(\alpha)) = \alpha$.

§260 **Proof (partial)**: The details of the proof can be found in [Hojo et al., 2001]. Here we present only the main idea by showing proposition (d) for $\alpha = 1/2$. We do this by proving the existence of a set $\mathcal{C}$ of sequences that are Kolmogorov compressible at rate $1/2$, that is, $\mathcal{C} \subseteq \mathcal{K}(1/2)$, yet such that no sequence of $\mathcal{C}$ is typical for any stationary ergodic measure, that is, $\mathcal{C} \cap \mathcal{E} = \emptyset$. Furthermore, we show that $\dim_H(C) \geq 1/2$ which leads to the proposition (d) for $\alpha = 1/2$.

Let $\mu$ be the i.i.d. measure on $A^\infty$ with $\mu(0) = \mu(1) = 1/2$ and write $\mathcal{N} = \mathcal{T}(\mu)$. We now describe the set $\mathcal{C}$: Let $n_1 = 2$ and in general $n_i = i(n_1 + \cdots + n_{i-1})$ and note, that we then have $\lim_{i\to\infty}(n_1 + \cdots + n_{i-1})/(n_1 + \cdots + n_i) = 0$. We then define

$$\mathcal{C} = \{\underbrace{x_1 \cdots x_{n_1}}_{n_1}\underbrace{0\cdots 0}_{n_1}\underbrace{x_{n_1+1}\cdots x_{n_1+n_2}}_{n_2}\underbrace{0\cdots 0}_{n_2}\cdots \mid \mathbf{x} \in \mathcal{N}\}$$

In words, each element of $\mathcal{C}$ is spliced from blocks of growing length. For each block-size one block of white noise (the bits from $\mathbf{x}$) and one block of zeros is concatenated to form the element of $\mathcal{C}$. Let $\mathbf{y} \in \mathcal{C}$, that is, $\mathbf{y} = x_1^{n_1}0^{n_1}x_{n_1+1}^{n_1+n_2}0^{n_2}\cdots$ where $0^m$ is short for $m$ zeros in a row. Let

$m_1 = n_1$, $m_2 = 2n_1 + n_2$, $m_3 = 2(n_1 + n_2) + n_3$ and so forth, that is, $m_i$ is the index in $\mathbf{y}$ where the i'th block of white noise ends. We now have

$$\limsup_{n\to\infty} p_1(1|y_1^n) \geq \limsup_{i\to\infty} p_1(1|y_1^{m_i}) = \limsup_{i\to\infty} \frac{\sum_{j=1}^{n_1+\cdots+n_i} x_j}{2(n_1+\cdots+n_{i-1})+n_i} = \frac{1}{2}$$

where the last equality follows since $2(n_1 + \cdots + n_{i-1}) + n_i = (n_1 + \cdots + n_i)(1 + (n_1 + \cdots + n_{i-1})/(n_1 + \cdots + n_i))$. By a similar argument we get

$$\liminf_{n\to\infty} p_1(1|y_1^n) \leq \liminf_{i\to\infty} p_1(1|y_1^{k_i}) = \liminf_{i\to\infty} \frac{\sum_{j=1}^{n_1+\cdots+n_i} x_j}{2(n_1+\cdots+n_i)} = \frac{1}{4}.$$

We conclude that $\mathbf{y} \notin \mathcal{E}$.

It is easily seen that $\mathbf{y} \in \mathcal{K}(1/2)$ because let $s_i = n_1 + \cdots + n_i$ then a computer program can be made which produces $y_1^{2s_i}$ from $x_1^{s_i}$ so that $K(y_1^{2s_i}) = K(x_1^{s_i}) + O(1)$ as $i \to \infty$. This shows that $\liminf_{n\to\infty} K(y_1^n)/n \leq 1/2$.

Finally, let $\mathbf{y} \in \mathcal{C}$ and consider the number of bits in $y_1^n$ which originally come from $\mathbf{x}$. Let this number be $d(n)$ (it does not depend on $\mathbf{y}$). We have for all $n \in \mathbb{N}$ that $2s_{i-1} \leq n \leq 2s_i$ for suitable $i$. It is now seen that

$$\text{For } 2s_{i-1} \leq n \leq 2s_{i-1} + n_i: \quad \frac{d(n)}{n} = \frac{s_{i-1} + n - 2s_{i-1}}{n} = 1 - \frac{s_{i-1}}{n} \geq \frac{1}{2}$$

$$\text{For } 2s_{i-1} + n_i \leq n \leq 2s_i: \quad \frac{d(n)}{n} = \frac{s_i}{n} \geq \frac{1}{2}$$

where the inequality in the first line becomes an equality when $n = 2s_i$ for some $i$. Thus we have shown that $\liminf_{n\to\infty} d(n)/n = 1/2$, that is, asymptotically at least half of the bits of any prefix of $\mathbf{y}$ originates from $\mathbf{x}$. We now assume $\dim_H(\mathcal{C}) < 1/2$. By the coding theorem for combinatorial sources (see theorem §131) there is a code $\varphi_\infty$ such that $c(\varphi_\infty, \mathcal{C}) = \beta < 1/2$. In the construction of $\mathcal{C}$ we implicitly defined a map from $\mathcal{N}$ into $\mathcal{C}$. Let this map be $\psi_\infty$ and observe, that $c(\psi_\infty, \mathcal{N}) = 2$. The map $\varphi_\infty \circ \psi_\infty$ is then a code on $\mathcal{N}$ with the cost $2\beta < 1$ but that is impossible by the coding theorem for combinatorial sources since $\dim_H(\mathcal{N}) = 1$ by another well known theorem (see theorem §133). We conclude that $\dim_H(\mathcal{C}) = 1/2$ and we are done. Extending the proof to cover any value of $\alpha$ is not difficult. It is done in two steps, firstly by extending the result to cover any rational value of $\alpha$ and secondly by extending the result to full generality.

§261 **Observation**. The theorem is illustrated in figure 1. The term "invisible set" originates (to the authors best knowledge) in [Ryabko et al., 1999] and the term points to the fact that any stationary ergodic measure of entropy $\alpha$ has its support inside the set $\mathcal{E}(\alpha)$. Thus any subset from the light gray area on the figure is "invisible" to the stationary ergodic measures. This is interesting in two aspects. Firstly, we note that the invisible set is made from sequences that are actually compressible. The probabilistic term "almost surely" or "with probability one" makes it easy to forget the fact the probabilistic framework can "see" only a small part of $A^\infty$. Secondly, the existence of compressible sequences outside the view of the stationary ergodic model is made even more interesting by the fact that we have at our disposal a way of determining the size (or complexity) of sets of these sequences, namely the Hausdorff dimension, and that this dimension agrees with the entropy where both are meaningful.

§262 **Remark**. It is probably best to explain by an example why we feel the Hausdorff dimension is an important tool when working with coding in $A^\infty$: The Context-Tree Weighting (CTW) method is a weakly universal code (see [Willems et al., 1995] for details). This code is quite different from the Lempel-Ziv code so it is interesting to compare their performance on individual sequences, as they will of course perform equally well in the stationary ergodic model. This was done in [Åberg et al., 1998] where an infinite sequence $x$ was constructed such that the finite state complexity of $x$ equals 1 (and hence it is incompressible by SLZ) yet such that it is compressible by the CTW code. However, such a sequence does not guarantee that there are no sequences for which the opposite hold (compressible by SLZ but not by CTW). And if there are sequences with this opposite property, we may ask of which type are there most? These questions were addressed in [Ryabko et al., 1999]. Let $CTW(\alpha)$ be those infinite binary sequences that can be compressed by the CTW code at a rate no more than $\alpha$ and let $SLZ(\alpha)$ be the corresponding set for the SLZ encoder. It was then shown that $\dim_H(CTW(\alpha)\backslash SLZ(\alpha)) = 0$ and $\dim_H(SLZ(\alpha)\backslash CTW(\alpha)) = \alpha$. In the following section we will pursue this idea of comparing the performance of universal codes using the Hausdorff dimension rather than just isolated cases of individual sequences with specific compressibility properties.

## The MPM code

§263 **Remark**. The MPM code is a weakly universal code which is fundamentally different from the Lempel-Ziv code. In this section we briefly recapture how the MPM code works, by means of providing an example of encoding a specific sequence. For a more detailed and concise definition of the code, we refer to [Kieffer et al., 2000].

