Kolmogorov complexity toolkit

All the definitions that follow are directly taken from the book [LV19]. We specify on which page the definition is given with brackets, for example we write {12} for page 12. If there is a special numbering (i.e. if the statement is in a Theorem), we might write $\{12, T2.3\}$ to refer to Theorem 2.3 of the book. Same goes for Lemmas (L), Definitions (D), Properties (P), Exercises (Ec) and Examples (Ep).

1 Basic and very useful facts

1.1 Equivalence between integers and binary sequences

{7} $\mathcal{N}$ denotes the integers (including 0).

Definition *S1.4* - $\{12, 13\}$*: (Enumeration of binary sequences).* We enumerate the binary sequences as follows

$$(0, \epsilon), (1, 0), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), \dots \tag{1}$$

We will make the confusion between a binary string and its position in the lexicographical ordering, i.e. we make the confusion between $\mathcal{N}$ and $\{0, 1\}^*$. For $x, y \in \mathcal{N}$, we denote by xy the concatenation of x and y, and x^R denotes the flipped version of $x \in \mathcal{N}$. We write $l(x)$ to denote the length of $x \in \mathcal{N}$.

1.2 Pairing functions

Definition *S1.2* - {7}*: (Definition Pairing function).* A pairing function $\langle \cdot \rangle : \mathcal{N}^2 \to \mathcal{N}$ is a total one-to-one function. A pairing function can always be extended to arbitrary $\mathcal{N}^n$ according to the following recursive definition.

$$\langle x_1, \dots, x_n \rangle := \langle x_1, \langle x_2, \dots, x_n \rangle \rangle. \tag{2}$$

Definition *S1.2* - {7}*: (Definition Cantor pairing function).* A common example of a pairing function is the Cantor pairing function defined as

$$\langle x, y \rangle_C := y + (x + y + 1)(x + y)/2. \tag{3}$$

Lemma *E1.7.13* - {43}*: (Cantor pairing function is bijective).* The Cantor pairing function has the advantage to be bijective, but it is not a prefix-code. Moreover, for Kolmogorov complexity, it is not very intuitive, and we rather use the following standard pairing function.

Definition *E1.4.5* - {15}*: (Standard pairing function).* We define the standard pairing function as

$$\langle y, x \rangle = \bar{y}x, \ \forall x, y \in \mathcal{N}, \tag{4}$$

$$\langle x_1, x_2, \dots, x_n \rangle = \bar{x}_1 \bar{x}_1 \dots \bar{x}_{n-1} x_n, \quad \forall x_1, \dots, x_n \in \mathcal{N}. \tag{5}$$

Definition *S1.7.3* - {35 – 36}*: (Standard pairing function).* The following notations are not explicitly mentionned in the book, but appear implicitly. Let $x \in \mathcal{N}$, and $y \in \mathcal{Q}$, and denote by $p/q = y$ the irreducible form of y. $x = y$ means $x = \langle p, q \rangle$. (x, y) denotes $\langle x, \langle p, q \rangle \rangle$, and (y, x) denotes $\langle \langle p, q \rangle, x \rangle$.

1.3 Asymptotic notation

{16} We write $f(x) = O(1)$ if $\exists c > 0$ such that $f(x) \leq c$ for all $x \in \mathcal{N}$, and we define $f(x) \leq O(1)$ analogously.

2 Prerequisites in probability and information theory

2.1 Probability theory

{19} A measure is a function defined on a field to $\mathcal{R}_+$.

Definition *E1.6.6* - {23}*: (Measure on integers).* We introduce a uniform measure on $\mathcal{N}$, as

$$L(x) = 2^{-2l(x)-1}. \tag{6}$$

One has

$$L_n(x) := L(x|l(x) = n) = \frac{1}{2^n} 1_{l(x)=n}. \tag{7}$$

Definition *E1.6.5* - {21}*: (Cantor space).* Let $\Gamma_x := \{x\omega : \omega \in \{0, 1\}^\infty\}$. The sets $\Gamma_x,\ x \in \mathcal{N}$ are called the *cylinder sets*. Note that $\Gamma_x = \Gamma_{x0} \bigsqcup \Gamma_{x1}$, where $\bigsqcup$ denotes disjoint union. The closure $\mathcal{F}$ of cylinder sets under union, intersection and difference is an algebra (or field).

