Characterization of Chaitin Machine Satisfying The Algorithmic Coding Theorem

By Yoshibumi Amaya

A thesis submitted to
School of School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of School of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Hajime Ishihara
Professor Hiroakira Ono
Professor Tetsuo Asano

March, 2001

Contents

1 Introduction **3**

2 Preliminaries **6**
2.1 Notation and Defined 6
2.2 Recursive Function Theory 7

3 Noiseless Coding **9**
3.1 Prefix-Free Sets 9
3.2 Prefix Coding 10

4 Program-Size Complexities **14**
4.1 Machines and Complexities 14
4.2 Algorithmic Properties of Complexities 16
4.3 Quantitative Estimates 17
4.4 Halting Probabilities 20

5 Recursively Enumerable Prefix Codes **22**
5.1 Kraft-Chaitin Theorem 22
5.2 Algorithmic Coding Theorem 30

6 Coding with Minimal Programs **33**
6.1 The Condition Under the Semi-distribution 33
6.2 Minimal Programs are Optimal 35
6.3 Algorithmic Coding Theorem Revisited 37

7 Concluding Remarks **40**

Acknowledgment

The author is deeply grateful to Associate professor Hajime Ishihara,
Professor Hiroakira Ono and Professor Tetsuo Asano for their helpful guidance.

Special thanks to members of Ono-Ishihara laboratory and the author's friends who he has met in JAIST for their encouragement.

Finally, the author would also like to thank to my family.

Chapter 1

Introduction

Algorithmic information theory (AIT) is the result of combining Shannon's information theory and Turing's computability theory.

In algorithmic information theory, the primary concept is that of the information content of an individual object which is a measure of difficulty of specifying or describing and constructing or calculating the object. This notion is also known as information-theoretic complexity or Kolmogorov complexity.

On a concrete target, it is to measure the complexity $H_U(x)$ of a string x by the size in bits of the smallest program p for computing x by a partial recursive universal machine U. That is,

$$H_U(x) = \min\{|p| \mid U(p) = x\}.$$

AIT originated with the discovery of universal descriptions. It is based on the foundation of probability theory and randomness and information theory. This fields was independently introduced by A.N.Kolmogorov, R.J.Solomonoff and G.J.Chaitin in the mid 1960s.

First, we describe the difference between Shannon's information theory and AIT. While the classical theory of information is based on Shannon's concept of entropy, AIT adopts the information-theoretic complexity of an individual object as a primary concept. The entropy $\mathcal{H}_P$ is a measure of the degree of ignorance concerning which possibility holds in a set Y endowed with an a priori probability distribution P. Its point of view is largely global; Shannon's amount of information depend on the numbers of objects in Y and don't refer the required bits in an individual object. The entropy is defined as

$$\mathcal{H}_P = -\sum_{y \in Y} P(y) \cdot \log P(y).$$

The classical definition of randomness considered in probability theory allows one to call a process, such as a tossing coin, random. It does not allow one to call a particular outcome (or string of out comes, or sequence of out comes) random, except in an intuitive, heuristic sense. The information-theoretic complexity of an individual object is a measure of the difficulty of specifying the object; it focuses the attention on the individual, allowing us to formalize the randomness intuition.

There are two flows in AIT . In the original formulation of AIT (called AIT_1), the first important result is the Invariance Theorem by A.N.Kolmogorov, R.J.Solomonoff, G.J.Chaitin. respectively, in 1965,in 1964, in 1969.
For some universal machine U, for every machine M, for each finite object (string) x,

$$K_U(x) \leq K_M(x) + O(1).$$

It means that each the object (string) x has an intrinsic complexity which is independent form the ways U, M of description. Hence, we can measure an upper bound of a program-size complexity by fixing the universal machine. The substance of the Invariance Theorem credit A.M.Turing who discovered the universal Turing machine. The other important properties exist in randomness of finite strings, non-computability of program-size complexity, computability of approximations of the program-size complexity, and so on.

In the second formulation is the AIT_2 (called the prefix complexity), which was first introduced by L.A.Levin and G.J.Chaitin in 1970s. The strategy is to measure the program size complexity H_U by (Chaitin) machine U having the prefix free set as its domain. This has nicer mathematical properties than the original AIT_1, and has therefore become something of standard in field. For example, the Kraft's inequality is satisfied, the machine's halting probability is defined, there are much fewer than 2^N possible programs (strings) of size N, and so on. Two important results in AIT_2 was shown by Chaitin, in 1974. One is Kraft-Chaitin Theorem ; Given a recursively enumerable (r.e.) set $S = \{(x_i, n_i) \in \Sigma^* \times N \mid i \geq 0\}$ such that $\sum_i 2^{-n_i} \leq 1$, we can effectively construct a machine M, which is $M(u_i) = x_i$ and $|u_i| = n_i$ for all i.
By using the result, The other important result in AIT_2 is the Algorithmic Coding Theorem; the complexity $H_U(x)$ of any universal machine U are asymptotically optimal (i.e. optimal up to at most a constant $1 + c$) with respect to the machine's algorithmic probabilities $P_U(x)$,

$$H_U(x) + \log P_U(x) \leq 1 + c.$$

$$H_U(x) = \min\{|u| \mid U(u) = x,\ u \in \Sigma^*\},\ P_U(x) = \sum_{U(u)=x} 2^{|u|}.$$

In order to make the Algorithmic Coding Theorem more general, we are interested in a class of machines, not necessarily universal, and any semi-distribution, not the machine's algorithmic probabilities, and a constant c. So, the aim of this thesis is, after adequately investigating and considering fundamental concepts, which is the noiseless coding and the program-size complexity with Chaitin machine, to investigate a class of machines satisfying the Algorithmic Coding Theorem, not necessarily universal, for any semi-distribution. Finally, we characterize all machines satisfying the Algorithmic Coding Theorem, and we show a class of machines satisfying the Algorithmic Coding Theorem with constant $c = 0$.

Finally, we easily describe about the application. One can distinguish three application areas. That is, we can use the fact that many strings are extremely compressible; that using fact that the compressibility of strings as a selection criterion; using fact that many strings are not compressible at all; that fact that some strings may be compressed, but that it takes a lot of effort to do so. The concepts and results of AIT are relevant for other subjects, for instance for logic (new light is shed on Gödel's incompleteness result), physics

(chaotic motion), biology (how likely is life to appear and evolve?), and metaphysics (how ordered is universe?) and so on.

We give the structure of this thesis . In Chapter 2, we present the prerequested background, i.e. relevant notation of sets, functions, strings and results from recursion theory which is vary important in measuring the program size. In Chapter 3, we show the property of prefix-free sets and prefix codes using in Chaitin machine, and Kraft's inequality. Latter on, we show the relevant of the prefix code strings and the Shannon's entropy as noiseless coding theorem. In Chapter 4, we show the definition of Chaitin machines, program-size complexity, the machine's algorithmic probability and halting probability. We derive non-computability of program-size complexity, computability of approximations to the program-size complexity and some elementary estimation for complexities. In Chapter 5 we introduce two important tools: the Kraft-Chaitin Theorem (an extension of Kraft's inequality for the construction of prefix codes corresponding to arbitrary recursively enumerable codes) and relativized complexities and probabilities. Consequently, we show the Algorithmic Coding Theorem. In Chapter 6 we investigate machines, not necessarily universal, satisfying Algorithmic Coding Theorem under conditions of a given semi-distribution. Finally, we show the characterization of all machines satisfying the Algorithmic Coding Theorem.

Chapter 2

Preliminaries

The first we present the notation and the basic background required to read this thesis, i.e. relevant set, strings, function and results from recursion theory.

2.1 Notation and Defined

The set of natural numbers is denoted by N; $N_+ = N \setminus \{0\}$. The set of real is denoted by R; $R_+ = R \setminus \{x \in R \mid x \leq 0\}$.

The concepts about functions play an very important role in the computer science. A *partial function* $f : X \xrightarrow{o} Y$ is a defined on a subset Z of X. Z is called *domain* of the function f (written $dom(f)$). In case $dom(f) = X$, a function f is called a *total* function, and exhibited by writing $f : X \to Y$. If $x \in dom(f)$, then we say that $f(x)$ is defined (written $f(x) \neq \infty$). And if $x \notin dom(f)$, then we say that $f(x)$ is undefined (written $f(x) = \infty$). The *range* of f is $range(f) = \{f(x) \mid x \in dom(f)\}$.

A (partial) function f is *injective* if, for all $x, y \in dom(X)$, $f(x) = f(y)$ implies $x = y$. Stated differently, if $x \neq y$ implies $f(x) \neq f(y)$. A (partial) function f is *surjective* if, for all $y \in Z$, there is a $x \in X$ such that $f(x) = y$. And a (partial) function f is called bijective if f is surjective and injective. For example, the total function $f : N \to N$,where is $f(x) = x^2$, is injective and surjective. But in case $f : R \to R$, it is not injective and not surjective. About the sets and functions more see [10].

We will use the following function. The function $\lfloor \alpha \rfloor$ is floor of the real α, $\lceil \alpha \rceil$ the is ceiling of the real α. For example, $\lfloor 3.14 \rfloor = 3$, $\lfloor -3.14 \rfloor = -4$, $\lceil 3.14 \rceil = 3$, $\lceil -3.14 \rceil = -3$. The other functions will be defined when it is requested.

We fix an alphabet $\Sigma = \{0, 1\}$. A element of alphabet is called a *letter*. By $\Sigma^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, \cdots\}$, denote the set of all strings $a_1 a_2 \cdots a_n$ with elements $a_i \in \Sigma (1 \leq i \leq n)$ (with λ denoting the empty string, with no letters.); $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. Notice that a '*string*' is a finite binary string, Σ^* contains the empty string λ and don't contain infinite binary strings. In case a infinite binary string, which is usually called a '*sequence*'. Clearly, for all strings $u \in \Sigma^*$, $\lambda x = x\lambda = x$. The length of a string $s \in \Sigma^*$ is denoted by $|s|$ ($|\lambda| = 0$). For all $x \in \Sigma^*$ and all $i \in N_+$, x^i is the concatenation $xxx \cdots x$ (i times); $x^0 = \lambda$. For $n \in N$, the set $\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}$. If the set S is a finite

set, then $\sharp S$ is the *cardinality* of S. For example, $\sharp\{x \in \Sigma^* \mid |x| = n,\ n \in N\} = 2^n$ and $\sharp\{x \in \Sigma^* \mid |x| \leq n,\ n \in N\} = 2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^n = 2^{n+1} - 1$. We now consider a correspondence of binary strings and natural numbers. Hence, we induce a *quasi-lexicographical* order (written $x < y$) on Σ^*: In case $|x| < |y|$ or if $|x| = |y|$ then there exists a natural k, $1 \leq k \leq n$, $a_1 \cdots a_{k-1} = b_1 \cdots b_{k-1}$, $a_k = 0$ and $b_k = 1$ (where is $a, b \in \{0, 1\}$, $x = a_1 \cdots a_n$, $y = b_1 \cdots b_n$). That is,

$$\lambda < 0 < 1 < 00 < 01 < 10 < 11 < 000 < 001 < \cdots.$$

Notice that for every $x \in \Sigma^*$, $x \not< x$. We denote by $string(n)$ the n th string according to the quasi-lexicographical order. In this way we get a bijective function $string : N \to \Sigma^*$;

$$\{(n, string(n))\} = \{(0, \lambda), (1, 0), (2, 1), (3, 00), (4, 01), (5, 10), (6, 11), (7, 000), \cdots\}.$$

Notice that binary representation for the natural numbers is different from the standard binary representation. It is seen that $|string(n)| = \lfloor \log(n + 1) \rfloor$. Notice that we regard log as the base 2 logarithm. The base 10 logarithm is not used in this thesis. For example, $|string(5)| = \lfloor \log 6 \rfloor = \lfloor \log 2^{2.32\cdots} \rfloor = 2$.

Finally, we exhibit the notation of an order as the following is often used in order to measure program-size complexity. (see Chapter 4)

Let $f,\ g : \Sigma^* \to R_+$ be two (partial) functions. In this thesis, we often use $f(x) \leq g(x) + O(1)$ if there exists a constant $c > 0$ such that $f(x) \leq g(x) + c$, for all strings $x \in \Sigma^*$. And we will use the notation $f(x) = g(x) + O(1)$ if $f(x) \leq g(x) + O(1)$ and $g(x) \leq f(x) + O(1)$. This is used to measure the program-size. In general,

$$\begin{aligned} O(F) \ = \ & \{G : \Sigma^* \to R_+ \mid \text{ there exist } c \in R_+, m \in N \text{ such that} \\ & G(x) \leq cF(x), \text{ for all strings } x, |x| \leq m\}. \end{aligned} \tag{2.1}$$

So, $O(1)$ means the set of the constant functions i.e.$O(1) = \{G \mid G(x) \leq c\}$.