§264 **Example** (MPM ENCODING). Consider the sequence

$$0000\ 1001\ 1010\ 1111\ 1001\ 0000\ 1111\ 1010$$

of 32 binary symbols, where spaces are inserted for every 4 symbol for readability. The MPM encoding begins by determining the parameter $I$ (chosen such that $2^I \leq |x|$ where $x$ is the sequence to be encoded). According to [Kieffer et al., 2000] the algorithm achieves good compression for values of $C \log\log|x| \leq I \leq \log|x|$ with $C$ a constant. In the following we use $I = \lfloor \log|x| \rfloor$, which in our case amounts to $I = 5$. The MPM works in $I + 1$ steps, corresponding to $i = 0, \ldots, I$. At each step, $i$, the sequence to be encoded is split into non-overlapping subsequences of length $2^{I-i}$.

So in our example, in the first step $i = 0$, we split the input into sequences of length $2^5 = 32$. This is the whole sequence, so the result is simply the sequence itself:

$$\text{Step } i = 0: \quad 00001001101011111001000011111010$$

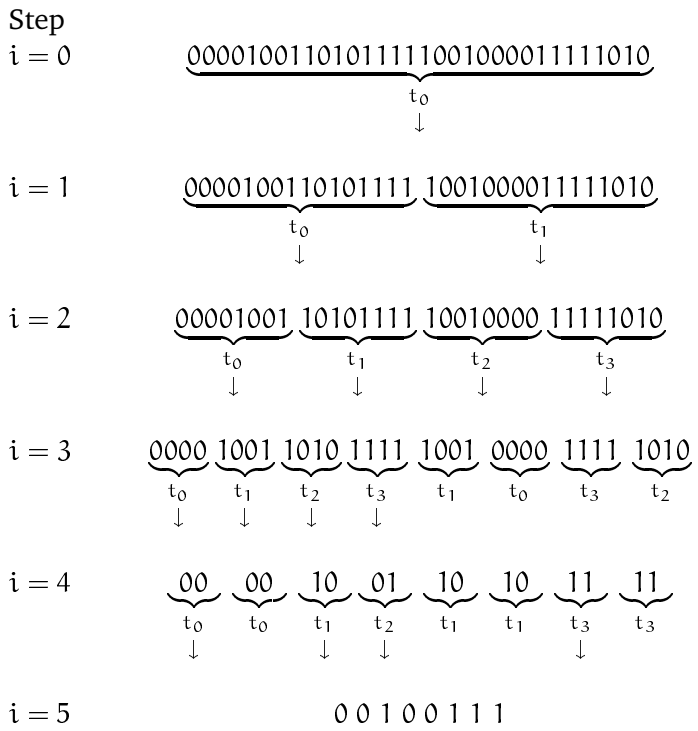
Furthermore, at each step after the splitting has taken place, distinct blocks are identified and named $t_j$, with $j = 0, 1, \ldots$ (the names are known as "tokens"):

$$\begin{array}{rc} & 00001001101011111001000011111010 \\ \text{Token:} & t_0 \end{array}$$

In the next step things get a bit more interesting. The splitting creates two subsequences each of length $2^{5-1} = 16$, and they are distinct, hence named $t_0$ and $t_1$ respectively:

$$\begin{array}{lcc} & 0000100110101111 \quad , & 1001000011111010 \\ \text{Tokens:} & t_0 & t_1 \end{array}$$

If, as a result of splitting the input sequence, the same subsequence occurs more than once yielding a token sequence of, say $t_0, t_1, t_0, t_0$, then only the distinct subsequences (corresponding to $t_0$ and $t_1$) are passed on to the next step. This is illustrated in the complete example below, where the downward arrows show which block are propagated to the next step.




Finally, the codeword representing the message is constructed by encoding the following information: The length (32) of the message, the token sequences created in the above scheme ($t_0$ and $t_0t_1$ and $t_0t_1t_2t_3$ and $t_0t_1t_2t_3t_1t_0t_3t_2$ and $t_0t_0t_1t_2t_1t_1t_3t_3$) as well as the last sequence of symbols from $A$ (00100111).

§265 **Example** (MPM DECODER). Continuing the example, we explain how the decoder works: The sequence of symbols from $A$ is split into pairs (00 10 01 11). The sequence of tokens $t_0t_0t_1t_2t_1t_1t_3t_3$ may now be decoded by the procedure of replacing any occurrence of $t_0$ by 00, any occurrence of $t_1$ by 10 and so on. This yields 0000100110101111. The decoding procedure now continues in steps corresponding to the encoding. Thus in the next step we split into substrings of length 4: 0000 1001 1010 1111. These substrings are then used when decoding the next sequence of tokens, $t_0t_1t_2t_3t_1t_0t_3t_2$, and so forth.

§266 **Observation**. The MPM encoder is an example of a so called grammar based codes. As the name indicates these codes transform the message to be encoded into a grammar which is then compressed. In the example above we may uniquely identify each token by extending the

notation so that we write $t_{i,j}$ for the token $t_j$ generated at step $i$. We then find that the MPM encoding implies the transformation of the message into the following grammar:

$$\begin{array}{llllll}
x & \to t_{0,0} & t_{2,2} & \to t_{3,1}t_{3,0} & & \\
t_{0,0} & \to t_{1,0}t_{1,1} & t_{2,3} & \to t_{3,3}t_{3,2} & t_{4,0} & \to 00 \\
t_{1,0} & \to t_{2,0}t_{2,1} & t_{3,0} & \to t_{4,0}t_{4,0} & t_{4,1} & \to 10 \\
t_{1,1} & \to t_{2,2}t_{2,3} & t_{3,1} & \to t_{4,1}t_{4,2} & t_{4,2} & \to 01 \\
t_{2,0} & \to t_{3,0}t_{3,1} & t_{3,2} & \to t_{4,1}t_{4,1} & t_{4,3} & \to 11 \\
t_{2,1} & \to t_{3,2}t_{3,3} & t_{3,3} & \to t_{4,3}t_{4,3} & &
\end{array}$$

The decoding is now equivalent to the evaluation of the symbol $x$ in the above grammar, that is, $x \to t_{0,0} \to t_{1,0}t_{1,1} \to t_{2,0}t_{2,1}t_{2,2}t_{2,3} \to \cdots$, all the way through to the original message.

§267 **Observation**. We mention briefly some aspects of the codeword length of the MPM code: For long messages, the contribution due to the encoding of the length of the message is negligible. The same goes for the contribution due to the encoding of the final sequence of symbols from $A$. Each of the token sequences are encoded using an arithmetic encoder with the following adaptive model:

$$p(t_i|u) = \begin{cases} \frac{f(t_i|u)}{|u|+m}, & 0 \le i < m \\ \frac{m}{|u|+m}, & i = m \end{cases} \tag{3}$$

where $u$ denotes the sequence of tokens and $f(t_i|u)$ is the number of times $t_i$ occurs in $u$. The value $m$ is defined as $m = \min\{i \mid f(t_i|u) = 0\}$ that is, the token $t_m$ does not occur in $u$. The special case for $m$ is to enable the gradual extension of the possible tokens. It may then be checked that there is an arithmetic encoder $\varphi$ such that

$$|\varphi(u)| = \left\lceil \sum_{j=1}^{|u|-1} -\log p(u_{j+1}|u_1^j) \right\rceil + 1 \tag{4}$$

§268 **Observation**. As mentioned above the MPM code is a weakly universal code, that is, for any stationary ergodic measure $\mu$ we have $\lim_{n\to\infty}(1/n)E_\mu(|\varphi_{MPM}(x_1^n)|) = H(\mu)$. This is proved in [Kieffer et al., 2000]. It is interesting, however, that the proof is based on a more general theorem stating that any so called *asymptotically compact grammar based* code is a weak universal code. The Lempel-Ziv code is an asymptotically compact grammar based code (a fact which follows directly from the compactness lemma, see lemma §178), thus we have yet another proof of the weak universality of the Lempel-Ziv code. For details on grammar based codes including the definition of asymptotically compact grammar codes, see [Kieffer and Yang, 2000].

## Comparing Lempel-Ziv and MPM

§269 **Remark**. For our purpose the most important fact about the MPM code is that it is weakly universal and hence it is of interest to consider sequences where the asymptotic performance of MPM differs from other weakly universal codes, say the Lempel-Ziv code. As was indicated by the results by Shields and others, it seems to be the case that LZ77 (or ULZ) is the most

powerful of the Lempel-Ziv variations, so we are going to compare ULZ and MPM. In this section we outline results in this regard.