Definition *E1.6.5* - {21}*: (Uniform distribution on the Cantor space).* We define the Lebesgue measure on $\mathcal{F}$ as $\lambda(\Gamma_x) = 2^{-l(x)}$, and by respecting {18} the Kolmogorov axioms of probability field for the rest. By {21} the Kolmogorov extension theorem, we can define a smallest σ-algebra σ containing the cylinder sets, associated with a probability measure on σ, which agrees with the Lebesgue measure on $\mathcal{F}$.

Definition *E1.7.8* - {36}*: (Computable measure).* A measure is said to be computable (upper, lower semicomputable) if the function $f : \mathcal{N} \to \mathcal{R}$ defined by $f(x) = \mu(\Gamma_x)$ for all $x \in \mathcal{N}$ is computable (upper, lower semicomputable).

2.2 Entropy, Information Theory

{67} The entropy of random variable X defined on $\Omega \to \mathcal{X}$ is

$$H(X) = \sum_{x \in \mathcal{X}} P(X = x) \log \frac{1}{P(X = x)} \tag{8}$$

{68} Length of Shannon-Fano code tends to entropy.

{69} We define the information given by $X = x$ to Y as $I(X = x : Y) = H(Y) - H(Y|X = x)$.

{70} We define the mutual information as $I(X : Y) = \mathbb{E}(I(X = x : Y)) = \mathbb{E}(I(Y = y : X))$.

{72} We define the Kullback-Leiber divergence as

$$D(q_1||q_2) := \sum_{x \in \mathcal{X}} q_1(x) \log \frac{q_1(x)}{q_2(x)}. \tag{9}$$

2.3 Statistics

{83} A satistic is a function S from a countable set $\mathcal{X}$ (data points) to a set $\mathcal{S}$ (satistic). Let $\{P_\theta : \theta \in \Theta\}$ be a class of distributions, indexed in a class Θ of parameters. A statistic is sufficient if $p_\theta(x|S(x) = s)$, does not depend on θ.
{84} A statistic is sufficient $I(\Theta, X) = I(\Theta, S(X))$ for all any prior distribution on Θ.

3 Codes

3.1 Prefix-codes, uniquely decodable codes and self-delimiting codes

Definition *S1.4* - {13}*: (Prefix).* We say that $x \in \mathcal{N}$ *is a prefix of* $y \in \mathcal{N}$ if there exists $z \in \mathcal{N}$ such that $y = xz$.

Definition *S1.4* - {13}*: (Prefix-free set).* $A \subseteq \mathcal{N}$ is said to be prefix-free if no element in A is the prefix of another element in A.

Definition *S1.4* - {13}*: (Code, prefix code).* Any function $D : \{0,1\}^* \to \mathcal{N}$ is called a *decoding function*. $\mathrm{dom}(D)$ is called the set of *code words*, and $\mathrm{range}(D)$ is called the set of *source words*. D is a prefix-code if $\mathrm{dom}(D)$ is prefix-free. $E := D^{-1}$ is aid to be the encoding relation (or encoding function in the case that it is a function).

Definition *S1.4* - {13}*: (Self-delimiting code).* We define the self-delimiting encoding as

$$\bar{x} = 1^{l(x)}0x, \ \forall x \in \mathcal{N}. \tag{10}$$

Note that $l(\bar{x}) = 2l(x) + 1$. $\bar{x}$ is also called the *self-delimiting version of* x. Furhter note that $\{\bar{x} : x \in \mathcal{N}\}$ is a prefix-free set.

Definition *S1.11.1* - {74}*: (Extending codes).* {74} E is a relation on $\mathcal{N} \times \{0,1\}^*$, that we can naturally extend to $\mathcal{N}^* \times \{0,1\}^*$, by considering $E(\varepsilon = \varepsilon)$.