2.2 Recursive Function Theory

Algorithmic information theory is essentially based on recursion theory. Informally, an algorithm(program) for computing a partial function $\varphi : N \xrightarrow{o} N$ is a finite set of statements which, given an input $x \in dom(\varphi)$ an algorithm (program), yields an output $y = \varphi(x)$ after a finite number of steps. And the algorithm must specify how to obtain each step in the computation from the previous steps and the input. In case we can specifically exhibit an algorithm computing the function φ, we call it a *partial computable* (or *recursive*) *function*-abbreviated *p.r.function* function. If φ is total, then it is called *computable (recursive) function*. Formally, there are many equivalent ways to define, for example, with Turing machines, λ-calculus, S programming languages, etc. (see [7], [8], [9])

In case a input x is a string, we can say that is not distinguish form the definition of the above. Because there is the bijective relation between all strings and all natural numbers by the total function $string : N \to \Sigma^*$.

The main result is the possibility of enumerating all p.r.functions

$$\varphi_e : (\Sigma^*)^n \xrightarrow{o} \Sigma^*,$$

where e is a code number of some a p.r.function and n is a argument.
That is,

Theorem 2.1 (Universality Theorem) *There is a p.r.function such that*

$$\varphi^2(e, x) = \varphi_e^1(x).$$

Notice that since there exist the pairing functions i.e.recursive bijective functions $<>$: $\Sigma^* \times \Sigma^* \to \Sigma^*$, it is no problem in case the input x has many variables ($x = (x_1, x_2, x_3, \cdots)$). A p.r.function which satisfy Universality Theorem is called *universal p.r.function*, which is given as inputs to the p.r.function both the coding number (program) and the input . And the universal p.r.function simulate with a p.r.function corresponding the code number e consequently, returns same outputs. This result is very important, we will use Universality Theorem in case we define a universal Chaitin machine and, consequently we will measure a program size by using it'machine in Chapter 4

We describe about the basic property and definition in recursive theory of which will be used in this thesis . To more see [7].

A set $x \subset \Sigma^*$ is *recursive* if its characteristic function is recursive. A set $X \subset \Sigma^*$ is *recursively enumerable*-abbreviated-r.e.-if it is either empty or else the range of some recursive function. Equivalently, X is r.e. if it is the domain of a p.r.function.

An infinite r.e.set is the range of some injective recursive function. Every infinite r.e.set has an infinite recursive subset, etc. These facts will be used in Chapter 4, 5.

Chapter 3

Noiseless Coding

In this Chapter we consider the problem of safe transmission of a message over a channel, which cannot be affected by noise. We rely mainly on the following two central tools: prefix-free sets and Shannon's entropy. The prefix-free sets is the easiest codes to construct and exists most interesting problems on codes. And the prefix-free set is used to the input of Chaitin machines. Shannon entropy is a measure of the degree of ignorance concerning which possibility holds in an ensemble with a given a priori probability distribution. We show the property of prefix-codes and prefix-free set. The concepts of prefix-free and prefix code is used the Chaitin machine in Chapter 4. Later on, we show the relevant of the Shannon entropy and the lengths of prefix-code strings as Shannon's Noiseless Coding Theorem .

3.1 Prefix-Free Sets

Definition 3.1 a) A string $x \in \Sigma^*$ is a *prefix* of another string $y \in \Sigma^*$ (written $x <_p y$) if $y = xz$, for some string $z \in \Sigma^*$. For example, for all $x \in \Sigma^*$, $\lambda <_p x$.
b) A set $S \subset \Sigma^*$ is called *prefix-free* in case for all strings $x, y \in \Sigma^*$, $x <_p y$ implies $x = y$. For example, The set $S = \{1^i 0 \mid i \in N\}$ is prefix-free set.

Example 3.2 For every natural n, the set $S = \Sigma^n$ is a prefix-free set. Moreover, every prefix-free set S containing the empty string λ is equal to $\Sigma^0 = \{\lambda\}$.

Here, we consider about the way of the corresponding a positive number n and a element of a prefix-free sets. For example, the set $S = \{1^n 0 \mid n \in N_+\}$ requires $n+1$ bits to represent n. We improve this condition with the following the way.

We introduce the function $bin : N_+ \to \Sigma^*$, where $(n)_2 = 1bin(n)$. $(n)_2$ is the base-2 representation of the number n, i.e. $\{(n, bin(n))\} = \{(1, \lambda), (2, 0), (3, 1), (4, 00), \cdots\}$. It is seen that for every natural $n \in N$, $|bin(n)| = \lfloor logn \rfloor$. For every $x \in \Sigma^*$ we construct the new string $\overline{x}$ by inserting a 0 in front of each letter in x, and adding finally 1; $\lambda = 1$. For example,

$$\overline{0} = 001,\ \overline{1} = 011,\ \overline{00} = 00001,\ \overline{01} = 0001.$$

It is clear that $|\overline{x}| = 2|x| + 1$. Finally, let

$$d(x) = \overline{bin(|x|)}x$$

, for all $x \in \Sigma^+$. $d(x)$ is called the *self-delimiting code* of x. For example,

$$d(0101) = \overline{bin(4)}0101 = \overline{00}0101 = 000010101.$$

Example 3.3 The set $S = \{d(x) \mid x \in \Sigma^+\}$ is prefix-free and every string $x \in \Sigma^+$ can be represented by using $|d(x)| \leq |x| + 2\log|x| + 1$ bits. Moreover, each positive number n can be represented by a string in S within $\log n + 2\log\log n + 1$ bits. (notice $\log n + 2\log\log n + 1 < n + 1$.)

A way of thinking a self-delimiting code is what it was used by C.E.Shannon.

In next section, we define about the average length of prefix-code and Shannon's entropy, we show about those relations.

3.2 Prefix Coding

Let Y be the information source set and A be the set of input of a communication channel, where consider finite or infinite $Y \subset \Sigma^*$ and $A = \Sigma^*$.

Definition 3.4 a) A *code* is an injective function $C : Y \subset \Sigma^* \to \Sigma^*$. The element of C are called *code-strings*.
b) An *prefix code* (*instantaneous code*) is a code C such that $C(Y)$ is prefix free.

Example 3.5 Let $Y = \{y_1, y_2, y_3, y_4\}$. Consider the following functions defined on Y :

Y :	y_1	y_2	y_3	y_4
C_1 :	00	01	10	11
C_2 :	10	110	1110	11110
C_3 :	01	011	0111	01111
C_4 :	0	00	000	0000
C_5 :	10	10	110	110

The codes C_1, C_2 are prefix codes while the code C_3, C_4 is not, C_5 is not even a code.

In what follows we are concerned with prefix codes. Their main property is *uniqueness of decodability*. A code is *uniquely decodable* if the unique extension of from any finite sequence made by a code C to sequence of strings in Y is injective. For example the sequence

$$0010001101$$

constructed by strings in code $C_1(Y)$ can be split as

$$00, 10, 00, 11, 01$$

and is uniqueness decoded as

$$y_1, y_3, y_1, y_4, y_2.$$

Not every uniquely decodable code is prefix code. For example, C_3 is uniquely decodable, but it is not prefix code. The advantage of sets of prefix code strings is possible to decode without delay. Because the end of a code-string can be immediately recognized and subsequent parts of message don't have to be observed before the decoding is started.

A simple way of building prefix code-strings is given by the following theorem. Moreover, in Chapter 5 this theorem is extended for every r.e.sets.

Theorem 3.6 (Kraft) *Let $n_1,\ n_2, \cdots n_N$ be a finite sequence of positive integers. These numbers are the lengths of each prefix code-string iff $\sum_{i\geq 1} Q^{-n_i} \leq 1$.*

Proof. Let the set $Y \subset \Sigma^*$ and the prefix code $C : Y \to \Sigma^*$ such that $|C(y_i)| = n_i$ with $i \in N$. Let r_k be the number of the code-strings having length k. Let $m = \{n_1, \cdots, n_N\}$. In case $m < j,\ r_j = 0$. Since the code C is a prefix code, the following relations hold true:

$$\begin{aligned} r_1 &\leq 2 \\ r_2 &\leq (2 - r_1)2 = 2^2 - r_1 2^1, \\ r_3 &\leq ((2 - r_1)2 - r_2)2 = 2^3 - r_1 2^2 - r_2 2^1, \\ &\vdots \\ r_m &\leq 2^m - r_1 2^{m-1} - \cdots - r_{m-1} 2^1. \end{aligned} \tag{3.1}$$

Dividing the inequality (3.1) by 2^m we get

$$\sum_{k=1}^{m} r_k 2^{-i} \leq 1.$$

Hence,

$$\sum_{k=1}^{m} r_k 2^{-i} = \sum_{i=1}^{N} 2^{-n_i} \leq 1. \tag{3.2}$$

For the converse implication, we use (3.1):

$$\begin{aligned} r_1 2^{-1} &\leq \sum_{i=1}^{m} \leq 1, \\ r_1 2^{-1} + r_2 2^{-2} &\leq \sum_{i=1}^{m} \leq 1, \\ &\vdots \\ r_1 2^{-1} + r_2 2^{-2} + \cdots + r_m 2^{-m} &\leq \sum_{i=1}^{m} \leq 1. \end{aligned}$$

Multiply each the inequality by 2^1, 2^2, $\cdots$, 2^m, respectively.
We get:

$$r_1 \leq 2^1,$$

$$\begin{aligned} r_2 \leq (2 - r_1)2 &= 2^2 - r_1 2^1, \\ &\vdots \\ r_m &\leq 2^m - r_1 2^{m-1} - \cdots - r_{m-1} 2^1. \end{aligned}$$

It means that we can construct the prefix code whose code-strings have lengths $n_1, n_2, \cdots,$. Notice that this result holds also in infinite sequence (see [6]). ∎

Remark. The inequality $\sum_{i \geq 1} 2^{-n_i} \leq 1$ is called Kraft's inequality.

Proposition 3.7 *For every prefix-free set $S \subset \Sigma^*$,*

$$\sum_{u \in S} 2^{-|u|} \leq 1.$$

By the direct implication in Theorem 3.6, it is clear. ∎

Kraft's Theorem does not assert that every code which satisfies the inequality therein must be a prefix code. A counter-example is offered by the C_2: it satisfies Kraft's inequality, but it is not prefix-free. Nevertheless, there is a prefix code C_2 whose lengths of string-codes are equal to those of the code C_3. The relation between these codes is a special instance of the following more general result.

Theorem 3.8 *If a code is uniquely decodable with code-strings of lengths $n_1, n_2, \cdots, n_N$, then Kraft's inequality is satisfied.*

Proof. Let be a positive integer. Then

$$\begin{aligned} \left(\sum_{k=1}^{N} 2^{-n_k} \right)^r &= \sum_{k_1=1}^{N} 2^{-n_{k_1}} + \sum_{k_2=1}^{N} 2^{-n_{k_2}} + \cdots \sum_{k_r=1}^{N} 2^{-n_{k_r}} \\ &= \sum_{k_1=1}^{N} \sum_{k_2=1}^{N} \cdots \sum_{k_r=1}^{N} 2^{-(n_{k_1} + n_{k_2} + \cdots + n_{k_r})}, \end{aligned}$$

because a finite number of terms can be always rearranged without affecting their sum. Now $n_{k_1} + n_{k_2} + \cdots + n_{k_r}$ is exact the number of code letters in some sequence of r code-strings. The numbers $k_1, k_2, \cdots, k_r$ vary, so all possible sequence of r code-strings are in this way generated. Let r_i be the number of sequences of r code-strings which contain i letters;clearly, $1 \leq i \leq rm$, where $m = max\{n_1, n_2, \cdots, n_r\}$. So,

$$\left(\sum_{k=1}^{N} 2^{-n_k} \right)^r = \sum_{i=1}^{rm} r_i 2^{-i}.$$

Since the code is uniquely decodable all sequences of r code-strings with a total of i letters have to be distinct, i.e. $r_i \leq 2^i$. Accordingly,

$$\sum_{k=1}^{N} 2^{-n_k} \leq \left(\sum_{i=1}^{rm} r_i 2^{-i} \right)^{\frac{1}{r}} = (rm)^{\frac{1}{r}}.$$

Allowing r to tend to N , the right-hand side tends to 1.