§270 **Theorem** ([RYABKO AND SUZUKI, 2001]). *For any* $\alpha \in ]0,1[$ *there is a set* $M(\alpha) \subset \{0,1\}^\infty$ *of infinite sequences such that*

$$\liminf_{n\to\infty} \frac{|\varphi_{LZ77}(x_1^n)|}{n} = \alpha$$

*and*

$$\liminf_{n\to\infty} \frac{|\varphi_{MPM}(x_1^n)|}{n} = 1$$

*and the size of the set* $M(\alpha)$ *is such that* $\dim_H(M(\alpha)) > \alpha$.

§271 **Proof**: See [Ryabko and Suzuki, 2001] for details. However we will describe the method used in constructing sequences that are impossible to compress using MPM but can be compressed by the ULZ code. The construction is based on the fact that for any sequence $x = x_1^n \in A^n$ the sequence $xS^ix$ where $S$ is the cyclical shift is compressible by ULZ. To see this, observe that the second half of $xS^ix$ can contribute with no more than 2 phrases. Thus $|\varphi_{LZ77}(xS^ix)| \leq |x| + O(\log|x|) = n + O(\log n)$, while of course $|xS^ix| = 2n$ so that the compression attained is as close as we want to $1/2$. Having made this discovery we search for sequences of the form $xS^ix$ that are incompressible for the MPM code. The proof that such sequences exists (and that there are quite many of them) is the heart of this result. Here we only indicate the construction by an example. For any value of $d \in \mathbb{N}$ a sequence is made as a concatenation of words of length $2^d$. We will treat the case $d = 1$ here, that is, we work with words of length 2. The sequence that is made has the property all the $2^2 = 4$ possible words of length 2 occurs exactly once. For example,

$$x = 00\ 01\ 10\ 11$$

is such a sequence. We then shift the sequence cyclically one word length to the left and concatenate with the original sequence to get

$$xS^2x = 00\ 01\ 10\ 11\ 01\ 10\ 11\ 00.$$

Since this sequence has the form $xS^2x$ it can be compressed by the LZ77 algorithm. However, for the MPM code we observe the following behavior,

$$\begin{aligned}
i = 0 &\qquad \underbrace{0001101101101100}_{t_0}\\
i = 1 &\qquad \underbrace{00011011}_{t_0}\underbrace{01101100}_{t_1}\\
i = 2 &\qquad \underbrace{0001}_{t_0}\underbrace{1011}_{t_1}\underbrace{0110}_{t_2}\underbrace{1100}_{t_3}\\
i = 3 &\quad \underbrace{00}_{t_0}\ \underbrace{01}_{t_1}\ \underbrace{10}_{t_2}\ \underbrace{11}_{t_3}\ \underbrace{01}_{t_1}\ \underbrace{10}_{t_2}\ \underbrace{11}_{t_3}\ \underbrace{00}_{t_0}.
\end{aligned}$$

The MPM algorithm is unable to detect any recurring blocks (tokens) until at the step where the block-size is as small as the word size (2 in our case), and even there each of the tokens occurs with the same frequency which is bad for the MPM encoding (recalling the adaptive model described in formula (3) of observation §267).

The presentation given here has just covered the most basic steps involved in the proof. Apart from a somewhat more sophisticated construction of the sequences (mainly due to the need for

handling all cases of $\alpha \in ]0,1[$ and not just $\alpha = 1/2$ as we have done here), quite a bit of work must be allotted to the calculation of the number of sequences with the desired property. Also we have only considered finite sequences, while the result works with infinite sequences.

§272 **Observation**. It must be stressed (as is also done in [Ryabko and Suzuki, 2001]) that even though this theorem proves the existence of a large set of sequences for which ULZ performs better than MPM, the opposite might be true as well. That is, it is conceivable that there is a set $L \subset A^\infty$ of sequences that are incompressible by ULZ yet can be compressed to a certain rate, say $\alpha$, by the MPM code. More precisely $L = \{x \in A^\infty \mid \mathrm{ULZ}(x) = 1 \text{ and } \mathrm{MPM}(x) < \alpha\}$ with $\alpha < 1$. And this set might even be large, for example having $\dim_H(L) = \alpha$. It seems to be an open question if such a set exists.

§273 **Observation**. A number of attempts at the construction of sequences that are members $L$ have been made by Boris Ryabko and the author. It currently seems that there are not many sequences in $L$. Therefore it is conjectured that the set $L$ has Hausdorff dimension $0$.

§274 **Observation**. The comparison of weak universal codes using individual sequences and Hausdorff dimension is still not fully explored. While we know (by theorem §248) that there are sequences for which ULZ performs better than SLZ we do not know the Hausdorff dimension of the set of these sequences nor if there exists sequences with the opposite property. As mentioned, it is conjectured that there are only few sequences for which MPM provides compression but ULZ does not. The relation between MPM and SLZ in this respect has not yet been explored. It would be interesting to create a "map of efficiency" (a term coined by Ryabko) of not only the codes we have covered here, but also other variants of the Lempel-Ziv and grammar based codes. Such a map would describe the sets where one code outperforms another as well as the Hausdorff dimension of such sets. Finally, and perhaps most interestingly, would be to understand better the relation between the parsings involved in LZ, MPM (and other codes), and the Hausdorff dimension. Is there, for example, a Hausdorff-like dimension which corresponds to the ULZ code in the same way as the empirical entropy corresponds to the finite state compressibility?

## Notes and references

§275 **References**. In this chapter most references are given in the text. The main sources of inspiration, however, has been [Shields, 1999], [Ryabko et al., 1999] and [Hojo et al., 1998] (upon which [Hojo et al., 2001] is based). An important factor in the shaping of the ideas of this chapter has also been personal correspondence and discussions with Boris Ryabko.

*Use honest scales and honest weights, and honest ephah and an honest hin.*
LEVITICUS 19:36 (NIV)

# Afterword

*The best way to inspire fresh thoughts is to seal the envelope.*

ANONYMOUS

Looking back at the goal introduced in Chapter I gives a sense of distance, and indeed we have come quite a far way. If the reader has arrived at this point by regular means, that is, page by page, he or she will be the judge as to whether the road has been reasonable straight (except for perhaps a few detours along more rugged roads to catch a scenic view), and if the route has been chosen so that insight was gained.

However, to the author it has been a very instructive and worth-while process, answering old questions and asking new ones. And that is probably how things should be.

APPENDIX A

# Auxiliary results

§276 **Lemma** (LOG-SUM INEQUALITY). *Let* $x_1, \ldots, x_n$ *and* $y_1, \ldots, y_n$ *be positive numbers. Assume* $\sum_{i=1}^n x_i \geq \sum_{i=1}^n y_i$ *then*

$$\sum_{i=1}^{n} x_i \log x_i \geq \sum_{i=1}^{n} x_i \log y_i. \tag{1}$$

§277 **Proof**: By the inequality $\ln x \leq x - 1$ we find

$$\sum_{i=1}^{n} x_i(\ln y_i/x_i) \overset{(\dagger)}{\leq} \sum_{i=1}^{n} x_i(y_i/x_i - 1) = \sum_{i=1}^{n}(y_i - x_i) \leq 0, \tag{2}$$

from which the inequality follows. Assume $x_i = y_i$ for all values of $i$, then clearly we have equality in (§1). Assume conversely that we have equality in (§1). Then we find that the left hand side in (§2) must equal 0 and hence the inequality marked with $(\dagger)$ becomes an equality. Because $\ln x = x - 1$ only when $x = 1$ the results follows.