Definition *S1.11.1* - {74}*: (Uniquely decodable code).* We say that E is uniquely decodable, if for all source sequence $y \in \mathcal{N}^*$, the code words x satisfying $E(x) = y$ are different than for any other source sequence y'.

***Lemma** S1.11.1 - {75}: (Prefix codes are decodable).* Let D be a prefix-code. Then, $E = D^{-1}$ is uniquely decodable.

***Theorem** T1.11.1 - {76}: (Kraft inequality).* Let $l_1, l_2, \ldots$ be a finite or infinite sequence of natural numbers. There is a prefix-code (or a uniquely decodable code) $x_1, x_2, \ldots$ satisfying $l(x_n) = l_n$ iff

$$\sum_n 2^{-l_n} \leq 1. \tag{11}$$

***Lemma** S1.11.4 - {81}: (Shortest prefix-code).* $l^*(x) := \log x + \log\log x + \log\log\log x + \ldots$. Then, by setting $c := \sum_x 2^{-l^*(x)}$, one can define a prefix code of lengths $l_x := l^*(x) + \log c$.

***Definition** D1.11.4 - {79}: (Self-delimiting codes).* A prefix-code is called *self-delimiting* if there is a Turing machine that decides whether a given word is a code word, never reading beyong the word it self, and moreover computes the decoding function. The *minimum description length* is defined on the same page.

3.2 Optimal and universal codes

***Definition** S1.11.3 - {77}: (Complete code).* A uniquely decodable code is complete, if by adding one element to the code, the code is not uniquely decodable anymore.

***Theorem** S1.11.3 - {77}: (Characterization of complete codes).* A code is complete and uniquely decodable iff there is equality in the Kraft inequality.

***Definition** D1.11.3 - {77}: (Average code length).* Let D be an injective prefix-code, and define P to be a probability distribution on $\mathcal{N}$. We define the *average code length associated with* D as $L_{P,D} := \sum_{x\in\mathcal{N}} P(x)l(D^{-1}(x))$.

***Theorem** T1.11.2 - {78}: (Noiseless coding theorem).* Let P be a distribution on $\mathcal{N}$, and define the *minimal average code length*

$$L(P) = \min\left\{L_{P,D} : D \text{ is an injective prefix-code}\right\}. \tag{12}$$

Then, by letting $H(P) := \sum_{x\ in\mathcal{N}} P(x)\log 1/P(x)$, one has

$$H(P) \leq L \leq H(P) + 1. \tag{13}$$

***Definition** D1.11.5 - {80}: (Universal code).* {81} Let C be an infinite set of uniquely decodable words, and $\mathcal{N}$ be a set of source words with distribution P. C is *universal* if there exists $c > 0$ independant of P, s.t.

$$\frac{\sum_{x\in\mathcal{N}} P(x)l(C^{-1}(x))}{\max\{H(P), 1\}} \leq c. \tag{14}$$

A universal code C is said to be asymptotically optimal if there is a function f satisfying $\lim_{t\to\infty} f(t) = 1$ such than

$$\frac{\sum_{x\in\mathcal{N}} P(x)l(C^{-1}(x))}{\max\{H(P),1\}} \leq f(H(P)). \tag{15}$$

{81} $\{\bar{x} : x \in \mathcal{N}\}$ is universal but not asymptotically optimal. However, $\{\overline{l(x)}x : x \in \mathcal{N}\}$ is both universal and asymptotically optimal.

4 Computability theory

4.1 Turing machines

The definition and behavior of Turing machines is given on pages $\{24-28\}$. We here do not review it, but however we put an emphasis on the definition of inputs and outputs, which is crucial for the consistency of the theory.