Corollary 3.9 *Each uniquely decodable code can be replaced by a prefix code without changing the lengths of the code-strings.*

Proof. By Theorem 3.6 and Theorem 3.8, it is clear.

Shannon discovered that the average length of prefix code is about equal to the entropy of the source strings set. This is known as the *Noiseless Coding Theorem*. The adjective "noiseless" emphasizes that we ignore the possibility of errors. Shannon's classical argument can be expressed for semi-distributions as follows.

Definition 3.10 A *semi-distribution* is a function $P : Y \subset \Sigma^* \to [0, 1]$ such that

$$\sum_{y \in Y} p(y_i) \leq 1.$$

If $\sum_{y \in Y} P(y_i) = 1$ then P is called *a distribution*. The *self-information* of y is defined by

$$I(y) = -\log P(y).$$

Definition 3.11 The *entropy* is defined as the average of self-information with semi-distributions P on the set Y by

$$\mathcal{H}_P = -\sum_{y \in Y} P(y) \cdot \log P(y)$$

Definition 3.12 The *average code-string length* of a prefix-code C with respect to a semi-distribution P is

$$L_{C,P} = \sum_{y \in Y} P(y) \cdot |C(y)|.$$

For example, let the source set $Y = \Sigma^*$, the prefix code $C(y) = y_1 y_1 \cdots y_n y_n 01$, $P(y) = 2^{-2|y|-3}$. Then, $L_{C,P} = \sum_y 2^{-2|x|-3} \cdot 2|y| + 2, H_P = \sum_y 2^{-2|x|-3} \cdot 2|y| + 3$,

$$\mathcal{H}_P - 1 \leq \mathcal{H}_P - \frac{1}{4} = L_{C,P} < \mathcal{H}_P + 1.$$

The relation between the entropy and the average length of the prefix code is as the following. Shannon's classical argument can be expressed for semi-distributions. (see more [6])

Theorem 3.13 (Noiseless Coding Theorem;Shannon) *Let $C : Y \to \Sigma^*$ be an prefix code a semi-distribution P on the set Y. Then,*

$$\mathcal{H}_P - 1 \leq \mathcal{H}_P + \left(\sum_{y \in Y} P(y)\right) \cdot \log \left(\sum_{y \in Y} P(y)\right) \leq L_P \leq \mathcal{H}_P + 1.$$

Notice that $-\frac{1}{2} \leq \left(\sum_{y \in Y} P(y)\right) \cdot \log \left(\sum_{y \in Y} P(y)\right) \leq 0$.

Chapter 4

Program-Size Complexities

We consider a machine as a partial recursive function which reads a string (program) as an input and then may or not print another string, as output. With reference to a fixed machine, the complexity of a string x is defined as the length of the shortest string y which when the input to the machine will determine the output of x. If one chooses to think of the input as a program + date, then the machine acts as an unary partial function. If the program and data is read separately, then the machine will be a binary partial recursive function. Here we considers the machine as an unary partial function and uses Chaitin's motivation [4] [1]. Later on, we show the machine's algorithmic probability and halting probability and derive non-computability of program-size complexity, computability of approximations to the program-size complexity and some elementary estimation for complexities.

4.1 Machines and Complexities

Definition 4.1 A *Chaitin Machine* is a p.r. function $M : \Sigma^* \stackrel{o}{\to} \Sigma^*$, such that the domain of M is prefix-free.

If for every strings $x \in \Sigma*$ $M(x) \neq \infty$, $y <_p x$ and $y \neq x$, then $M(y) = \infty$. The Chaitin machine M can be imagined as a function computable by a special Turing machine; the domain of the function is prefix-free. In what follows we will operate only with Chaitin machines, so which will be simply referred as machines.

Definition 4.2 A machine U is *universal* if for every machine M there is a constant c_M with the following property: if $M(x) \neq \infty$, then there is a string $x' \in \Sigma^*$ such that $U(x') = M(x)$ and $|x'| \leq |x| + c_M$; c_M is simulation constant of M on U.

Theorem 4.3 *There exists a universal Chaitin machine.*

Proof. By Theorem 2.1 we can let F: $N_+ \times \Sigma^* \stackrel{o}{\to} \Sigma^*$ be a universal p.r. function for the class of all p.r. functions M: $\Sigma^* \stackrel{o}{\to} \Sigma^*$ such that the set $\{u \in \Sigma^* \mid M(u) \neq \infty\}$ is prefix-free i.e. $F(n, u) = M_n(u)$. Then put:

$$U(1^n 0u) = F(n, u).$$

We fix a universal machine U as standard universal machine for measuring program size complexities. Notice that the universal machine is not only what we fixed one.

Corollary 4.4 *Every universal machine U is surjective.*

Proof. For every the strings $z \in \Sigma^*$, let the machine $M(\lambda) = z$. By Definition 4.2, for every M there exists a string $x' \in \Sigma^*$ such that $U(x') = M(\lambda) = z$. ∎

Definition 4.5 a)The *program-size complexity* induced by the machine M is

$$H_M(x) = min\{|z| \mid z \in \Sigma, M(z) = x\}.$$

In case $M = U$ we put $H_M(x) = H_U(x)$. The minimum of empty set is undefined ($= \infty$).
b)The *minimal (canonical)* program is defined with respect to machine M by

$$x^*_M = min\{u \in \Sigma^* \mid M(u) = x\},$$

where the minimum is taken according to the quasi-lexicographical ordering of strings (the empty string $\lambda < 0 < 1 < 00 < 01 < 10 < 11 < 100 < \cdots$).
In case $M = U$ we put $x^*_M = x^*_U$.

We can image a string x as an object, a function M as the interpreter or decoding function and a string z as an program.

Theorem 4.6 (Invariance Theorem) *For some universal machine U, for every machine M there exists a constant $c > 0$ (depending upon M on U) such that for every $x \in \Sigma^*$,*

$$H_U(x) \leq H_M(x) + c. \tag{4.1}$$

Proof. We use Theorem 4.3. Let $H_U(x) = min\{|0^n 1u| \mid u \in \Sigma^*, U(0^n 1u) = x\}$, $H_M(x) = min\{|u| \mid u \in \Sigma^*, M(u) = x\}$ and $c_M = n + 1$. ∎
Remark. In Theorem 4.6, put $M = U'$. For every universal machines U, U' there exists a constant $c > 0$ such that

$$|H_U(x) - H_{U'}| \leq c,$$

for all $x \in \Sigma^*$. Hence, universal machine's complexities don't depend on the chosen universal machines for except the constant.

Lemma 4.7 *For every $x \in X$:*

$$x^*_U \text{ exists and } x^*_U \neq \lambda, \tag{4.2}$$

$$x = U(x^*_U), \tag{4.3}$$

$$H_U(x) = |x^*_U|. \tag{4.4}$$

Proof. For (4.2), since the universal machine is surjective by Corollary 4.4, for every $x\Sigma^*$ there is a $u \in \Sigma^*$ such that $U(u) = x$. If $\lambda \in dom(U)$, then $dom(U) = \{\lambda\}$. Since U is surjective, so $\lambda \notin dom(U)$. For (4.3), the string x^*_U is one of $dom(U)$ such that $U(z) = x$. For (4.4) is clear by Definition 4.5. ∎

4.2 Algorithmic Properties of Complexities

Let *the set of minimal(canonical) program* of the machine$M$ be

$$CP_M = \{x^*_M \mid x \in \Sigma^*\}.$$

The set $S \subset \Sigma^*$ is an *immune* set if the set S has no infinite r.e. subset. (see [8] about the immune set)

Theorem 4.8 *The set CP_U is infinite and has no infinite r.e.subset.*

Proof. Since a universal machine U is surjective, we can put $k : \Sigma^* \to \Sigma^*$ be the total function given by $k(x) = x^*_U$. If for every $x^*_U,\ x^{*'}_U \in \Sigma^*$, $x^*_U = x^{*'}_U \neq x'$ imply by (4.3) $x = U(x^*_U) = x' = x^{*'}_U$. Hence the function k is injective. So $rang(k) = CP_U$ is infinite. We prove now by contradiction, which there exists an infinite r.e.set $S \subset CP_U$. Let S be enumerated by the injective recursive function $f : N \to \Sigma^*$ i.e. $S = \{f(0), f(1), f(2) \cdots\}$. Define the function $g : N \to \Sigma^*$ by

$$g(0) = f(0), g(n+1) = f(\min j[|f(j)| > n+1]).$$

It is easy to check the function g is recursive, the set $S' = g(N^+)$ is r.e.set, infinite and $S' \subset S$. And since $g(|n+1|) = f(\min j[|f(j)| > n+1]) > n+1$, $|g(n)| > n$, for every $n > 0$. Here, we construct a machine M such that for every $i \geq 1$, there exists a string $u \in \Sigma^*$ such that

$$M(u) = g(i), |u| \leq \log i + 2\log\log i \leq 3\log i.$$

By Invariance Theorem we get a constant c_1 such that for all $i \in N$,

$$H(g(i)) \leq H_M(g(i)) + c_1 \leq 3\log i + c_1 \tag{4.5}$$

Intermediate Step. There exists a constant $c_2 \geq 0$ such that for every $x \in CP_U$,

$$H(x) \geq |x| - c_2. \tag{4.6}$$

Construct machine

$$D(u) = U(U(u))$$

and pick the constant c_2 coming from the Invariance Theorem. Take $x = y^*, z = x^*$. By (4.3), one has :

$$D(z) = U(U(z)) = U(U(x^*)) = U(x) = U(y^*) = y,$$

so

$$H_D(y) \leq H_U(x),$$

$$|x| = |y^*| = H_U(y) \leq H_D(y) + c_2 \leq H(x) + c_2.$$

For all $i \geq 1$, if $g(i) \in CP_U$, then $|g(i)| > i$, so by (4.5) and (4.6)

$$i - c_2 < |g(i)| - c_2 \leq H(g(i)) \leq 3\log i + c_1,$$

and consequently only a finite number of elements in S' can be in CP_U. ∎

Remark. The set CP is not r.e.set. (cf. Every infinite r.e.set has an infinite recursive subset.)

Corollary 4.9 *The function $f : \Sigma^* \to \Sigma^*$, $f(x) = x^*_U$ is not recursive.*

Proof. The range of the function f is CP_U, injective and CP_U is not r.e.set. So, the function f is not recursive. ∎

Definition 4.10 *labeldf:semi* A function f is *semi-computable from below(above)* in case the set $\{(x, r) \mid x \in \Sigma^*, r \in Q, f(x) > r(f(x) < r)\}$ is r.e. Q : the of rational numbers.