§278 **Lemma**. *Let* $m$ *and* $\alpha$ *be a positive integers. Then the following formula holds*

$$\sum_{i=1}^{m} i\alpha^i = \frac{m\alpha^{m+2} - (m+1)\alpha^{m+1} + \alpha}{(\alpha - 1)^2}$$

§279 **Proof**: Write $g(m) = \sum_{i=1}^m i\alpha^i$ and observe that $g(1) = \alpha$. We now see that

$$\begin{aligned} g(m) &= \alpha + \alpha \sum_{i=2}^{m} i\alpha^{i-1} \\ &= \alpha + \alpha \sum_{i=2}^{m} \alpha^{i-1} + \alpha g(m-1) \\ &= \alpha + \frac{\alpha^{m+1} - \alpha}{\alpha - 1} - \alpha + \alpha g(m-1) \end{aligned}$$

by the standard formula $\sum_1^m \alpha^i = (\alpha^{m+1} - \alpha)/(\alpha - 1)$. We may now write

$$\begin{aligned} g(m) &= \frac{\alpha^{m+1} - \alpha}{\alpha - 1} + \alpha\left(\frac{\alpha^m - \alpha}{\alpha - 1} + \alpha g(m-2)\right) \\ &= \frac{2\alpha^{m+1} - \alpha - \alpha^2}{\alpha - 1} + \alpha^2 g(m-2) \\ &\vdots \\ &= \frac{(m-1)\alpha^{m+1} - \sum_{i=1}^{m-1} \alpha^i}{\alpha - 1} + \alpha^{m-1}\alpha. \end{aligned} \tag{3}$$

Rearranging this yields the desired result.

§280 **Observation**. As a special case we note by (3) that for $\alpha = 2$ we have a particular simple formula,

$$\sum_{i=1}^{m} i2^i = (m-1)2^{m+1} + 2.$$

From the formula proven in the lemma we may derive an useful upper bound. Rewriting we get

$$\sum_{i=1}^{m} i\alpha^i = \left(\frac{m\alpha^{m+1}}{\alpha-1}\right)\left(\frac{\alpha - \frac{m+1}{m} + \frac{1}{m\alpha^m}}{\alpha-1}\right) = \left(\frac{m\alpha^{m+1}}{\alpha-1}\right)\left(\frac{\alpha - \frac{m\alpha^m+\alpha^m-1}{m\alpha^m}}{\alpha-1}\right)$$

from which follows, that for $\alpha > 1$ we have

$$\sum_{i=1}^{m} i\alpha^i \leq \frac{m\alpha^{m+1}}{\alpha-1}. \tag{4}$$

## Notes and references

§281 **Remark**. Lemma §278 is found in [Knuth, 1997] as exercise 1.2.3(16).

APPENDIX B

# Bibliography


Abbott, Edwin A. *Flatland : a romance of many dimensions*. Seeley & Co., Ltd., 1884. Dover edition, 1992.

Åberg, Jan, Paul Volf and Frans Willems. Compressing and incompressible sequence. In *Proc. ISIT 1998*, page 134, 1998.

Beeler, M., R.W. Gosper, R. Schroeppel et al. `HAKMEM`. Artificial Intelligence Memo 239, MIT, AI-Lab, 1972.

Bell, T.C., J.G. Cleary and I.H. Witten. *Text Compression*. Prentice-Hall, Englewoods Cliffs, NJ, 1990.

Berstel, J. and D. Perrin. *Theory of codes*. Academic Press, 1985.

Billingsley, P. *Ergodic theory and Information*. John Wiley & Sons, 1965.

Calude, C. *Information and randomness - an algoritmic perspective*. Springer, 1994.

Champernowne, D.G. The construction of decimals normal in the scale of ten. *Journal of the London Math. Soc.*, 8:254–260, 1933.

Cover, T. M. and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

Davisson, L. D. Universal noiseless coding. *IEEE Trans. Inform. Theory*, 17(6):783–795, 1973.

Davisson, L. D. Minimax noiseless universal coding for Markov sources. *IEEE Trans. Inform. Theory*, 29(2):211–215, 1983.

Elias, P. Universal codewords sets and representation of the integers. *IEEE Trans. Inform. Theory*, 21:194–203, 1975.

Escott, A. E. and S. Perkins. Binary Huffman equvivalent codes with a short synchronizing codeword. *IEEE Trans. Inform. Theory*, 44:346–351, 1998.

Falconer, K. *The geometry of fractal sets*. Cambridge University Press, 1985.

Falconer, K. *Fractal Geometry - Mathematical Foundations and Applications*. John Wiley & Sons, 1990.

Falconer, K. *Techniques in Fractal Geometry*. John Wiley & Sons, 1997.

Feder, M. and N. Merhav. Hierarchical universal coding. *IEEE Trans. Inform. Theory*, 42(5): 1354–1364, 1996.

Feder, M., N. Merhav and M. Gutman. Universal prediction of individual sequences. *IEEE Trans. Inform. Theory*, 38(4):1258–1270, 1992.

Ferguson, T. J. and J. H. Rabinowitz. Self-synchronizing Huffman codes. *IEEE Trans. Inform. Theory*, IT-30:687–693, 1984.

Gyorfi, L., I. Pali and E. C. van der Meulen. There is no universal source code for an infinite source alphabet. *IEEE Trans. Inform. Theory*, 40(1):267–271, 1994.

Hall, Jr., Marshall. *Combinatorial Theory*. Blaisdell, 1967.

Haussler, D. A general minimax result for relative entropy. *IEEE Trans. Inform. Theory*, 43(4): 1276–1280, 1997.

Hojo, K., B. Y. Ryabko and J. Suzuki. It is not enough to assume stationary ergodic sources for analyzing universal coding. In *Proc. of the ISIT '98*, Mexico City, 1998.

Hojo, Kouki, Boris Ya. Ryabko and Joe Suzuki. Performance of data compression in terms of hausdorff dimension. *IEICE Trans. Fundamentals*, 2001. To appear.

Huffmann, D.A. A method for the construction of minimum redundancy codes. *Proc. IRE*, 1952.

Jacobsen, M. *Videregående sandsynlighedsregning*. Institut for Matematisk Statistik, Københavns Universitet, 1995.

Kato, A. and H. Nagaoka. Huffman coding with an infinite alphabet. *IEEE Trans. Inform. Theory*, 42:977–984, 1996.

Kieffer, J. and E.-H. Yang. A simple technique for bouding the pointwise redundancy of the 1978 lempel-ziv algorithm. In *Proc. 1999 Data Comp. Conf. (Showbird, Utah)*, pages 434–441, 1999.

Kieffer, J. and E.-H. Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inform. Theory*, 46:737–754, 2000.

Kieffer, J., E.-H. Yang, Gregory Nelson and Pamela Cosman. Universal lossless compression via multilevel pattern matching. *IEEE Trans. Inform. Theory*, 46, 2000.

Kieffer, J.C. A unified approach to weak universal source coding. *IEEE Trans. Inform. Theory*, 24 (6):674–682, 1978.

Kieffer, J.C. and E. Yang. Sequential codes, lossless compression of individual sequences and Kolmogorov complexity. *IEEE Trans. Inform. Theory*, 42(1):29–40, 1996.

Knuth, D. E. Dynamic Huffman coding. *Journal of Algorithms*, 6:163–180, 1985.

Knuth, D. E. *The art of computer programming: Fundamental Algorithms*, volume 1. Addison-Wesley, third edition, 1997.

Langdon, G.G. A note on the Ziv-Lempel model for compressing individual sequences. *IEEE Trans. Inform. Theory*, 29(2):284–287, 1983.

Langdon, G.G. An introduction to arithmetic coding. *IBM J. Res. Develop.*, 28:135–149, 1984.

Lempel, A. and J. Ziv. On the complexity of finite sequences. *IEEE Trans. Inform. Theory*, 22: 75–81, 1976.

Li, M. and P.M.B. Vitanyi. *An introduction to Kolmogorov Complexity and its Applications*. North Holland, 1993.

Linder, T., V. Tarokh and K. Zeger. Existence of optimal prefix codes for infinite alphabet source alphabets. *IEEE Trans. Inform. Theory*, 43(6):2026–2028, 1997.

Louchard, G. and W. Szpankowski. On the average redundancy rate of the Lempel-Ziv code. *IEEE Trans. Inform. Theory*, 43(1):1–8, 1997.

Merhav, N. and M. Feder. A strong version of the redundancy-capacity theorem of universal coding. *IEEE Trans. Inform. Theory*, 41(3):714–722, 1995.

Merhav, N. and M. Feder. Universal prediction. *IEEE Trans. Inform. Theory*, 44(6):2124–2147, 1998.

Moffat, A. Linear time adaptive arithmetic coding. *IEEE Trans. Inform. Theory*, 36(2):401–405, 1990.

Neuhoff, D. L. and P. C. Shields. Simplistic universal coding. *IEEE Trans, Inform, Theory*, 44(2): 778–781, 1998.