Definition *S1.7.1* - {28}*: (I/O Turing Machines).* We say that $(x_1,\ldots,x_n) \in \mathcal{N}^n$ is *presented as an input* to a Turing machine M, if at time step 0, $\bar{x} = \bar{x}_1\bar{x}_2\ldots\bar{x}_n$ is written on the tape, and the first symbol (leftmost symbol) of $\bar{x}$ is scanned by the head of the Turing machine. Furthermore, if the machine M halts during a given computation, the integer represented by the **maximal binary string (bordered by blanks) of which some bit is scanned** by the time the machine halts is called the *output* of the computation.

Definition *E1.7.3* - {30}*: (Encoding of Turing machines).* For each Turing machine $T : Q \times A \to S \times Q$, we associate an encoding. We define first $s = |Q \cup S| = \lceil \log(d(Q)+5) \rceil$, and the encoding $e : Q \cup S \to \{0,1\}^s$. Let r be the number of rules in the transition table. Note that $r \leq 3d(Q)$. Then,

$$E(T) = \bar{s}\bar{r}e(p_1)e(t_1)e(s_1)e(q_1)\ldots e(p_r)e(t_r)e(s_r)e(q_r). \tag{16}$$

Note that $l(E(T)) \leq 4rs + 2logrs + 4$, and $E(T)$ is self-delimiting.

Definition *E1.7.3* - {30}*: (Gödel indexing of Turing machines).* {30} By ordering the $E(T)$ with lexicographical ordering with increasing length, one defines the Gödel indexing $n(T)$ of each Turing machine. Accordingly, the Turing machines are now denoted $T_1, T_2, \ldots$. {30} One can then define a universal Turing machine taking as input $1^i 0p$ and outputting $T_i(p)$.

$\{90-92\}$ Read those pages for the note on the state-symbol product.

4.2 Oracle Turing machines

Definition *D1.7.7* - {38}*: (Oracle Turing machine).* Let T be a Turing machine, and $A \subseteq \mathcal{N}$. T^A is said to be a Turing machine with oracle A, if at any time step of a computation, the machine can ask if the word written on its tape belongs to A, and make a decision about it.

Theorem *S1.7.4* - {39}*: (Polynomial time reducibility).* We write $A \leq_T^P B$ if there exists T an oracle Turing machine such that T^B accepts A in polynomial time.

4.3 Computable functions

$\{28-29\}$ Let $A \subseteq \mathcal{N}^n$ and $f : A \to \mathcal{N}$. We generally say that f is a *partial function.* In the case that $A = \mathcal{N}^n$, we say that f is *total.* A is called the *domain* of f. We usually write $f^{(n)}$ if f has n variables. {29} A partial function with range $\{0, 1\}$ is called a *predicate.*

Definition *D1.7.1* - {28}*: (Computable functions).* We say that $f : A \to \mathcal{N}$ is *computable* if there exists a Turing machine M, such that

1. for all $(x_1, \ldots, x_n) \in A$, the output of the computation when M is presented with input $(x_1, \ldots, x_n)$ is $f(x_1, \ldots, x_n)$,
2. for all $(x_1, \ldots, x_n) \notin A$, M does not halt.

{31} We denote the universal computable function defined in last section by $\nu^{(2)}(i, p)$. One can generalize universal computable functions to any number of variables.

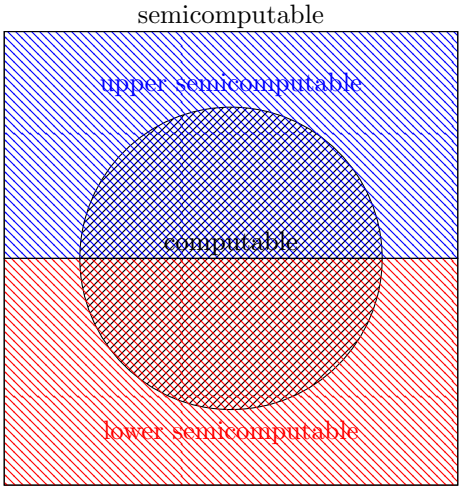
Lemma *E1.7.1/2* - {29}*: (Examples of computable functions).* $l(x)$, $\bar{x}$, $g(\bar{x}y) = x$, $h(\bar{x}y) = y$ are partial computable. The successor function $\gamma^{(1)}(x) = x+1$, the zero function $\zeta^{(n)}(x_1, \ldots, x_n) = 0$, the projection function $\pi_m^{(n)}(x_1, \ldots, x_n) = x_m$, and the pairing function $\langle x, y\rangle = \bar{x}y$ are total computable.