Theorem 4.11 *The program-size complexity $H_U(x)$ is semi-computable from above, but not recursive.*

Proof. We prove that the set $\{(x, n) \mid x \in \Sigma^*, n \in N, H_U(x) < n\}$ is r.e. Since $H_U(x) < n$ iff there exist $y \in \Sigma^*$ and $t \in N$ such that $|y| < n$ and $U(y) = x$ in most t steps.
For the second part of the theorem we prove a bit more, namely:
There is no p.r.function $\varphi : \Sigma^* \stackrel{o}{\to} N$ with infinite domain and such that $H_U(x) = \varphi(x)$, for all $x \in dom(\varphi)$.
Assume that $H(x) = \varphi(x)$, for all $x \in dom(\varphi)$, where $\varphi : \Sigma^* \stackrel{o}{\to} N$ is a p.r. function with an infinite domain. Let $B \subset dom(\varphi)$ be a recursive, infinite set and let $f : \Sigma^* \stackrel{o}{\to} \Sigma^*$ be the partial function given by

$$f(a_1^i a_2) = \min\{x \in B \mid H(x) \geq 2^i,\ i > 0\}.$$

Since $\varphi(x) = H(x)$,for $x \in B$, it follows that f is a p.r.function. Moreover, f has a recursive graph and f takes as values strings of arbitrarily long length. For infinitely many $i > 0$,

$$H(f(0^i 1)) \geq 2^i.$$

Accordingly, in view of the Invariance Theorem, for infinitely many $i > 0$, we have:

$$2^i \leq H_U(f(0^i 1)) \leq H_C(f(0^i 1)) + c \leq i + 1 + c.$$

This yields a contradiction. ∎

4.3 Quantitative Estimates

We give some elementary estimations for the program-size complexity

Theorem 4.12 *There exists a constant $c > 0$ such that for all $x \in \Sigma^+$; $\Sigma^+ = \Sigma^*/\{\lambda\}$,*

$$H_U(x) \leq |x| + 2\log|x| + c. \tag{4.7}$$

Proof. By Example 3.3, we construct the machine $M(d(x)) = x$, where $|d(x)| = |x| + 2\log|x| + 1$. For all $x \in \Sigma^+$,

$$H_M(x) \leq |d(x)| = |x| + 2\log|x| + 1.$$

By Invariance Theorem,

$$H_U(x) \leq H_M(x) + c \leq |d(x)| = |x| + 2\log|x| + c.$$

∎

Lemma 4.13 *For every machine M and each natural n,*

$$\sharp\{x \in \Sigma^* \mid H_M(x) = n\} \leq 2^n. \tag{4.8}$$

Proof. By $\sharp\Sigma^n = 2^n$, it is clear. ∎

Proposition 4.14 *Let $E \subset \Sigma^*$ be a set having $m \geq 1$ elements and $\varepsilon > 0$. Then for every machine M,*

$$\sharp\{x \in E \mid H_M(x) \leq \log m - \varepsilon\} > m\left(1 - 2^{1-\varepsilon}\right). \tag{4.9}$$

Proof.

$$\begin{aligned}
\sharp\{x \in E \mid H_M(x) \leq \log m - \varepsilon\} &= m - \sharp\{x \in E \mid H_M(x) < \log m - \varepsilon\} \\
&\geq m - \sharp\{x \in E \mid H_M(x) < \lfloor \log m - \varepsilon \rfloor + 1\} \\
&\geq m - \sharp\{x \in \Sigma^* \mid H_M(x) < \lfloor \log m - \varepsilon \rfloor + 1\} \\
&= m - \sum_{0 \leq n \leq \lfloor \log m - \varepsilon \rfloor} \sharp\{x \in \Sigma^* \mid H_M(x) = n\} \\
\text{by (4.8)}, \quad &\geq m - \sum_{0 \leq n \leq \lfloor \log m - \varepsilon \rfloor} 2^n \\
&= m - (2^{\lfloor \log m - \varepsilon \rfloor + 1} - 1) \\
&\geq m - (2^{\log m - \varepsilon + 1} - 1) = m - (2^{\log m} 2^{1-\varepsilon} - 1) \\
&= m(1 - 2^{1-\varepsilon}) + 1 > m(1 - 2^{1-\varepsilon})
\end{aligned}$$

∎

Corollary 4.15 *For every machine M, natural n and positive real ε,*

$$\sharp\{x \in \Sigma^n | H_M(x) \leq n - \varepsilon\} > 2^n(1 - 2^{1-\varepsilon}). \tag{4.10}$$

Proof. In Proposition4.14, for each $n \in N$ let $E = \Sigma^n$. By $m = \sharp\Sigma^n = 2^n$,

$$\sharp\{x \in \Sigma^n \mid H_M \leq n - \varepsilon\} > 2^n(1 - 2^{1-\varepsilon})$$

∎

Proposition 4.16 *If $F : \Sigma^* \to N$ is a function such that*

$$H_U(x) \leq F(x) + O(1)$$

then

$$\sharp\{x \in \Sigma^* | F(x) < m\} < 2^{m+O(1)}.$$

Proof. By $H_U(x) \leq F(x) + c \leq m + c$, for some constant $c > 0$, $\{x \in \Sigma^* \mid F(x) < m\} \subset \{x \in \Sigma^* \mid H_U(x) < m + c\}$. Hence,

$$\begin{aligned}
\log \sharp\{x \in \Sigma^* \mid F(x) < m\} &\leq \log \sharp\{x \in \Sigma^* \mid H_U < m + c\} \\
&= \log \sum_{0 \leq n \leq \lfloor m+c \rfloor} \{x \in \Sigma^* \mid H_U(x) = n\} \\
&= \log \sum_{0 \leq n \leq \lfloor m+c \rfloor} 2^n \\
&\leq \log 2^{\lfloor m+c \rfloor + 1} - 1 \\
&\leq \log(2^{m+c} - 1) \\
&< \log w^{m+c}.
\end{aligned}$$

∎

Proposition 4.17 *Let $F : A^* \to N$ be a function semi-computable from above. If there exists a constant $q > 0$ such that for all natural $m > 0$*

$$\sharp\{x \in \Sigma^* | F(x) < m\} < \log m + q,$$

then $H_U(x) \leq F(x) + O(1)$.

Proof. Let $\{(x_1, m_1), (x_2, m_2), \cdots\}$ be an injective recursive enumeration of the r.e.set $\{(x, m) \in \Sigma^* \times N \mid F(x) < m\}$. Construct machine M by the following algorithm:

1. All strings $y \in \Sigma^*$ is available.
2. For $i = 1, 2, \cdots$ generate (x_i, m_i), choose the first available $y_i \in \Sigma^{\log m_i + q}$ and put $M(d(y_i)) = x_i$.
3. The string y_i is no longer available.

In case of $F(x) < m$ there exists $y \in \Sigma^{\log m + q}$ such that $M(d(y)) = x$, so

$$\begin{aligned}
H_M(x) \leq |d(y)| &= |y| + 2\log|y| + 1 \\
&= \log(m + q) + 2\log(\log m + q) + 1 \\
&\leq \log m + 2\log\log m + k \quad (\text{ constant } k > 0).
\end{aligned}$$

In particular, $F(x) < F(x) + 1 = m$,so

$$\begin{aligned}
H_M &\leq \log(F(x) + 1) + 2\log\log(F(x) + 1) + O(1) \\
&\leq F(x) + O(1).
\end{aligned}$$

By Invariance Theorem,

$$H_U \leq H_M + O(1) \leq F(x) + O(1).$$

∎

4.4 Halting Probabilities

Following Chaitin, we switch the point of view, from a deterministic one to a probabilistic one.

Definition 4.18 a)The *algorithmic probability* of the machine M to produce the output x is

$$P_M(x) = \sum_{\{u \in \Sigma^* | M(u)=x\}} 2^{-|u|}.$$

b)The *halting probability* of M is

$$\Omega_M = \sum_{x \in \Sigma^*} P_M(x) = \sum_{u \in dom(M_\lambda)} 2^{-|u|}$$

In the case $M = U$, we put $P_M = P_U$ and $\Omega_M = \Omega_U$.

Lemma 4.19 *For every machine M and all strings x,*

$$0 \leq P_M(x) \leq \Omega_M \leq 1. \tag{4.11}$$

Proof. By Proposition 3.7, For every machine M, all strings $x \in \Sigma^*$,

$$P_M(x) \leq 1,$$

$$\Omega_M = \sum_{x \in \Sigma^*} \sum_{\{u \in \Sigma^* \mid M(u)=x\}} 2^{-|u|} = \sum_{u \in dom(M)} 2^{-|u|} \leq 1.$$

∎

Lemma 4.20 *For every machine M and all strings x,*

$$P_M(x) \geq 2^{-H_M(x)}. \tag{4.12}$$

Proof.

$$P_M(x) = \sum_{\{u \in \Sigma^* \mid M(u)=x\}} 2^{-|u|},$$

one of terms of summation is $2^{-H_M(x)}$. So, $P_M(x) \leq 2^{-H_M(x)}$. ∎
This Lemma is often used in Chapter 5.

Lemma 4.21 *For every universal machine U and all strings x,*

$$0 < P_U(x) < 1. \tag{4.13}$$

Proof. For $0 < P_U(x)$, by Lemma 4.20 and x^*_U is surjective, $P_U(x) \leq 2^{-H_U(x)} = 2^{-|x^*_U|} > 0$. For $P_U(x) < 1$, by (4.11), $\sum_{x \in A^*} P_U(x) \leq 1$. And suppose that there is $x \in \Sigma^*$ such that $P_U(x) = 1$. Then, $\sum_{x \in A^*} P_U(x) = 1$. Hence, there exists $x \in \Sigma^*$ such that $P_U(x) = 0$. This contradicts the fact $P_U(x) > 0$. ∎

Proposition 4.22 *For every machine M and all naturals $m, n \leq 1$,*

$$\sharp\{x \in \Sigma^* | H_M(x) < m\} \leq 2^m - 1, \tag{4.14}$$

$$\sharp\{x \in \Sigma^* | P_M > \frac{n}{m}\} < \frac{m}{n}. \tag{4.15}$$

Proof. For (4.14), by lemma4.13,

$$\begin{aligned} \sharp\{x \in \Sigma^* \mid H_M(x) < m\} &= \sum_{0 \leq i \leq m-1} \sharp\{x \in \Sigma^* \mid H_M(x) = i\} \\ &\leq 2^0 + 2^1 + \cdots + 2^{m-1} \\ &= 2^m - 1. \end{aligned}$$

For (4.15), let $S = \{x \in \Sigma^* \mid P_M(x) > \frac{n}{m}\}$. Assume that $\sharp S \leq \frac{m}{n}$. By (4.11),

$$1 \geq \sum_{x \in \Sigma^*} P_M(x) \geq \sum_{x \in S} P_M(x).$$

By $P_M(x) > \frac{n}{m}$,

$$\sum_{x \in S} P_M(x) > \frac{n}{m} + \frac{n}{m} + \cdots = \frac{n}{m} \sharp S \geq 1.$$

So,

$$1 > \frac{n}{m} \sharp S \geq 1.$$

This is a contradiction. ∎

In next Chapter we establish upper bounds of program-size complexity by using the defined universal Chaitin machine and algorithmic probability. Consequently, we will show Algorithmic Coding Theorem.

Chapter 5

Recursively Enumerable Prefix Codes

In this Chapter, we present two main tools used to design Chaitin machines and consequently to establish upper bounds: the extension of the Kraft's classical condition in Theorem 3.6 to arbitrary r.e.sets and relativized complexities and probabilities. Latter on, we show that the complexities of the universal machine equals, within O(1), the halting entropy (Algorithmic Coding Theorem) [3].

5.1 Kraft-Chaitin Theorem

We devote this section to proof of the following important result.

Theorem 5.1 *Let $\varphi : N_+ \stackrel{o}{\to} N$ be a p.r.function having as domain an initial segment of N_+.The following statements are equivalent:*

(1) We can effectively construct an injective p.r.function:
$\theta : dom(\varphi) \to \Sigma^*$ such that:

(a) for every $n \in dom(\varphi), |\theta(n)| = \varphi(n)$,

(b) range(θ) is prefix-free.

(2) One has $\sum_{i \in N_+} 2^{\varphi(i)} \leq 1$.

An initial segment of N_+ is a finite set of the form $\{1, 2, 3, \cdots, n\}$ or N_+. If $i \notin dom(\varphi)$, then $\varphi(i) = \infty$. Hence, the term of rightside in (2) is $2^{-\infty} = 0$, so we can write $\sum_{i \in N_+} 2^{-\varphi(i)} = \sum_{x \in range(\theta)} 2^{-|x|}$.

Proof. For the direct implication, By Proposition3.7 and (1.b),

$$\sum_{i \in N_+} 2^{-\varphi(i)} = \sum_{x \in range(\theta)} 2^{-|x|} \leq 1.$$

For the converse implication, we will finish after Proposition5.16.
The first, we defines the injective p.r.function θ:

1. put $\theta(1) = a_1^{\varphi(i)}$
2. if $\theta(1)$, $\theta(2)$, $\cdots$, $\theta(n)$ have been constructed and $\phi(n+1) \neq \infty$, then put: $\theta(n+1) = \min\{x \in \Sigma^{\varphi(n+1)} | x \not<_p \theta(i) \text{ and } \theta(i) \not<_p x, \text{ for all } 1 \leq i \leq n\}$, where the minimum is taken according to the quasi-lexicographical order.

Intermediate Step. (1) *holds(i.e.there is a p.r.function* θ *such that* $(a), (b)$*) iff for each positive integer* n *for which* $\varphi(n+1) \neq \infty$,

$$\text{the set } B_n = \{x \in \Sigma^{\varphi(n+1)} | x \not<_p \ \theta(i) \text{ and } \theta(i) \not<_p \ x, \text{ for all } 1 \leq i \leq n\} \neq \emptyset.$$

For the direct implication, if $B_n = \emptyset$, then $\theta(n+1) = min\emptyset = \infty$. This contradicts $\varphi(n+1) \neq \infty$. For the converse, If $B_n \neq \emptyset$, there is a p.r.function θ. Hence, we will prove that *if* (2) *holds, then for each* n *for which* $\varphi(n+1) \neq \infty$, $B_n \neq \emptyset$.