Ornstein, D. and B. Weiss. The Shannon-McMillan-Breiman theorem for amenable groups. *Israel J. Math.*, 44:53–60, 1983.

Ornstein, Donald Samuel and Bejamin Weiss. Entropy and data compression schemes. *IEEE Trans. Inform. Theory*, 39(1):78–83, 1993.

Pasco, R.C. *Source coding algorithms for fast data compression*. PhD thesis, Stanford, 1979.

Pierce II, Larry A. and Paul C. Shields. Sequences incompressible by slz (lzw), yet fully compressible by ulz. In *Ahlswede 60th Birthday conference proceedings*, 1998.

Pisinger, D., J. Weismann and C. Tønsberg. Bridge problems associated with arithmetic coding. Article submitted to *IEEE Trans. Inform. Theory.* The reference number is 98-384. Obtainable upon request to weismann@diku.dk, 1998.

Plotnik, E., M.J. Weinberger and J. Ziv. Upper bounds on the probability of sequences emitted by finite-state sources and on the redundancy of the Lempel-Ziv algorithm. *IEEE Trans. Inform. Theory*, 38:66–72, 1992.

Rissanen, J. Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Develop.*, 20:198–203, 1976.

Rissanen, J. A universal data compression system. *IEEE Trans. Inform. Theory*, 29:656–664, 1983.

Rissanen, J. Universal coding, information, prediction and estimation. *IEEE Trans. Inform. Theory*, 30(4):629–636, 1984.

Rissanen, J. Complexity of strings in the class of Markov sources. *IEEE Trans. Inform. Theory*, 32 (4):526–532, 1986.

Rissanen, J. *Stochastic Complexity in Statistical Inquiry*. World Scientific, 1989.

Rissanen, J. and G.G. Langdon. Aithmetic coding. *IBM J. Res. Develop.*, 23:149–162, 1979.

Rissanen, J. and G.G. Langdon. Universal modeling and coding. *IEEE Trans. Inform. Theory*, 27 (1):12–23, 1981.

Ryabko, B. Y. Twice-universal coding. *Problems of Inform. Trans.*, 20(3):173–177, 1984.

Ryabko, B. Y. Noiseless coding of combinatorial sources, Hausdorff dimension, and Kolmogorov complexity. *Problems of Inform. Trans.*, 22(3):170–179, 1986.

Ryabko, B. Y. Prediction of random sequences and universal coding. *Problems of Inform. Trans.*, 24(2):87–96, 1988.

Ryabko, B. Y., J. Suzuki and F. Topsøe. Hausdorff dimension as a new dimension in source coding and predicting. In *Proc. 1999 IEEE Inform. Theory Workshop*, Kruger National Park, South Africa, 1999.

Ryabko, Boris and Joe Suzuki. Comparing the multilevel pattern matching code and the lempel-ziv codes. In *Proc. of the ISIT 2001*, Washington D.C., 2001.

Ryabko, B.Ya. and A.N. Fionov. An efficient method for adaptive arithmetic coding of sources with large alphabets. *Problemy Peredachi Informatsii*, 35(4):95–108, 1999. (in Russian).

Savari, S.A. Redundancy of the Lempel-Ziv incremental parsing rule. *IEEE Trans. Inform. Theory*, 43(1):9–21, 1997.

Savari, S.A. Redundancy of the Lempel-Ziv string mathcing code. *IEEE Trans. Inform. Theory*, 44(2):787–792, 1998.

Shannon, C.E. A mathematical theory of communication. Technical report, Bell System, 1948.

Shields, P. C. *The ergodic theory of discrete sample paths*. Amer. Math. Soc., 1996.

Shields, P.C. Universal redundancy rates do not exists. *IEEE Trans. Inform. Theory*, 39:520–524, 1993.

Shields, P.C. The interactions between ergodic theory and information theory. *IEEE Trans. Inform. Theory*, 44(6):2079–2093, 1998.

Shields, P.C. Performance of LZ algorithms on individual sequences. *IEEE Trans. Inform. Theory*, 45(4):1283–1288, 1999.

Welch, T.A. A technique for high-performance data compression. *IEEE Computer*, 17:8–19, 1984.

Willems, F.M.J., Y.M. Shtarkov and T.J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Trans. Inform. Theory*, 41(3):653–664, 1995.

Witten, I.H., R.M. Neal and J.G. Cleary. Arithmetic coding for data compression. *Comm. ACM*, 30:520–540, 1987.

Wyner, A.D. and J. Ziv. The sliding-window Lempel-Ziv is asymptotically optimal. *Proc. IEEE*, 82:872–877, 1994.

Yamamoto, Hirosuke. A new recursive universal code of the positive integers. *IEEE Trans. Inform. Theory*, 46(2):717–723, 2000.

Ziv, J. Coding theorems for individual sequences. *IEEE Trans. Inform. Theory*, 24(4):405–412, 1978.

Ziv, J. and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory*, 23(3):337–343, 1977.

Ziv, J. and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inform. Theory*, 24(5):530–536, 1978.

APPENDIX C

# Computer programs

*Beware of bugs in the above code; I have only proved it correct, not tried it.*
DONALD E. KNUTH

## A Perl-implementation of LZ78 for study purposes

In the listing below is a Perl-implementation of the LZ78 algorithm for parsing and encoding. The method of parsing is described in algorithm §163 and the encoding is described in observation §169. The program is not suitable for practical compression tasks; for such purposes it is far too simple and lacking both in terms of running time and compression capability. It may seem strange that different implementations of the same encoding method should not have the same compression capability. However, almost none of the compression programs in practical are direct implementations of well known algorithms. In most cases a number of clever programming techniques and heuristics are used to tweak the implementation to perform well on one of the standard data sets that are used when evaluating implementations of compression programs. From a mathematical point of view these issues are not that interesting. For our purpose a simple program, as the one presented below, which may serve as a tool for studying the LZ78 parsing and encoding, is of more use.

```
#!/usr/bin/perl
# $Id: lz78.pl,v 1.3 2001/04/01 22:03:28 pandr Exp $
# ------------------------------------------------------------------------ #
# Usage:
#   lz78.pl -p <message>   -   displays the lz78 parsing of <message>
#   lz78.pl -c <message>   -   displays the lz78 encoding of <message>
# ------------------------------------------------------------------------ #

use POSIX qw(ceil);

# ---------------------------------------- Fetch arguments ---------------- #

my $mode = shift @ARGV;
my $str  = shift @ARGV;

die "Usage: $0 [ -p | -c ] <message>\n" unless $str and $mode =~ m/^(-p|-c)$/;

# ---------------------------------------- Inititalize statistical vars --- #

my $messagelength = length($str);
```

```
my $codelength    = 0;
my $phrases       = 0;

# ---------------------------------------- Find out which alphabet -------- #

my %alphabet;
foreach (split //, $str) {
        $alphabet{$_} = $symnum++ unless defined($alphabet{$_});;
}
my @alphabet = keys %alphabet;
my $letters = @alphabet;

print "Alphabet: @alphabet ($letters letters)\n\n";

# ---------------------------------------- Main loop ---------------------- #

my %dict = ();
my $p = 1;

while (my $l = length($str)) {

        my $i;
        for ($i=1; $i<=$l; $i++) {
                last unless $dict{substr $str, 0, $i};
        }

        my $phrase = substr($str,0,$i);
        $dict{$phrase} = $p++ unless $i>$l;
        $str = substr($str,$i);

        $phrases++;
        if ($mode eq '-p') {
                print $phrase.",";
        }
        else {
                $idx = $dict{substr $phrase, 0, $i-1} || 0;
                $new = substr $phrase, $i-1, 1;
                $blocksize = ceil(log($letters*($p-1))/log(2));
                printf "%3s,%1s) - %0${blocksize}b\n","($idx", $new,
                        $alphabet{substr $phrase, $i-1, 1}
                        + $letters*$dict{substr $phrase, 0, $i-1};
                $codelength += $blocksize;
        }
}

# ---------------------------------------- Print statistics --------------- #

print  "\n" if $mode eq '-p';
print  "\n";
printf "Messagelength: $messagelength $letters-its (idealized: %4.2f bits)\n",
        $messagelength*(log($letters)/log(2));
print  "Phrases      : $phrases\n";
print  "Codelength   : $codelength\n" if $mode eq '-c';

# ------------------------------------------------------------------------- #
```