{34} We define an ordeing of computable functions to be exactly the same as for Turing machines: let $i \in \mathcal{N}$, and consider T_i in the standard ordering of Turing machines. ϕ_i denotes the associated computable function, and Ω_i its domain.

4.4 Semi-computable functions

{35} We write $f(y/z, k) = p/q$ to mean $f(\langle\langle y, z\rangle, k\rangle = \langle p, q\rangle)$, with $y, z, k, p, q \in \mathcal{N}$, in order to define a computable function from $\mathcal{Q}_+ \times \mathcal{N} \to \mathcal{Q}_+$. We also write $f((-1)^y z, k) = (-1)^p q$ to mean $f(\langle\langle y, z\rangle, k\rangle = \langle p, q\rangle)$, with $z, k, q \in \mathcal{N}$ and $y, p \in \{0, 1\}$, in order to define a computable function from $\mathcal{Z} \times \mathcal{N} \to \mathcal{Z}$, and from $\mathcal{Q} \times \mathcal{N} \to \mathcal{Q}$ if $z, q \in \mathcal{Q}_+$.

Definition *D1.7.4* - {35}*: (Semicomputable functions).* {35} A total function $f : \mathcal{Q} \to \mathcal{R}$ is *upper semicomputable* if there exists a partial computable function $\phi : \mathcal{Q} \times \mathcal{N} \to \mathcal{Q}$ such that $\phi(x, k+1) \leq \phi(x, k)$ for all $k \in \mathcal{N}$ and $\lim_{k\to\infty} \phi(x, k) = f(x)$. {36} A total function $f : \mathcal{Q} \to \mathcal{R}$ is *lower semicomputable* if $-f$ is upper semicomputable. A total function $f : \mathcal{Q} \to \mathcal{R}$ is *semicomputable* if it is upper or lower semicomputable. A total function $f : \mathcal{Q} \to \mathcal{R}$ is *computable* if it is upper and lower semicomputable. Equivalently, a function $f : \mathcal{Q} \to \mathcal{R}$



is computable iff there exists a computable function $g : \mathcal{Q} \times \mathcal{N} \to \mathcal{Q}$ such that $|f(x) - g(x,k)| \leq 1/k$, for all $x \in \mathcal{Q}, k \in \mathcal{N}$ (the speed of convergence is arbitrary).

4.5 Computable real numbers

Definition *E1.7.22 -* {47}*: (Computable real number).* A real number $r \in [0,1]$ is called a *computable real number* if there exists a computable function such that $r = 0.\omega$, with $\phi(i) = \omega_i$ for all i. ω is called a *computable sequence.* A computable sequence of computable real numbers is a sequence $r_1, r_2, \ldots$ such that there is a computable function ψ satisfying $\psi(i,j) = r_{i,j}$.

4.6 Computable sets

{32} A set $A \subseteq \mathcal{N}$ is *computably enumerable* (or c.e.) if it is the range of a total computable function f. We say the f *enumerates* A. Equivalently, A is c.e. if it is accepted by a Turing machine.

{32} A set $A \subseteq \mathcal{N}$ is *computable* if it ihas a computable characteristic function.

Many lemmas about computably enumerable and computable sets are on pages $\{32 - 33\}$.

{34} We define $K_0 := \{\langle x, y\rangle : y \in \Omega_x\}$ and {36} $K := \{x : x \in \Omega_x\}$.

{36} A function $f : \mathcal{Q} \to \mathcal{R}$ is lower semicomputable iff the set $\{(x,y) \in \mathcal{N} \times \mathcal{Q} : y < f(x)\}$ is computably enumerable.