By absurdity, assume that there exists some positive integer k, such that $B_k = \emptyset$. We distinguish two cases according to the relation between $\varphi(k+1)$ and $m = max\{\varphi(1), \cdots, \varphi(k)\}$:

I) $\varphi(k+1) \geq m$,

II) $\varphi(k+1) \leq m$.

In case) for all $x \in \Sigma^{\varphi(k+1)}$, there exists a positive integer i, $1 \leq i \leq k$, such that $\theta(i) <_p x$. Since the set $\{\theta(1), \theta(2), \cdots\}$ is prefix free, Σ^m can be written as the following disjoint union of sets:

$$\Sigma^m = \bigcup_{i=1}^{k} \{y \in \Sigma^m | \theta(i) <_p y\}.$$

Passing to cardinalities we get $2^m = \sum_{i=1}^{k} 2^{m-\varphi(i)}$; dividing by 2^m,

$$1 = \sum_{i=1}^{k} 2^{-\varphi(i)}.$$

By $\varphi(k+1) \neq \infty$,

$$\sum_{i=1}^{k+1} 2^{-\varphi(i)} > 1,$$

which is contradicts (2).
In case) we define the sets

$$C_1 = \{x \in \Sigma^{\varphi(k+1)} | \theta(i) <_p x, \text{ for some } 1 \leq \mathrm{i} \leq \mathrm{k}\},$$

$$C_2 = \{x \in \Sigma^{\varphi(k+1)} | x <_p \theta(i), x \neq \theta(i), \text{ for some } 1 \leq \mathrm{i} \leq \mathrm{k}\}.$$

Since $B_n \neq \emptyset$, $C_1 \cap C_2 = \emptyset$ and $C_1 \cup C_2 = \Sigma^{\varphi(k+1)}$. To be more specific, let $C_2 = \{x_1, x_2, \cdots, x_r\}$, according to the quasi-lexicographical order.

For every $1 \leq t \leq r$ put

$$P_t = \bigcup_{j=1}^{t} \{y \in \Sigma^* |\ |y| \leq m, x_j <_p y\}$$

and note that

$$\{\theta(i) | 1 \leq i \leq k, \varphi(i) > \varphi(k+1)\} \subset P_r \subset \bigcup_{t=\varphi(k+1)}^{m} \Sigma^t. \tag{5.1}$$

Indeed, if $\varphi(i) > \varphi(k+1)$, then take x to be the prefix of length $\varphi(k+1)$ of $\theta(i)$ and notice that $x \in C_2$.

Definition 5.2 a) An element $x \in \Sigma^*$ is called a *free-string* if $x \in P_r$ and there is not a natural i, $1 \leq i \leq k$ such that $\theta(i) <_p x$ or $x <_p \theta(i)$.
b) A free-string x is *minimal* if for every $y <_p x$, $y \neq x$, there exists a natural i, $1 \leq i \leq k$, such that $y <_p \theta(i)$.

Remark. Every proper prefix of a minimal free-string is not free.

Lemma 5.3 *For every h, $\varphi(k+1) \leq h \leq m$, and for every free-string x of length h, if $x <_p y$ and $|y| \leq m$, then y is a free-string.*

Proof. First, if x is a free-string, $x <_p y$ and $|y| \leq m$, then there exists $x_j \in C_2$ such that $x_j <_p x <_p y$. Hence, certainly $y \in P_r$. Second, suppose that y is not free. (i.e. there exists a natural i, $1 \leq i \leq k$, such that $\theta(i) <_p y$ or $x <_p \theta(i)$.) In case $\theta(i) <_p y$, by $x <_p y$, $\theta(i) <_p x$ or $x <_p \theta(i)$. In case $y <_p \theta(i)$, by $x \in P_r$, $x <_p y <_p \theta(i)$. They contradict the fact that x is free. ∎

Corollary 5.4 *Let $\theta(k+1) \leq h < t \leq m$ and assume that x is a free-string of length h. Then:*

$$\sharp\{y \in \Sigma^t | x <_p y, y \text{ is free }\} = 2^{t-h}, \tag{5.2}$$

Proof. Since x is a free string, by Lemma 5.3, all strings y such that $|y| \leq m$ and $x <_p y$ are free. ∎

Lemma 5.5 *For every free-string $x \in \Sigma^m \cap P_{r-1}$ there exists a unique minimal free-string $x' <_p x$ with $x' \in P_{r-1}$.*

Proof. (Unicity) Let $x' \neq x''$, $x' <_p x$, $x'' <_p x$ and $\{x', x''\} \subset P_{r-1}$. Suppose that both x', x'' are minimal free-strings and $x' \neq x''$. Since $x' <_p x$ and $x'' <_p x$, $x' <_p x'' or x'' <_p x'$. In case $x' <_p x''$, there exists a natural i, $1 \leq i \leq k$, such that $x' <_p \theta(i)$. So, x'' is a minimal free-string. This contradicts that x' is free. The same argument works for in case $x'' <_p x'$.
(Existence) If x is a minimal free-string, then we take as a minimal free-string x' with $x = x'$. If x is free and $x \in P_{r-1}$, there exists a $x_j \in C_2$ such that $x_j <_p x$, for some $1 \leq j \leq r-1$. Hence, The string x_j is not free. So, there exists an $i, 1 \leq i \leq k$ such that $x_j <_p \theta(i)$. And let x' be y such that $x_i <_p y <_p x$. ∎

Corollary 5.6 *Put* $M = \{x \in P_{r-1} \mid$ *x is a minimal free-string*$\}$. *Then*

$$\{y \in \Sigma^m \cap P_{r-1} \mid \text{ } y \text{ is a free-string}\} = \cup_{x \in M}\{y \in \Sigma^* \mid x <_p y\}, \tag{5.3}$$

and the union on the right-hand side is disjoint.

Proof. By lemma 5.5 and lemma 5.3, it is clear. ∎

Definition 5.7 For $x, y \in \Sigma^*$ we say that x is *on the left of y(we write:* $x \ll y$*)* in case $x = y$ or there exists $x' <_p x, y' <_p y$ such that $|x'| = |y'|$ and x' is less than y' according to the quasi-lexicographical order.

For example, $100 \ll 11$, but 100 and 10 are incomparable. For every natural $n > 0$, the relation $\ll$ is a total order on Σ^n. We write $x \ll y$ in case x is "left" of $y$in a complete tree with *lambda*.

Lemma 5.8 *For all* $x, y \in \Sigma^*$, *if* $x \ll y$, $x' <_p x$, $y' <_p y$ *and* $|x'| = |y'|$, *then* $x' \ll y'$.

Proof. Since the relation $\ll$ is total order on Σ^*, it is clear. ∎

Lemma 5.9 *For all* $x, y \in \Sigma^*$, *if* $x \ll y$, $x \neq y$, $x <_p x'$, $y <_p y'$, *then* $x' \ll y'$.

Proof. By a complete tree, it is clear. ∎

Proposition 5.10 *If* $x, y \in \Sigma^*$ *are minimal free-strings and* $x \ll y$, *then* $|x| \geq |y|$.

Proof. For direct implication, suppose that $|x| < |y|$. Take $x' <_p y$ with $|x'| = |x|$. Since y is a minimal string, there exist a natural i, $1 \leq i \leq k$, such that $x' <_p \theta(i)$. By Lemma 5.3(x is free), there exists a free-string $x"$ such that $x <_p x"$ and $|x"| = |\theta(i)|$. By $x \ll y$, $x <_p x$, $x' <_p y$, $|x| = |x'|$ and Lemma 5.8, $x \ll x'$. In case $x = x'$, the minimal free-string $x = x' <_p y$. It contradicts the fact that y is a minimal free-string. So $x \neq x'$. By $x \ll x'$, $x \neq x'$, $x' <_p \theta(i)$, and Lemma 5.9, we get $x" \ll \theta(i)$. It contradicts the construction of *theta*(i). Notice that θ is a minimum according to the quasi-lexicographical order, $x"$ is free and $|\theta(i)| = |x"|$. So, $x"$ must not be on the left of $\theta(i)$. ∎

Proposition 5.11 *Let* $x, y \in \Sigma^*$ *be minimal free-strings,* $x = x_0 a_i$, $y = y_0 a_j$, $a_i, a_j \in \Sigma$ *and* $x \ll y$. *Then*

$$x_0 = y_0 \text{ } \textit{iff} \text{ } |x| = |y|$$

Proof. For direct implication, if $x_0 = y_0$, then $|x| = |y|$.
For converse, assume that $x_0 \neq y_0$. By $x \ll y$, $x_0 <_p x$, $y_0 <_p y$, $|x_0| = |y_0|$ and lemma 5.8, then $x_0 \ll y_0$. Since the string y is minimal free, there exists a t, $1 \leq t \leq k$ such that $y_0 <_p \theta(t)$, $y_0 \neq \theta(t)$. Since $m \geq |\theta(t)| \geq y_0 + 1 = |x_0| + 1 = |x|$ and x is free, by lemma 5.3, there exists a free-string x' such that $x <_p x'$ and $|x'| = |\theta(t)| \leq m$. By $x_0 \ll y_0$, $x_0 <_p x <_p x'$, $y_0 <_p \theta(t)$, and lemma 5.9, then $x' \ll \theta(t)$. Since $|x'| = |\theta(t)|$ and x' is free, the fact that $x' \ll \theta(t)$ contradicts the construction of $\theta(t)$. ∎

Corollary 5.12 *For every* $\varphi(k+1) \leq h \leq m$,

$$\sharp\{x \in \Sigma^h \mid x \text{ is a minimal free-string}\} \leq 1. \tag{5.4}$$

Proof. All strings Σ^h are comparable with respect to $\ll$, so by Proposition 5.11, there exists a string $x_0 \in \Sigma^{h-1}$ such that

$$\{x \in \Sigma^h \mid x \text{ is a minimal free-string}\} \subset \{x \in \Sigma^h \mid x = x_0, a \in \Sigma\}.$$

So, $c_h = \sharp\{x \in \Sigma^h \mid x \text{ is a minimal free-string }\} \leq 2$. Moreover, since x is minimal, there exists a t, $1 \leq t \leq k$ such that $x_0 <_p \theta(t)$. In case $x_0 = \theta(t)$, then $\theta(t) <_p x$, which contradicts that x is free. So, $x_0 \neq \theta(t)$, $x_0 <_p \theta(t)$. Hence, there exists a $a' \in \Sigma$ such that $x_0 a' <_p \theta(t)$. The string $x_0 a'$ is not free. So, $c_h \leq 1$. ∎

Proposition 5.13 *Let $\varphi(k+1) \leq h \leq m$ and assume that x is a minimal free-string of length h with $x \ll \theta(i)$. Then $\varphi(i) \leq h - 1$.*

Proof. Assume that $\varphi(i) = |\theta(i)| \leq h$. Since x is free, by lemma 5.3, there exists a free-string x' such that $x <_p x'$, $|x| = \varphi(i) \leq h$. Since $x \ll \theta(i)$, $x \neq \theta(i)$, $x <_p x'$ and $\theta(i) <_p \theta(i)$, by lemma 5.9, hence $x' \ll \theta(i)$. Since $|\theta(i)| = |x'|$ and x' is free, so the fact that $x' \ll \theta(i)$ contradicts the construction of $\theta(t)$.) ∎

Proposition 5.14 *One has:*

$$\sharp\{x \in \Sigma^m \cap P_r | x \textit{ is a free-string}\} \leq 2^{m-\varphi(k+1)} - 1. \tag{5.5}$$

Proof. Put

$$Y_1 = \{x \in \Sigma^m \cap P_{r-1} \mid x \text{ is a free-string }\},$$

$$\begin{aligned} Y_2 &= \{x \in \Sigma^m \cap (P_r \setminus P_{r-1}) \mid x \text{ is a free string}\}, \\ &= \{x \in \Sigma^m \mid x \text{ is a free string and } x_r <_p x\}, \end{aligned}$$

and

$$Y = Y_1 \cup Y_2.$$

By Corollary 5.6,

$$Y_1 = \bigcup_{x \in \Sigma} \{y \in \Sigma^m \mid x <_p y\}, \text{ where } M = \{x \in P_{r-1} \mid x \text{ is a minimal free-string}\}.$$