## Example output

The program works in one of two modes: parsing and coding. When invoked in the parsing mode, the program displays the parsing done by the LZ78 algorithm. In the following example, we try out the program in this mode, using as message the sequence `abbabbaaabdbabbb`:

```
$ ./lz78.pl -p abbabbaaabdbabbb
Alphabet: a b d (3 letters)

a,b,ba,bb,aa,ab,d,bab,bb,

Messagelength: 16 3-its (idealized: 25.36 bits)
Phrases      : 9
$
```

In the coding mode, the program shows the actual encoding into binary digits. Using the same message, we get the following result:

```
$ ./lz78.pl -c abbabbaaabdbabbb
Alphabet: a b d (3 letters)

 (0,a) - 00
 (0,b) - 001
 (2,a) - 0110
 (2,b) - 0111
 (1,a) - 0011
 (1,b) - 00100
 (0,d) - 00010
 (3,b) - 01010
 (4, ) - 01100

Messagelength: 16 3-its (idealized: 25.36 bits)
Phrases      : 9
Codelength   : 37
$
```

Note that even though the length of the message is 16 symbols, when comparing with the code length, which is 37 bits, we need to normalize with the factor $\log 3 \simeq 1.58$ so that the message length expressed in bits is about 25. Still, for short messages like in this example, we see that the LZ78 encoding actually does more harm than good in terms of compression.

*Beware of bugs in the above code; I have only tried it, not proved it correct.*

THE AUTHOR

APPENDIX D

# Definition of dimensions

*"Fool," said he, "Space is Length. Interrupt me again, and I have done."*
THE KING OF LINELAND. [ABBOTT, 1884]

§282 **Remark**. This appendix presents some auxiliary results about dimensions including the classical definition of the Hausdorff measure and dimension. The goal is to show the relationship with the Hausdorff dimension as defined on $A^\infty$ in Chapter II (see §127).

§283 **Definition**. Let $S$ be a subset of $[0,1]$, let $\delta > 0$ and let $\alpha \geq 0$. A $\delta$-*cover* of $S$ is a countable collection $\{C_i\}_{i\in\mathbb{N}}$ of subsets of $[0,1]$ each of which has $\operatorname{diam}(C_i) \leq \delta$ and whose union covers $S$, that is, $\cup_{i\in\mathbb{N}} C_i \supseteq S$. We define

$$\mathcal{H}_\alpha(S,\delta) \stackrel{\text{def}}{=\!=} \inf \sum_{i\in\mathbb{N}} \operatorname{diam}(C_i)^\alpha$$

where the infimum is over all $\delta$-covers of $S$. We then define

$$\mathcal{H}_\alpha(S) \stackrel{\text{def}}{=\!=} \lim_{\delta\to\infty} \mathcal{H}_\alpha(S,\delta).$$

§284 **Observation**. Let us check that the definition is well made: It is clear, that for any $S$ and any $\delta > 0$ there exist at least one $\delta$-cover, namely made from the sets $C_i = [\delta(i-1), \delta i]$ for $0 \leq i \leq 1/\delta$ plus the final one $C_j = [\delta(j-1), 1]$ for a suitable $j$ (we may set $C_i = \emptyset$ for $i > j$ for completeness.) Furthermore, $\mathcal{H}_\alpha(S,\delta)$ is clearly non-decreasing for $\delta \to 0$ because the infimum is taken over a decreasing number of possible covers as $\delta \to 0$. Thus $\mathcal{H}_\alpha(S)$ is well defined as a value in $[0,\infty]$.

§285 **Observation**. The set map $\mathcal{H}_\alpha(\cdot)$ is in fact Borel measure, and is known as the $\alpha$-*dimensional Hausdorff measure*. A bit of work (see e.g. [Falconer, 1985]) is needed in order to prove that $\mathcal{H}_\alpha$ is indeed a measure and since we will not pursue this aspect of $\mathcal{H}_\alpha$, we only check that it is an *outer measure*. Firstly, we have $\mathcal{H}_\alpha(\emptyset) = 0$ since the empty cover is a valid cover of $\emptyset$. Secondly, if $S \subseteq S'$ we note that any $\delta$-cover of $S'$ is also a $\delta$-cover of $S$, hence $\mathcal{H}_\alpha(S,\delta) \leq \mathcal{H}_\alpha(S',\delta)$ an thus $\mathcal{H}_\alpha(S) \leq \mathcal{H}_\alpha(S')$. Finally, we must check that if $\{S_n\}_{n\in\mathbb{N}}$ is a countable collection of subsets of $[0,1]$, then $\mathcal{H}_\alpha(\cup_n S_n) \leq \sum_n \mathcal{H}_\alpha(S_n)$. We may assume $\mathcal{H}_\alpha(S_n) < \infty$ for all $n$ since otherwise the result is trivial. To check the general case, let $S = \cup_{n=1}^\infty S_n$. Pick a $\delta > 0$ and an $\epsilon > 0$. To any $n \in \mathbb{N}$ there is a $\delta$-cover $\{C_{n,i}\}_{i\in\mathbb{N}}$ of $S_n$ so that $\sum_i \operatorname{diam}(C_{n,i})^\alpha \leq \mathcal{H}_\alpha(S_n,\delta) + \epsilon/2^n$. It is clear, that $\{C_{n,i}\}_{n\in\mathbb{N},i\in\mathbb{N}}$ is a $\delta$-cover of $\cup_{n\in\mathbb{N}} S_n$. This implies, that

$$\mathcal{H}_\alpha(\cup_{n\in\mathbb{N}} S_n,\delta) \leq \sum_{n\in\mathbb{N},i\in\mathbb{N}} \operatorname{diam}(C_{n,i})^\alpha \leq \sum_{n\in\mathbb{N}} \mathcal{H}_\alpha(S_n,\delta) + \sum_{n\in\mathbb{N}} \epsilon/2^n \leq \sum_{n\in\mathbb{N}} \mathcal{H}_\alpha(S_n,\delta) + \epsilon.$$

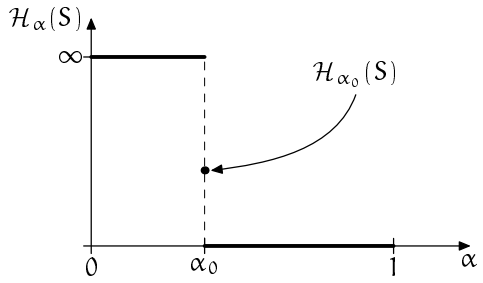



Figure 1: The characteristic behavior of a Hausdorff measure as described in observation §286.

As $\epsilon$ was chosen arbitrarily, we conclude that $\mathcal{H}_\alpha(\cup_{n\in\mathbb{N}} S_n, \delta) \leq \sum_{n\in\mathbb{N}} \mathcal{H}_\alpha(S_n, \delta)$. Taking the limit $\delta \to 0$ we find $\mathcal{H}_\alpha(\cup_{n\in\mathbb{N}} S_n) \leq \sum_{n\in\mathbb{N}} \mathcal{H}_\alpha(S_n)$.

§286 **Observation**. The most important feature, from our point of view, of the Hausdorff measure is its behavior when we fix the set that is measured, and let the dimension $\alpha$ vary. We find that $\mathcal{H}_\alpha(S)$ exhibits the following curious "jumping spot". Assume $\mathcal{H}_\alpha(S) > 0$ for some value $\alpha$. We then have $\mathcal{H}_\alpha(S,\delta) \leq \delta^{\alpha-\beta}\mathcal{H}_\beta(S,\delta)$ for $\beta < \alpha$ (this follows from $\operatorname{diam}(C_i)^\alpha = \operatorname{diam}(C_i)^{\alpha-\beta+\beta} \leq \delta^{\alpha-\beta}\operatorname{diam}(C_i)^\beta$.) When $\delta$ (and hence $\delta^{\alpha-\beta}$) becomes vanishingly small $\mathcal{H}_\beta(S,\delta)$ must grow correspondingly and we conclude that $\mathcal{H}_\beta(S) = \infty$. Similarly, if $\mathcal{H}_\alpha(S) < \infty$ we find $\mathcal{H}_\beta(S) = 0$ for any $\beta > \alpha$. Figure 1 illustrates this. We note, that $\mathcal{H}_0(S) > 0$ for any non-empty set $S$. Also, $\mathcal{H}_1(S) \leq 1$ for any set $S$. In fact, we have $\mathcal{H}_1(S) = \mathcal{L}(S)$ where $\mathcal{L}$ denotes the Lebesgue measure.