5 Time and space complexity

{37} Let T be a Turing machine. We say that T has time complexity $t(n)$ if for all $x \in \mathcal{N}$ satisfying $l(x) = n$, T halts in less than $t(n)$ steps, for all $n \in \mathcal{N}$. Further, we say that T has time complexity $s(n)$ if for all $x \in \mathcal{N}$ satisfying $l(x) = n$, if T uses at most $s(n)$ cells of the tape during the computation, for all $n \in \mathcal{N}$.

To see the definition of the complexity classes $DTIME$, $NTIME$, $DSPACE$, $NSPACE$, P, NP, $PSPACE$, go to page {38}.

{40} A language over an alphabet Σ is a subset of Σ^*. If a language is accepted in deterministeic (nondeterministic) polynomial time using oracle A, we write $L \in P^A$ ($L \in NP^A$). We define the Polynomial Hierarchy according to $\Sigma_1^p = NP$, $\Delta_1^p = P$, $\Sigma_{i+1}^p = NP^{\Sigma_i^p}$, $\Delta_{i+1}^p = P^{\Sigma_i^p}$.

6 Plain complexity $C(x)$

6.1 Definitions

Definition *D2.1.1* - {105}*: (Plain complexity with respect to a method).* Let T be a computable function. {105} We define the complexity of $x \in \mathcal{N}$ with respect to T as

$$C_T(x|y) = \min \{l(p) : T(\langle y, p \rangle) = x\}, \ \forall x, y \in \mathcal{N}. \tag{17}$$

{105} Choose an enumeration of computable function $T_1, \ldots, T_n, \ldots$. {105} Define $U := T_0$ to be a universal Turing machine satisfying

$$U(\langle y, n, x \rangle) = T_n(\langle y, x \rangle), \ \forall n, x, y \in \mathcal{N}. \tag{18}$$

Theorem *T2.1.1* - {105}*: (Existence of additively optimal functions).* U is an additively optimal universal computable function.

Definition *D2.1.2* - {106}*: (Plain Kolmogorov complexity).* We define the conditional Kolmogorov complexity $C(\cdot|\cdot)$ as follows

$$C(x|y) := C_U(x|y), \ \forall x, y \in \mathcal{N}, \tag{19}$$

and the unconditional Kolmogorov complexity $C(\cdot)$ as follows

$$C(x) := C(x|\epsilon), \ \forall x \in \mathcal{N}. \tag{20}$$

{105} Let $p \in \mathcal{N}$ such that $l(p) = C_T(x)$ and $T(\langle \epsilon, p \rangle) = x$. Then, p is called a *shortest program for x by ϕ*. {109} We also define

$$C(x, y|z) := C(\langle x, y \rangle | z) \tag{21}$$

and

$$C(x|y, z) := C(x|\langle y, z \rangle) \tag{22}$$

6.2 Length-conditional complexity

This is an attempt to eliminate the non-monotonicity on prefixes.

Definition *D2.2.2 - {119}: (Length-conditional complexity).* The quantity $C(x|l(x))$ is called the *length-conditional complexity* of $x \in \mathcal{N}$.

Lemma *S2.2 - {119}: (Upper bound length-conditional complexity).*

$$C(x|l(x)) \leq C(x) + O(1). \tag{23}$$

{119} $C(x|l(x))$ is however still non-monotonic on prefixes.

6.3 Main results

Lemma *S2.1.1 - {107}: (Kolmogorov complexity as minimization of a two-part code).*

$$C(x|y) = \min_{n,p \in \mathcal{N}} \{l(\bar{n}p) : T_n(\langle y, x \rangle)\} + O(1), \ \forall x, y \in \mathcal{N}. \tag{24}$$

Theorem *T2.1.2 - {108}: (Main upper bound of plain complexity).*

$$C(x) \leq l(x) + O(1), \quad C(x|y) \leq C(x) + O(1), \quad \forall x \in \mathcal{N}. \tag{25}$$