By (5.1),

$$P_{r-1} \subset P_r \subset \bigcup_{j=\varphi(k+1)}^{m} \Sigma^j.$$

Let

$$t = \min\{|x| \mid x \text{ is a minimal free-string and } x \in P_{r-1}\} \leq m. \tag{5.6}$$

For every i, $0 \leq i \leq m - t$ put

$$k_i = \sharp\{x \in \Sigma^{m-i} \mid x \text{ is a minimal free-string}\}.$$

By Corollary 5.12, for every i, $k_i \leq 1$. Let

$$M_i = \{x \in \Sigma^i \cap P_{r-1} \mid x \text{ is a minimal free-string}\}.$$

By (5.3),

$$Y = \bigcup_{i=t}^{m} \bigcup_{x \in M_i} \{y \in Sigma^m \mid x <_p y\}$$

Since the union is disjoint and by Corollary 5.4,

$$\sharp Y = \sum_{i=t}^{m} k_{m-i} 2^{m-i} = \sum_{j=0}^{m-t} k_j 2^j.$$

Intermediate Step. If $x \in \Sigma^t \cup P_{r-1}$ is a minimal free-string, then there exists an i, $1 \leq i \leq k$, such that $\varphi(i) \leq |x| - 1 = t - 1$.
The string $x_r \in C_2$, so there is an i with $1 \leq i \leq k$, $x_r <_p \theta(i)$. If $x \in \Sigma^t capP_{r-1}$ is a minimal free-string, then there exists a $j \leq r-1$ such that $x_j <_p xwithx_j \in C_2$. By $x_j \ll x_r$, $x_j \neq x_r$, $x_j <_p x$, $x_r <_p \theta(i)$ and using lemma 5.9, so $x \ll \theta(i)$. By Proposition 5.13, $\varphi(i) = |\theta(i)| \leq |x| - 1 = t - 1$.
In view of the Intermediate Step,

$$\sharp\{x \in \Sigma^m \mid x_r <_p x, x \text{ is not free}\} \leq 2^{m-(t-1)},$$

because if $x \in \Sigma^m$ and $x_r <_p \theta(i) <_p x$, then x is not free.
Obviously,

$$\{x \in \Sigma^m \mid x_r <_p x\} = \{x \in \Sigma^m \mid x_r <_p x, x \text{ is a free-string}\} \cup \{x \in Sigma^m \mid x_r <_p x, x \text{ is not free}\},$$

so,

$$\begin{aligned} \sharp Y_2 &= \sharp\{x \in Sigma^m \mid x_r <_p x, x \text{ is a free-string }\} \\ &= 2^{m-\varphi(k+1)} - \sharp\{x \in \Sigma^m \mid x <_p x, x \text{ is not free }\} \\ &\leq 2^{m-\varphi(k+1)} - 2^{m-t+1}. \end{aligned}$$

Finally, by $\sharp Y_2 \leq 2^{m-\varphi(k+1)} - 2^{m-t-1}$ and (5.4),

$$\begin{aligned} \sharp Y &= \sharp Y_1 + \sharp Y_2 \\ &= \sum_{j=0}^{m-t} k_j 2^j + \sharp Y_2 \\ &\leq \sum_{j=0}^{m-t} k_j 2^j + 2^{m-\varphi(k+1)} - 2^{m-t+1} \\ &\leq \sum_{j=0}^{m-t} 2^j + 2^{m-\varphi(k+1)} - 1^{m-t+1} \\ &= (2^{m-t+1} - 1) + 2^{m-\varphi(k+1)} - 2^{m-\varphi(k+1)} - 2^{m-t+1} \\ &= 2^{m-\varphi(k+1)} - 1. \end{aligned}$$

∎

Corollary 5.15 *The following inequality is valid:*

$$\sharp\{x \in \Sigma^m \cap P_r | x \text{ is not free}\} \leq 2^m(1 - \sum_{\{i=1,\varphi(i)\leq\varphi(k+1)\}}^{k} 2^{-\varphi(i)}) - 2^{m-\varphi(k+1)} + 1. \quad (5.7)$$

Proof.

$$\begin{aligned}\Sigma^m \cap P_r &= \{x \in \Sigma^m \cap P_r \mid x \text{ is a free-string}\} \cup \{x \in \Sigma^m \cap P_r | x \text{ is not free}\}. \\ &= \bigcup_{j=1}^{r} \{x \in \Sigma^m \mid x_j <_p x\}\end{aligned}$$

But,

$$\begin{aligned}r &= \sharp C_2 \\ &= \sharp\{x \in \Sigma^{\varphi(k+1)} \mid x <_p \theta(i), x \neq \theta(i), \text{ for some } 1 \leq i \leq k\} \\ &= 2^{\varphi(k+1)} - \sharp\{x \in \Sigma^{\varphi(k+1)} \mid \theta(i) <_p x, \text{ for some } 1 \leq i \leq k\} \\ &= 2^{\varphi(k+1)} - \sharp \bigcup_{i=1}^{k} \{x \in \Sigma^{\varphi(k+1)} \mid \theta(i) <_p x\} \\ &= 2^{\varphi(k+1)} - \sum_{\{i\in N \mid \varphi(i)\leq\varphi(k+1), 1\leq i\leq k\}} 2^{\varphi(k+1)-\varphi(i)}.\end{aligned}$$

$$\begin{aligned}\sharp(\Sigma^m \cap P_r) &= r2^{m-\varphi(k+1)} \\ &= 2^m - \sum_{\{i\in N \mid \varphi(i)\leq\varphi(k+1), 1\leq i\leq k\}} 2^{m-\varphi(i)},\end{aligned}$$

by Proposition 5.5,

$$\begin{aligned}\sharp\{x \in \Sigma^m \cap P_r \mid x \text{ is not free}\} &= \sharp(\Sigma^m \cap P_r) - \sharp\{x \in \Sigma^m \cap P_r \mid x \text{ is a free string}\} \\ &\leq 2^m - \sum_{\{i\in N \mid \varphi(i)\leq\varphi(k+1), 1\leq i\leq k\}} 2^{m-\varphi(i)} - 2^{m-\varphi(k+1)} + 1.\end{aligned}$$

∎

Proposition 5.16 *The following equality is valid:*

$$\sum_{\{i=1,\varphi(i)\leq\varphi(k+1)\}}^{k} 2^{\varphi(i)} = \sharp\{x \in \Sigma^m \cap P_r | \ x \text{ is not free}\}2^{-m}. \quad (5.8)$$

Proof. First, we prove the equality:

$$\{x \in \Sigma^m cap P_r \mid x \text{ is not free } = \bigcup_{\{i\in N} \mid \varphi(i) \leq \varphi(k+1), 1 \leq i \leq k\}.$$

For the direct implication, if $x \in \Sigma^* cap P_r$, there exists $x_j \in C_2$ such that $|x_j| = \varphi(k+1)$ and $x_j <_p x$. Since x is a free and $|x| = m$, so there exists i_j such that $x_j <_p \theta(i_j)$,

$x_j \neq \theta(i_j)$. Hence $\theta(i_j) <_p x$ and $\varphi(k+1) < \varphi(i_j)$.
For the converse, if $x \in \Sigma^m$, $\theta(i) <_p x$, then there exists $|x'| \in \Sigma^{\varphi(k+1)}$ such that $x' <_p \theta(i) <_p x$ and $x' \neq \theta(i)$. So, $x \in P_r \cap \Sigma^m$) and x is not free.
Passing to cardinals we get (5.8).

Here we prove Theorem5.1.

Proof. Put

$$X_1 = \{i \in N \mid 1 \leq i \leq k, \varphi(i) \leq \varphi(k+1)\}$$
$$X_2 = \{i \in N \mid 1 \leq i \leq k, \varphi(i) > \varphi(k+1)\}.$$

$$\begin{aligned}
\sum_{i=1}^{k+1} 2^{\varphi(i)} &= \sum_{i=1}^{k} 2^{-\varphi(i)} + 2^{-\varphi(k+1)} \\
&= \sum_{i\in X_1} 2^{-\varphi(i)} + sum_{i\in X_2} 2^{-\varphi(i)} + 2^{-\varphi(k+1)} \\
by(5.8) &= \sum_{i\in X_1} 2^{-\varphi(i)} + \sharp\{x \in \Sigma^m \cap P_r \mid x \text{ is a free string}\}2^{-m} + 2^{-\varphi(k+1)} \\
by(5.7) &\leq \sum_{i\in X_1} 2^{-\varphi(i)} + 2^m(1 - \sum_{i\in X_1} 2^{-\varphi(i)}) - 2^{m-\varphi(k+1)} + 1)2^{-m} + 2^{-\varphi(k+1)} \\
&= 1 - 2^{-\varphi(k+1)} + 2^{-m} + 2^{-\varphi(k+1)} \\
&= 1 + 2^{-m} \\
&< 1,
\end{aligned}$$

which contradicts (2) in Theorem 5.1. ∎

Theorem 5.17 *Let $f : N_+ \stackrel{o}{\to} \Sigma^* \times N_+$ be a p.r. function whose domain is an initial segment of N_+. For every $k \in dom(f)$ put $f(k) = (x_k, n_k)$. If*

$$\sum_{k=1}^{\infty} 2^{-n_k} \leq 1,$$

then we can effectively construct a Chaitin machine C such that for every $k \in dom(f)$ there exists a string u_k of length n_k with $C(u_k) = x_k$. Furthermore, for every string v,

$$P_C(v) = \sum_{x_k = v} 2^{-n_k}, \tag{5.9}$$

and

$$H_C(v) = min\{n_k | x_k = v\}. \tag{5.10}$$

Proof. The p.r.function $\varphi : dom(f) \to N_+$ given by $\varphi(k) = n_k$ does satisfy the hypothesis of Theorem 5.1. So, We can define the machine M:

$$M(\theta(k)) = x_k, \ \textit{for every } k \in dom(k).$$

θ is from Theorem 5.1. Notice that θ is injective. For (5.9) and (5.10), the defined machine M is clearly satisfied.

∎

5.2 Algorithmic Coding Theorem

Theorem 5.18 *For every Chaitin machine M there exists a constant $c > 0$ (depending upon U and M) such that for all $x, y \in \Sigma$,*

$$H_U(x) \leq -\log P_M(x) + c. \tag{5.11}$$

Proof. The set

$$T = \{(x,n) \in \Sigma^* \times N \mid P_M(x) > 2^{-n}\}$$

$$= \{(x,n) \in \Sigma^* \times N \mid \sum_{i=1}^{m} 2^{-|y_i|} > 2^{-n}, \text{ for some } , \cdots, y_m \in dom(M)\}.$$

is r.e. Let $B = \{(x, n+1) \in \Sigma^* \times N \mid (x,n) \in T\}$ and $M = \sum_{(x,n+1)\in B} 2^{-(n+1)} = 2^{-1} \sum_{(x,n)\in T} 2^{-n}$. Here, We prove $M \leq 1$. To aim we first introduce the following.
For every real, if α, $2^n < \alpha \leq 2^{n+1}$ for some natural n, then put $n = lg\alpha$. The following relations hold true:

(1) if $\alpha > 0$, then $2^{lg\alpha} < \alpha$,

(2) if $\alpha > 0$, then $lg\alpha < \log\alpha \leq lg\alpha + 1$,

(3) if $\alpha > 0$ and m is an integer, then $lg\alpha \geq m$ iff $\log\alpha > m$.