§287 **Definition**. The *Hausdorff dimension* of a set $S \subseteq [0,1]$ is defined as

$$\dim_H(S) = \inf\{\alpha \mid \mathcal{H}_\alpha(S) = 0\} = \sup\{\alpha \mid \mathcal{H}_\alpha(S) = \infty\}.$$

§288 **Lemma**. *Let $S \subseteq S' \subseteq [0,1]$, then $\dim_H(S) \leq \dim_H(S')$. Let $\{S_i\}_{i\in\mathbb{N}}$ be a countable collection of subsets of $[0,1]$, then $\dim_H(\cup_{i\in\mathbb{N}} S_i) = \sup_{i\in\mathbb{N}} \dim_H(S_i)$.*

§289 **Proof**: For the first proposition, note that it follows directly from the fact that $\mathcal{H}_\alpha(\cdot)$ is an outer measure (see observation §285.) For the second proposition let us write $S = \cup_{i\in\mathbb{N}} S_i$ and, by the same argument, we have $\dim_H(S) \geq \sup_{i\in\mathbb{N}} \dim_H(S_i)$, since clearly $S_i \subseteq S$. Assume then, that $\alpha > \sup_{i\in\mathbb{N}} \dim_H(S_i)$. This implies $\mathcal{H}_\alpha(S_i) = 0$ for all $i \in \mathbb{N}$ since $\alpha > \dim_H(S_i)$. By the sub-additivity of $\mathcal{H}_\alpha$, it follows that

$$\mathcal{H}_\alpha(S) \leq \sum_{i\in\mathbb{N}} \mathcal{H}_\alpha(S_i) = 0$$

and we have $\dim_H(S) \leq \alpha$ and we conclude $\dim_H(S) \leq \sup_{i\in\mathbb{N}} \dim_H(S_i)$.

§290 **Definition**. Let $r \in \mathbb{N}$ and $r \geq 2$. A *r-ary interval (of order n)* is an interval in $[0,1[$ of the form $[\frac{i}{r^n}, \frac{i+1}{r^n}[$ for some values of $n, i \in \mathbb{N}_0$. Let $S \subseteq [0,1[$ and define

$$\mathcal{N}_\alpha(S,\delta) \stackrel{\text{def}}{=\!=} \inf \sum_{i\in\mathbb{N}} \operatorname{diam}(I_i)^\alpha \qquad \text{and} \qquad \mathcal{N}_\alpha(S) \stackrel{\text{def}}{=\!=} \lim_{\delta\to 0} \mathcal{N}_\alpha(S,\delta),$$

where the infimum is taken over all $\delta$-covers made entirely from $r$-ary intervals. The set map $\mathcal{N}_\alpha(\cdot)$ is known as the *$\alpha$-dimensional net measure*. Finally, set

$$\dim_{\mathcal{N}}(S) = \inf\{\alpha \mid \mathcal{N}_\alpha(S) = 0\} = \sup\{\alpha \mid \mathcal{N}_\alpha(S) = \infty\}$$

§291 **Observation**. It is clear, that in the definition of net measures the fact that we only consider subsets of $[0,1[$ and not $[0,1]$ is non-essential. We may assume $\mathcal{N}_\alpha$ is defined on the whole $[0,1]$ because a single point $\{1\}$ makes no difference to the value of the measure. Even though the net measure may not agree with the Hausdorff measure, it does give rise to the same dimension, as we shall see next.

§292 **Lemma** (NET DIMENSION LEMMA). *For any set $S \subseteq [0,1]$ we have* $\dim_H(S) = \dim_{\mathcal{N}}(S)$.

§293 **Proof**: Let $C \subseteq [0,1]$ be a set which we may think of as part of a cover of $S$. We can assume $C$ to be an interval , since if is not, we replace it by the interval $C' = [\inf C, \sup C]$ and so forth for all sets. In terms of the definition of the Hausdorff measure, only the diameter is needed, so when $\operatorname{diam}(C) = \operatorname{diam}(C')$ we get the same measure[1]. Let $n$ be the smallest integer so that $[i/r^n, (i+1)/r^n[ \subseteq C$. We want to show that $C$ can be covered by $2r$ intervals each of length $1/r^n \leq \operatorname{diam}(C)$. Now we demonstrate that the interval $C$ can not touch more than $2r$ $r$-ary intervals of order $n$. Assume that $C$ did indeed touch $2r+1$ intervals, then $2r-1$ of those would be completely contained in $C$. Consider all the $r$-ary intervals of order $n$, starting with $[0/r^n, 1/r^n[$, then $[1/r^n, 2/r^n[$, and so on. The first $r$ of these intervals may be coalesced into a single $r$-ary interval of order $n-1$, likewise the next $r$ intervals, and so forth. Thus, when $C$ contains $2r-1$ $r$-ary intervals of order $n$ there must be among them at least one sequence of $r$ intervals which can be collapsed into a single interval of order $n-1$, still contained in $C$. Yet we assumed $n$ to be the smallest integer for which this situation could occur. We conclude that each $C$ can be covered by at most $2r$ $r$-ary intervals, each of length $1/r^n \leq \operatorname{diam}(C)$. This can be used in showing that

$$\inf \sum_{i \in \mathbb{N}} 2r \operatorname{diam}(C_i)^\alpha \geq \mathcal{N}_\alpha(S, \delta)$$

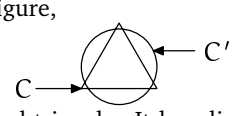
where the infimum is over all $\delta$-coverings of $S$. The inequality follows from the fact just shown, that to any $\delta$-cover $\{C_i\}_{i \in \mathbb{N}}$ of $S$ we may replace each set $C_i$ by at most $2r$ $r$-ary intervals of length at most $\operatorname{diam}(C)$. By definition $\mathcal{N}_\alpha$ is the infimum of all covers of $r$-ary intervals, and the result follows. Using the trivial inequality $\mathcal{H}_\alpha(S,\delta) \leq \mathcal{N}_\alpha(S,\delta)$ and taking the limit $\delta \to 0$ we arrive at

$$\mathcal{H}_\alpha(S) \leq \mathcal{N}_\alpha(S) \leq 2r\mathcal{H}_\alpha(S).$$

From this follows that $\dim_{\mathcal{N}}(S) = \dim_H(S)$.

§294 **Observation**. One may construct all $r$-ary intervals by the following procedure: Subdivide $[0,1[$ into $r$ intervals of equal length. This yields $[0,1/r[$, $[1/r, 2/r[, \ldots,$ $[(r-1)/r, 1[$, which is all the

[1]This does not generalize to higher dimensions. Instead, we need to convert every set into a sphere of diameter $2\operatorname{diam}(C)$. The problem is illustrated by this figure,



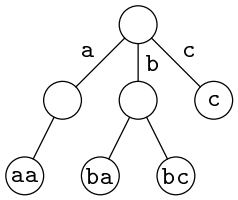

where the role of $C$ is played by the equilateral triangle. It has $\operatorname{diam}(C) = 1$ but cannot be replaced by a disc of diameter 1.

$r$-ary intervals of order 1. Further subdividing each of the 1-order intervals into $r$ equally sized subintervals, yields all 2-order intervals, and so forth. By this construction is seen that whenever any two $r$-ary intervals are not disjoint, then one is contained in the other. In the definition of $\mathcal{N}_\alpha(S,\delta)$ we can limit the infimum to be over only those covers which are *disjoint* collections of $r$-ary intervals of length less than $\delta$.