For (1) and (2), by $n = lg\alpha$, which is clear. For the direct implication in (3), by $2^n < \alpha \leq 2^{n+1}$ and $lg\alpha \geq m$,

$$m \leq lg\alpha = n = \log 2^n < \log\alpha.$$

For the converse, by $2^n < \alpha \leq 2^{n+1}$ and $\log\alpha > m$,

$$m < \log\alpha \leq n + 1.$$

Hence $m < n + 1$. Since m, n is integers, so

$$m \leq n = \lg\alpha.$$

For every $x \in \Sigma^*$, we define the sets

$$N_x = \{n \in N \mid P_M(x) > 2^{-n}\}.$$

Since if $n \in N_x$, then $n + 1 \in N_x$, so N_x is infinite. Moreover,

$$M = 2^{-1} \sum_{\{n\in N_x \mid x\in\Sigma^*\}} 2^{-n},$$

and

$$\begin{aligned} n \in N_x &\iff P_M(x) > 2^{-n} \\ &\iff \log P_M(x) > -n \\ \text{by (3)}, &\iff lgP'_M x) \geq -n. \end{aligned}$$

So,

$$\begin{aligned}\sum_{n\in N_x} 2^{-n} &= \sum_{n\geq -l_M(x)} 2^{-n} \\ &= 2\cdot 2^{lgP_M(x)}, \\ \text{by (2)}, &< 2\cdot P_M(x).\end{aligned}$$

Finally,

$$M = \frac{1}{2}\sum_{x\in\Sigma^*}\sum_{n\in N_x} 2^{-n} \leq \sum_{x\in\Sigma^*} P_M(x) \leq 1.$$

Using the Kraft-Chaitin Theorem we construct a Machine $D : \Sigma^* \stackrel{o}{\to} \Sigma^*$ satisfying the following property: *For every $(x,n) \in T$ there exists a string $v \in \Sigma^*$ such that $D(v) = x$ and $|v| = n+1$.*
Notice that:

$$D(v) = x \Longleftrightarrow (x,|v|) \in B \Longleftrightarrow P_M(x) > 2^{1-|v|}.$$

and

$$\begin{aligned}H_D(x) &= \min\{|v| \mid v\in\Sigma^*, D(v) = x\} \\ &= \min\{|v| \mid v\in\Sigma^*, P_M(x) > 2^{1-|v|}\} \\ &= \min\{|x| \mid v\in\Sigma^*, |v| \geq 1 - lgP_M(x)\} \\ &= 1 - lgP_M(x).\end{aligned}$$

By Invariance Theorem,

$$H_U(x) \leq H_D(x) + c = -lgP_M(x) + 1 + c < -\log P_M(x) + 1 + c.$$

∎

Remark. Since $P_D(x) = \frac{1}{2}\sum_{n\in N_x} 2^{-n}$ *and* $\frac{1}{2}\sum_{n\in N_x} 2^{-n} < P_M(x)$ it follows that

$$P_D(x) < P_M(x).$$

Corollary 5.19 *For every machine M there exists a constant $c > 0$ (depending upon U, M) such that for all $x, y \in \Sigma^*$:*

$$P_U(x) \geq 2^{-c} P_M(x). \tag{5.12}$$

Proof. By formulae 5.11, there exists a constant $c > 0$ such that for all $x, y \in \Sigma^*$:

$$P_M(x) \leq 2^{c - H_U(x)}.$$

By Lemma4.20, we get

$$2^{-c} P_M(x) \leq 2^{-H_U(x)} \leq P_U(x).$$

∎

Theorem 5.20 *The following formulae are true:*

$$H_U(x) = -\log P_U(x) + O(1). \tag{5.13}$$

Proof. For $H_U(x) \leq -\log P_U(x) + O(1)$, we use Theorem 5.18. For $-\log P_U(x) \leq H_U(x) + O(1)$, by Lemma 4.20,

$$-\log P_U(x) \leq H_U(x).$$

∎

Remark. Actually, in Theorem 5.20, we can rewrite for every U there exists a constant $c \leq 0$ such that for all strings $x \in \Sigma^*$,

$$0 \leq H_U(x) + \log P_U(x) \leq 1 + c. \tag{5.14}$$

This result is means that program-size complexity of any universal machine are asymptotically optimal(i.e. optimal up to at most an additive, unknown constant) with respect to the machine's algorithmic probabilities. Now, we are interested in a class of machines ,not necessarily universal, and a class of any semi-distribution, not the machine's algorithmic probabilities, and a constant c.

Chapter 6

Coding with Minimal Programs

we investigate machines, not necessarily universal, satisfying Algorithmic Coding Theorem under condition of a given semi-distribution. Finally, we show the characterization of all machines satisfying the Algorithmic Coding Theorem.

6.1 The Condition Under the Semi-distribution

We investigate conditions under which given a semi-distribution P, we can find a machine M such that $H_M(x)$ is equal, up to an additive constant $c \leq 0$, to $\log P(x)$.

Definition 6.1 a)A *semi-distribution* is a function $P : \Sigma^* \to [0,1]$ such that

$$\sum_{x \in \Sigma^*} P(x) \leq 1.$$

In case $\sum_{x \in \Sigma^*} P(x) = 1$, P is called *distribution*.
b)Let Q be the set of rational number. A function is a *semi-computable from below(above)* if the set

$$\{(x,r) \in \Sigma^* \times Q | P(x) > r\}(\{(x,r) \in Q \times \Sigma^* | r < P(x)\}$$

is r.e. and a *computable* if the above set is computable. For example, $P_M(x)$ is semi-distribution semi-computable from below. The function $P(x) = 2^{-2|x|-3}$ is a computable semi-distribution.

Theorem 6.2 *Assume that P is a semi-distribution and there exist a r.e. set $S \subset \Sigma^* \times N$ and a constant $c \geq 0$ such that the following two conditions are satisfied for every $x \in \Sigma^*$:*

(1) $\sum_{(x,n) \in S} 2^{-n} \leq P(x)$,

(2) *for all $n \in N$, if $P(x) > 2^{-n}$, then there exists some $k \in S$ such that $(x,k) \in S$ and $k \leq n + c$.*

Then, there exists a machine M (depending upon S) such that for all $x \in \Sigma^*$,

$$-\log P(x) \leq H_M(x) \leq (1+c) - \log P(x). \tag{6.1}$$

Proof. In view of the condition (1),

$$\sum_{x\in\Sigma^*}\sum_{(x,n)\in\Sigma^*} 2^{-n} \leq \sum_{x\in\Sigma^*} P(x) \leq 1.$$

So, by the Kraft-Chaitin Theorem we can construct a machine M such that for every $(x,n)\in S$ there exists a string v with $|v|=n$ such that $M(v)=x$. If $(x,m)\notin S$, for all $m\in N$, then since $M(x)$ is undefined, $P(x)=0$ and $H_M(x)=\infty$, so the equation6.1 is satisfied. If $(x,m)\in S$, for some m, by conditions (2) and Theorem5.18 in Chapter4 we get:

$$\begin{aligned} H_M(x) &= \min\{|v| \mid v\in\Sigma^*, M(v)=x\} \\ &= \min\{m \mid m\in N, (x,m)\in S\} \\ &\leq \min\{n \mid n\in N, P(x)>2^{-n}\}+c \\ &= \min\{n \mid n\in N, n>-\log P(x)\}+c \\ &= \min\{n \mid n\in N, n\geq 1-lgP(x)\}+c \\ &\leq (1+c)-\log P(x). \end{aligned}$$

For $-\log P(x)\leq H_M(x)$, if $(x,n)\in S$, then by (1) $P(x)\leq 2^{-n}$, hence

$$\begin{aligned} H_M(x) &= \min\{|v| \mid v\in\Sigma^*, M)v=x\} \\ &= \min\{n \mid n\in N, (x,n)\in S\} \\ &= \min\{n \mid n\in N, P(x)\leq 2^{-n}\} \\ &= \min\{n \mid n\in N, n\leq -\log P(x)\} \\ &\geq -\log P(x). \end{aligned}$$

∎

Theorem 6.2 makes no direct computability assumptions on P

Lemma 6.3 *Let M be a machine such that $\Omega_M<1$. Then, there exist a universal machine U satisfying the equality $H_U(x)\leq H_M(x)$, for all x.*

Proof. By hypothesis, $\Omega_M<1$, so there exists a $k\in N$ such that Ω_M+2^{-k}. Let V be a universal machine.

$$\text{The set } S=\{(M(x),|x|) \mid x\in dom(M)\}\cup\{(V(x),|x|+k) \mid x\in dom(V)\}$$

is r.e. and by $0<\Omega<1$,

$$\sum_{(y,n)\in S} 2^{-n} \leq \Omega_M+2^{-k}\Omega_V \leq \Omega_M+2^{-k}\leq 1.$$

By using Kraft-Chaitin Theorem there exists a machine U such that for every $(y,n)\in S$ there exists a string $z\in dom(U)$ with $|z|=n$ such that $U(z)=y$. Clearly, for all $x\in\Sigma^*$,

$$H_U(x)\leq \min\{|w|+k \mid V(w)=x\}=H_V(x)+k,$$

and

$$H_U(x)=\min\{|v| \mid U(v)=x\}\leq H_M(x),$$

so U is universal and satisfies the required inequality. ∎

Lemma 6.4 *Let M be a machine. Then, there exists a machine M' such that $\Omega_{M'} < 1$ and $H_{M'}(x) = H_M(x) + 1$, for all $x \in \Sigma^*$.*

Proof. Let the set:

$$S = \{(M(v), |v| + 1) \mid M(v) = x\}.$$

The set S is r.e. and

$$\sum_{(M(v),|v|+1)\in S} 2^{-(|v|+1)} = \Omega_M \cdot 2^{-1} < 1.$$

So, by using Kraft-Chaitin theorem, there exists a machine M' such that for every $(y, n) \in S$ there exists a string z with $|z| = n = |v| + 1$ such that $M'(z) = y$.
So, for all $x \in \Sigma^*$

$$\begin{aligned} H_{M'}(y) &= \min\{|z| \mid M'(z) = y, |z| = |x| + 1, x \in dom(M)\} \\ &= \min\{|x| \mid x \in dom(M)\} + 1 \\ &= H_M(y) + 1. \end{aligned}$$

$$\begin{aligned} \Omega_{M'} &= \sum_{z\in dom(M')} 2^{-|z|} = \sum_{x\in dom(M)} 2^{-(|x|+1)} = 2^{-1} \sum_{x\in dom(M)} 2^{-|x|} \\ &< \sum_{x\in dom(M)} 2^{-|x|} \leq 1. \end{aligned}$$

∎

Corollary 6.5 *Under the hypotheses of Theorem 6.2, a universal machine U can be constructed such that for all string x,*

$$H_U(x) \leq (2 + c) - \log P(x). \tag{6.2}$$

Proof. By Lemma 6.4, Lemma 6.3 and hypotheses of Theorem 6.2 , clearly

$$H_U(x) \leq H_{M'}(x) = H_M(x) + 1 \leq -\log P(x) + (2 + c), \text{ for all string } x.$$

∎

6.2 Minimal Programs are Optimal

Now, let $C_M = x^*_M$.

Proposition 6.6 *Assume that P is a semi-distribution semi-computable from below. Then, there exists a machine M (depending upon P) such that for all string x,*

$$-\log P(x) \leq H_M(x) \leq 2 - \log P(x). \tag{6.3}$$

Consequently, minimal programs for M are almost optimal: the code C_M satisfies the inequalities:

$$0 \leq L_{C_M,P} - \mathcal{H}_P \leq 2.$$

Proof. Let the set $S = \{(x, n+1) \mid P(x) > 2^{-n}\}$. For all strings x,

$$\sum_{(x,n)\in S} 2^{-n} = \sum_{n>1-\log P(x)} 2^{-n} = 2^{lgP(x)} < P(x),$$

so condition (1) in Theorem 6.2 is satisfied. Condition (2) holds for $c = 1$. Hence by (1),

$$0 \leq L_{C_M,P} - \mathcal{H}_P = \sum_{x\in\Sigma^*} P(x) \cdot (H_M(x) + \log P(x)) \leq 2.$$

∎

Corollary 6.7 *Assume that $f : \Sigma^* \to N$ is a function such that the set $\{(x,n)|f(x) < n\}$ is r.e. and $\sum_x 2^{-f(x)} \leq 1$. Let $P(x) = 2^{-f(x)}$. Then P is a semi- distribution semi-computable from below, and there exists a machine M (depending upon f) such that for all x,*

$$H_M(x) \leq 1 + f(x). \tag{6.4}$$

Minimal Programs M are almost optimal: the code C_M satisfies the inequalities:

$$0 \leq L_{C_M,P} - \mathcal{H}_P \leq 1.$$

One more bit is enough to guarantee universality of the constructed machine, that is, there exists a universal machine U (depending upon f)such that the code C_U satisfies the inequalities:

$$0 \leq L_{C_U,P} - \mathcal{H}_P \leq 2.$$

Proof. Let the set $S = \{(x,n) \mid n > f(x)\}$. Clearly, $S = \{(x,n) \mid P(x) > 2^{-n}\}$. The first condition in Theorem 6.2 is satisfied as $\sum_{n>f(x)} 2^{-n} = P(x) \leq 1$, for every x, and the second condition is satisfied for $c = 0$. ∎

Remark. When the semi-distribution P is given, an optimal prefix-code can be found for P. However, that code may be far from optimal for a different semi-distribution. For example let C be a prefix-code such that $|C(x)| = 2^{|x|+2}$, for all x. Let $\alpha > 0$ and consider the distribution

$$P_\alpha(x) = (1 - 2^{-\alpha})2^{-(\alpha+1)|x|}.$$

Two radically different situations appear: if $\alpha \leq 1$, then

$$L_{C,P_\alpha} - \mathcal{H}_{P_\alpha} = \infty,$$

but if $\alpha > 1$, then

$$L_{C,P_\alpha} - \mathcal{H}_{P_\alpha} < \infty,$$

So, C is asymptotically optimal for every distribution P_α with $1 < \alpha$, but C is far away from optimality if $0 < \alpha \leq 1$. Note that P_α is computable provided α is computable.