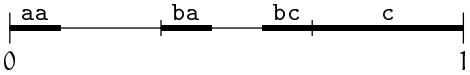
§295 **Observation**. It is well known, that a prefix free set $C \subset A^*$ corresponds to a $|A|$-ary tree which again corresponds to a certain collection of non-overlapping $|A|$-ary subintervals of $[0,1]$. We only indicate this correspondence by an example: Consider the alphabet $A = \{\mathtt{a},\mathtt{b},\mathtt{c}\}$ and the prefix free set

$$C = \{\mathtt{aa},\mathtt{ba},\mathtt{bc},\mathtt{c}\}. \tag{1}$$

Any set of finite sequences over $A$ may be represented as certain nodes in a $|A|$-ary tree and in the case of $C$ we would have




where the nodes representing $C$ are marked with the word they represent. The prefix property of $C$ corresponds to the fact that all elements of $C$ are found in leaf nodes. Another way of thinking of prefix sets (and thus also the leafs of trees) is as subsets of $[0,1]$. In particular subsets made from non-overlapping $|A|$-ary intervals. Continuing our example from before, we could represent $C$ as




where the portion of $[0,1]$ marked with thick line is the set in question. The $|A|$-ary intervals that make up the set is labeled according to which element of $C$ they represent. Note the similarities with the arithmetic coding scheme (see observation §149).
Finally, if we identify each of the symbols, $\mathtt{a},\mathtt{b},\mathtt{c}$ with the numbers $0,1,2$ respectively, we may write any number in the interval $[0,1]$ as $0.x_1x_2x_3\cdots$ where $x_i \in A$ using base 3 expansion. We find then, that any number $\alpha$ in the interval corresponding to, say $\mathtt{ba}$, does in fact have the form $\alpha = 0.\mathtt{ba}\ldots$. In general, the interval representing a sequence $x_1^n \in A^n$ is precisely the interval containing all numbers whose base 3 expansion has $x_1^n$ as prefix (ignoring the leading "0.", of course). Furthermore, the interval has length $r^{-n}$ (see observation §294 if this is not clear).

§296 **Lemma**. *Let $A$ be a finite alphabet with $|A| > 2$. There is then a surjective map $\tau : A^\infty \to [0,1]$ such that*

$$\dim_H(\tau(S)) = \dim'_H(S) \tag{2}$$

*for any $S \subseteq A^\infty$, where $\dim'_H$ is the Hausdorff dimension as defined for subsets of $A^\infty$ in Chapter II (see definition §127).*

§297 **Proof**: We may, without loss of generality assume $A = \{0,\ldots,r-1\}$ for some number $r$. Let $\tau$ be defined as the map which sends an infinite sequence of symbols from $A$, that is, an infinite sequence of numbers between 0 and $r-1$, into the $r$-ary expansion. More precisely $\tau(\mathbf{x}) \stackrel{\text{def}}{=\!=} 0.x_1x_2\cdots$, where the right hand side is to be interpreted as a number expressed in base

$r$. It is clear that $\tau$ is surjective. Let $S \subseteq A^\infty$. Recall from the definition of the net measure, that we considered $\delta$-covers made from $r$-ary intervals. Let $\mathcal{I}$ be one such cover of $\tau(S)$. For each $I \in \mathcal{I}$ we have $I = [j/r^n, (j+1)/r^n[$ and $|I| \leq \delta$ or, equivalently, $r^{-n} \leq \delta$ leading to $n \geq n_\delta = \lceil -\log n / \log \delta \rceil$. We now state the following equality:

$$\inf_{\mathcal{I}} \sum_{I \in \mathcal{I}} \operatorname{diam}(I)^\alpha = \inf_{C} \sum_{c \in C} r^{-|c|\alpha}, \tag{3}$$

where the infimum on the left hand side is taken over all $\delta$-covers of $\tau(S)$ made from $r$-ary intervals; and the infimum on the right hand side is taken over all $n_\delta$ covers of $S$ (see §127 for the definition of covers of subsets of $A^\infty$.) As we saw in observation §295 all numbers in an interval $I$ of the form $[j/r^n, (j+1)/r^n[$ shares the first $n$ digits when expressed as $0.x_1x_2\cdots x_n\cdots$ in base $r$. Thus to any interval $I \in \mathcal{I}$ we may select a word $c_I \in A^*$ of length $|c_I| = n$. Furthermore, the set $\{c_I\}_{I\in\mathcal{I}}$ is a $n_\delta$ cover of $S$, for assume $x \in S$, then $0.x \in \tau(S)$ and hence $0.x$ must be contained in some interval $I$ of $\mathcal{I}$ yielding the existence of $c_I$ which covers $x$. We note, that

$$\operatorname{diam}(I) = r^{-|c_I|}. \tag{4}$$

Assume conversely, that $C$ is a $n_\delta$-cover of $S$. Then there is to each $c \in C$ an interval $I_c = [j/r^{|c|}, (j+1)/r^{|c|}[$ containing all numbers from the interval $[0,1]$ having the form $0.c\cdots$ in base $r$. The intervals $\{I_c\}_{c\in C}$ is a $\delta$-cover of $\tau(S)$ since firstly, $|c| \geq n_\delta$ so that $r^{-|c|} \leq \delta$, and secondly, each element of $\tau(S)$ is of the form $0.x$ with $x \in S$ there is a $c \in C$ which is a prefix of $x$ so that the interval $I_c$ contains $0.x$. We note, that

$$r^{-|c|} = \operatorname{diam}(I_c). \tag{5}$$

We have shown that it is possible to create one type of cover (e.g. of a subset of $A^\infty$) from the other type of cover (e.g. of a subset of $[0,1]$) and by this construction and (4) and (5) we conclude that (3) holds. The proof of the lemma is now right before us: Taking the limit $\delta \to 0$ on (3) leads to the net measure on the left hand side, and to the number $l_\alpha(S)$ on the right hand side (see definition §127) because $n$ is forced towards infinity when $\delta$ vanishes. By the net dimension lemma (lemma §292) the desired result follows.

§298 **Summary**. It is in the light of the results of this appendix, that the definition of Hausdorff dimension made in Chapter II (see §127) should be seen. The concept of $n$-*cover* of a subset of $A^\infty$, corresponds to a $|A|^{-n}$-cover of non-overlapping $|A|$-ary intervals in the traditional definition of Hausdorff dimension, a fact used in the preceding lemma to show equivalence between the Hausdorff dimension of Chapter II and the classical one. It is reasonable at this point to ask what is the use of a result like lemma §296 when the involved map, $\tau : A^\infty \to [0,1]$ is not bijective, but only surjective. The answer is that the result is of great use, since the map is *almost* injective, in the following sense. One can show, that the set $\{x \in A^\infty \mid \exists y \in A^\infty : y \neq x \text{ and } \tau(x) = \tau(y)\}$ is countable and thus by lemma §288 should have Hausdorff dimension zero by any sound definition. We conclude, that when working with Hausdorff dimension we may use either of $\dim_H(\tau(S))$ (the classical definition) or $\dim_H(S)$ (the definition given in Chapter II).

## Notes and references

§299 **References**. The material presented in this appendix is quite standard, although perhaps seen

from a slightly different angle here. An excellent presentation can be found in [Falconer, 1990] and the information theoretical connection was first made (explicitly, at least) in [Billingsley, 1965].

§300 **The identity of** $1 = 0.999\cdots$. The non-injectiveness of the map $\tau : A^\infty \to [0,1]$ as described above is quite a source of irritation, not less so because it is intuitively almost clear that it is nothing but a technical nuisance. Defining the Hausdorff dimension directly on subsets of $A^\infty$ solves this once and for all. However, we then miss out on a neat trick that is employed in the original proof (see [Ryabko, 1986]) of theorem §131 where the Hausdorff dimension used is the classical one. Recall that we must prove the existence of a code $\varphi_\infty$ on $A^\infty$ such that $c(\varphi_\infty, S) \leq \dim_H(S) \log |A|$. In order to get rid of the problematic elements of $S$ (that is, those elements which make $\tau : S \to [0,1]$ non-injective), we observe that they are countable so we may list them like this: $\tilde{S} = \{x_1, x_2, \ldots\}$. We then proceed by defining $\tilde{\varphi}_\infty$ as the map which carries $x_i$ into $0\cdots010\cdots$ where the 1 is preceded by $i$ zeroes. This map has $c(\tilde{\varphi}_\infty, \tilde{S}) = 0$ and hence we can go ahead and create $\hat{\varphi}_\infty$ on the remaining sequences and then define $\varphi_\infty(x) = \tilde{\varphi}_\infty(x)$ if $x \in \tilde{S}$ and $\hat{\varphi}_\infty(x)$ otherwise. This argument brings into focus the relationship between the Hausdorff dimension and coding, and demonstrates that it is a strictly asymptotic (or limiting). One may also say, that it is an information theoretical version of lemma §288.

Appendix I

# Index

*If you don't find it in the index, look very carefully through the entire catalogue.*

The Sears & Roebuck Catalogue, 1897

**Citations**