The next result shows that minimal programs are asymptotical optimal for every semi-distribution semi-computable from below.

Theorem 6.8 *Let P is a semi-distribution semi-computable from below, and U a universal machine. Then, there exists a constant c_P (depending upon P) such that*

$$0 \leq L_{C_U,P} - \mathcal{H}_P \leq 1 + c_P.$$

Proof. Let M be the machine constructed in Proposition 6.6 and let c_M be the simulation constant of M on U. Then,

$$0 \leq L_{C_U,P} - \mathcal{H}_P \leq L_{C_U,P} + c_M - \mathcal{H}_P \leq 1 + c_M.$$

so take $c_p = c_M$. ∎

Proposition 6.9 *If P is a computable semi-distribution. Then there exists a machine M such that*

$$-\log P(x) \leq H_M(x) \leq 1 - \log P(x).$$

Proof. Note that $-lgP(x) = \min\{n \mid n \in N, P(x) > 2^{-n}\}$ and then apply Theorem 6.2 to the set $S = \{(x, -lgP(x)) \mid x \in \Sigma^*\}$, by $-lgP(x) \leq P(x)$, (1) is satisfied and by $-lgP(x) \leq n$, (2) is satisfied with a constant $c = 0$. ∎

Corollary 6.10 *Let P be a computable semi-distribution. Then, there exists a universal machine U such that*

$$H_U(x) \leq 1 - \log P(x).$$

6.3 Algorithmic Coding Theorem Revisited

We characterize all machines satisfying the Algorithmic Coding Theorem and we construct a class of (universal) machines for which the inequality is satisfied with constant $c = 0$.

Proposition 6.11 *Let M be a machine and $c \geq 0$. The following statements are equivalent:*

(a) *for all x, $H_M(x) \leq (1 + c) - \log P_M(x)$,*

(b) *for all natural $n \geq 0$, if $P_M(x) > 2^{-n}$, then $H_M(x) \leq n + c$.*

Proof. For (a) implies (b), by $H_M(x) \leq (1 + c) - \log P_M(x)$ and $P_M(x) > 2^{-n}$,

$$2^{-n} < P_M(x) \leq 2^{(1+c)-H_M}.$$

For (b) implies (b), since for all $n \in N$, $P_M(x) > 2^{-n}$,

$$\begin{aligned} H_M(x) - c &\leq \min\{n \mid n \in N, P_M(x) > 2^{-n}\} \\ &= \min\{n \mid n \in N, n > \log P_M(x)\} \\ &= \min\{n \mid n \in N, n \leq 1 + lgP_M(x)\} \\ &= 1 + lgP_M(x) \\ &\leq 1 + \log P_M(x) \end{aligned}$$

Remark. For every machine M satisfying one of the equivalent conditions in Proposition 6.11, the Algorithmic Coding theorem follows (see Theorem 5.20):

$$|H_M(x) + \log P_M(x)| \leq 1 + c. \tag{6.5}$$

In fact, a machine M satisfies (6.5) if and only if condition (b) is satisfied. Every universal machine U satisfies (a) and (b) (see Theorem 5.18), but not all machines satisfy this condition. For example, in case considering enumeration of $2^{|x|}$ copies of pair $(x, 3|x|+1)$, using Kraft-Chaitin Theorem, construct machine M such that for every string x there exist $2^{|x|}$ different strings u_x^i such that

$$M(u_x^i) = x, i = 1, 2, 3 \cdots 2^x.$$

It seen that $P_M(x) = \sum_{M(u)=x} 2^{-u} = 2^{-2|x|-1}$. So, taking $n_x = 2|x| + 2$ we get $P_M(x) > 2^{-n_x}$,

$$H_M(x) = 3|x| + 1 \leq 2|x| + 2 + c.$$

So there is not constant c.

Next, incase $c = 0$, for example, for every x, consider enumeration of 3 of pair $(x, 3|x|), (x, 3|x| + 1), (x, 3|x| + 3)$. Use Kraft-Chaitin Theorem to construct a machine M such that for every string x there exist 3 different strings u_x^i such that $M(u_x^i) = x, i = 1, 2, 3$. It seen that $P_M(x) = \sum_{M(u)=x} 2^{-u} = 2^{-3|x|+0.\cdots}$. So, taking $n_x = 3|x|$ we get $P_M(x) > 2^{-n_x}$, so for all x,

$$H_M(x) = 3|x| \leq 3|x|.$$

This is holds for $c = 0$.

Some machines satisfy condition (b) with $c = 0$, so their minimal programs begin almost optimal. Hence next in Proposition 6.12 we will provide a class of (universal) machines satisfying the condition.

Proposition 6.12 *Let M be a machine such that if for all strings $u \neq u'$ with $M(u) = M(u')$ implies $|u| \neq |u'|$. Then, for all strings x,*

$$H_M(x) \leq 1 - \log P_M(x). \tag{6.6}$$

Proof. Let the set $S = \{(x, |u|) \mid M(u) = x\}$, and notice that

$$P_M(x) = \sum_{(x,k)\in S} 2^{-k},$$

as programs producing the same output x have different lengths. So, $|u|$ of the set S is different natural numbers. In view of the hypothesis,

$$\text{For all } n \in N,\ P_M(x) = \sum_{(x,k)\in S} 2^{-k} > 2^{-n}$$

$$\Leftrightarrow \exists (x,k_1) \in S \left[(k_1 < n) \vee [(k_1 = n) \wedge \exists k_2 (k_2 \neq k_1 \wedge (x,k_2) \in S)] \right],$$

To prove above, we will show that for all $n \in N$, $P_M(x) = \sum_{(x,k)\in S} 2^{-k} > 2^{-n}$ and not $\exists (x,k_1) \in S$ such that $(k_1 < n)$ imply $\exists (x,k_1) \in S$ such that $(k_1 = n) \wedge \exists k_2 (k_2 \neq k_1 \wedge (x,k_2) \in S)$. For the direct implication, if for all $n \in N$, $x \in \Sigma^*$, $P_M(x) > 2^{-n}$, then there exists a $(x,k) \in S$ such that $2^{-k} > 2^{-n}$ or $2^{-k} \leq 2^{-n}$. In case there is not a $(x,k) \in S$ such that $2^{-k} > 2^{-n}$, for all $(x,k) \in S$, $n \in N$, $2^{-k} \leq 2^{-n}$. Hence, for all $(x,k) \in S$, $n \in N$, $2^{-k} = 2^{-n}$ or $2^{-k} < 2^{-n}$. Here, assume that for all $(x,k) \in S$, $n \in N$, $2^{-k} \neq 2^{-n}$. Then, for all $(x,k) \in S$, $n \in N$, $2^{-k} < 2^{-n}$ i.e. for all $(x,k) \in S$, $n \in N$, $k > n$. Since each k is different natural numbers, $\sum_{(x,k)\in S} 2^{-k} \leq \sum_{k=n+1}^{\infty} 2^{-k} \leq 2^{-n}$. This contradicts that $\sum_{(x,k)\in S} 2^{-k} > 2^{-n}$. Hence, there exists a $(x,k_1) \in S$ such that $k_1 = n$. Moreover, since one have to satisfy $\sum_{(x,k)\in S} 2^{-k} > 2^{-n}$ there exists k_2 such that $(x,k_2) \in S$ and by machine's definition, $k_2 \neq k_1$.

For the converse implication, in case there exists a $(x,k_1) \in S$ such that $k_1 < n$, clearly $\sum_{(x,k)\in S} 2^{-k} > 2^{-n}$. In case for some $(x,k_1) \in S$, $k_1 = n$ and for some $(x,k_2) \in S$, $k_2 \neq k_1$, clearly $\sum_{(x,k)\in S} 2^{-k} \geq 2^{-k_1} + 2^{-k_2} > 2^{-k_1} = 2^{-n}$. Hence, the condition (2) in Theorem 6.2 is satisfies with $c = 0$. Using Theorem 6.2 we deduce the existence of a machine M' such that $H_{M'}(x) \leq 1 - \log P_{M'}(x)$, for all strings x. ∎

Chapter 7

Concluding Remarks

In Chapter 3 we gave definition of the prefix-free sets, prefix codes and Shannon entropy, and showed the mainly property of the prefix code. Propaties are that the prefix-code is the uniqueness of decodability, that the set of the prefix code-strings (prefix-free set) satisfy Kraft's inequality and that the average lengths of the prefix code-strings is about equal to the shannon' entropy. In Chapter 4 we gave the definition of the program-size complexities using Chaitin machine and the machine's algorithmic probability and halting probability. We showed the Invariant Theorem, non-computability of program-size complexity and computability of approximations to the program-size complexity, and derive some elementary estimation for complexities. In Chapter 5 we showed the extention to an arbitrary recursively enumerable set the classical Kraft's inequality condition (Kraft-Chaitin Theorem). We showed the Algorithmic Coding Theorem which is the important result in Algorithmic Information Theory; $-\log P_U(x) \leq H_M(x) \leq c - \log P_U(x)$. The stronger version of the result was shown in Chapter 6. We give the basic result.
When P is a semi-distribution and $S \subset \Sigma^* \times N$ and a constant $c \geq 0$ such that the following two conditions are satisfied for every $x \in \Sigma^*$:

(1) $\sum_{(x,n)\in S} 2^{-n} \leq P(x)$,

(2) *for all* $n \in N$, *if* $P(x) > 2^{-n}$, *then there exists some* $k \leq n+c$ *such that* $(x,k) \in S$.

Then,there exists a machine M (depending upon S) such that for all $x \in \Sigma^*$, $-\log P(x) \leq H_M(x) \leq (1+c) - \log P(x)$.

By using this result, we investigated under variable conditions. When P is a semi-distribution semi-computable from below. Then, there exists a machine M such that for all string x,

$$-\log P(x) \leq H_M(x) \leq 2 - \log P(x).$$

When P is a computable semi-distribution. Then there exists a machine M such that for all string x,

$$-\log P(x) \leq H_M(x) \leq 1 - \log P(x).$$

Finally, a class of the machines satisfying the Algorithmic Coding Theorem satisfy the follow condition that there exists constant $c \geq 0$ for all natural n, if $P_M(x) > 2^{-n}$, then $H_M(x) \leq n + c$. Furthermore, a class of the machines satisfying the Algorithmic

Coding Theorem with $c = 0$ was under condition that for all different programs $u \neq u'$, $M(u) = M(u')$ implies $|u| \neq |u'|$.

Bibliography

[1] Cristian.S.Calude, *Information and Randomness :An Algorithmic Perspective*, Springer-Verlag, 1994.

[2] Cristian.S.Calude, Hajime Ishihara and Takeshi Yamaguchi, *Minimal programs are almost optimal*, CDMTCS Reserch Report Series, CDMTCS-116, 1999, Spring.

[3] Greg.J.Chaitin, *Information, Randomness and Imcompleteness, Paper on Algorithmic Information Theory*, World Scientific, Singapore, 1990(2nd ed.,).

[4] Greg.J.Chaitin, *Algorithmic Information Theory*, Cambridge University Press, Cambridge, 1987.

[5] Ming Li and Paul Vitanyi, *Kolmogorov Complexity and Its Applications*, Springer, New York, 1997(2nd ed.,).

[6] T.M.Cover, J.A.Thomas, *Element of Informatin Theory*, Jhon Wiley, New York, 1991.

[7] Martin D.Davis and Ron Sigal, *Computability, Complexity, and Languages*, Jhon Wiley, New York, 1991.

[8] Robert I.Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, Berlin Heidelberg, 1987.

[9] Osamu Watanabe and Naoki Yonezaki, *Keisanron Nyuumonn*, Nihon-Hyouronsya, 1997 (in Japanese).

[10] K.Matsuzaka, *Shugo Iso Nyumon*, Iwanami-shoten, 1993 (in Japanese).