Theorem *S2.1.3 - {111}: (Invariance theorem w.r.t. reference machine).* Let T_n, T_m be two additively optimal universal Turing machines

$$C_{T_n}(x|y) = C_{T_m}(x|y) + O(1). \tag{26}$$

Theorem *S2.1.3 - {112}: (Invariance theorem w.r.t. numbering).* $\{T_{n'} : n' \in \mathcal{N}\}$ be an acceptable numbering, and let U' be such that

$$U'(\langle y, n', x \rangle) = T_{n'}(\langle y, x \rangle), \ \forall n', x, y \in \mathcal{N}. \tag{27}$$

Then,

$$C_{U'}(x|y) = C(x|y) + O(1). \tag{28}$$

6.4 Useful results

Lemma *E2.1.3/4, P2.1.2 - {109, 113}: (Invariant operations).*

$$C(xx|y) = C(x|y) + O(1) \tag{29}$$

$$C(x^R|y) = C(x|y) + O(1) \tag{30}$$

$$C(T_n(x)|y) = C(x|y) + O(1) \tag{31}$$

$$C\left(x^{C(x)}\right) = C(x) + O(1), \ \forall x \in \mathcal{N} \tag{32}$$

Lemma *E2.1.5, P2.1.9 - {109, 110, 114}: (Almost subaddtivity).*

$$C(x, y|z) \leq C(x|z) + C(y|z) + 2\log_2 \min(C(x|z), C(y|z)) + O(1) \tag{33}$$

$$C(x, y|C(x), z) \leq C(x|z) + C(y|z) + O(1) \tag{34}$$

$$C(x, y) \leq C(x) + 2l(C(x)) + C(y|x) + O(1) \tag{35}$$

If there exists $n \in \mathcal{N}$ s.t. $C(x), C(y) \leq n$, then $C(x, y) \leq 2n + O(1)$.

***Lemma** P2.1.1 - {113}: (Simple sequence).*

$$C(0^n|n) = O(1) \tag{36}$$

***Lemma** P2.1.5 - {113}: (Manipulation of variable).*

$$C(x|y) \leq C(x,z|y) + O(1) \tag{37}$$
$$C(x|y,z) \leq C(x|y) + O(1) \tag{38}$$
$$C(x,y|z) = C(y,x|z) + O(1),\ \forall x,y,z \in \mathcal{N} \tag{39}$$

***Theorem** P2.1.6 - {114}: (Relations through computability).*

$$C(T_n(x)|y) \leq C(x|y) + 2l(n) + O(1) \tag{40}$$
$$C(y|T_n(x)) \geq C(y|x) - 2l(n) + O(1),\ \forall x,y,n \in \mathcal{N} \tag{41}$$

If T_n is one-to-one, it becomes

$$|C(T_n(x)|y) - C(x|y)| \leq 2l(n) + O(1) \tag{42}$$
$$|C(y|T_n(x)) - C(y|x)| \geq 2l(n) + O(1),\ \forall x,y,n \in \mathcal{N} \tag{43}$$

6.5 Properties as an integer function

***Theorem** T2.3.1 - {126}: (Unboundedness).* (a) $C(x)$ is unbounded,

(b) $m(x) = \min\{C(y) : y \geq x\}$ is unbounded,

(c) $m(x)$ is unbounded slowlier than any other unbounded computable function.

***Theorem** T2.3.2 - {127}: (Uncomputabilility).* $C(x)$ is uncomputable, and there is not partial function defined on infinitely many points that agrees with $C(x)$ on its domain of definition. However, $C(x)$ is semicomputable from above.

***Theorem** S2.3 - {121}: (Continuity, Logarithmicity, Fluctuability, High-complexity runs).* (a)

$$|C(x+y) - C(y)| \leq 2l(y) + O(1). \tag{44}$$

(b) $C(x)$ hugs $\log x$, i.e. $C(x) \leq \log x - k$ for at most 2^{n-k} numbers between 2^{n-1} to 2^n.

(c) $C(x)$ fluctuates rapidly, i.e. there exists x_1, x_2 within distance $\sqrt{x}$ of x such that $C(x_1) \geq l(x/2) + c$ and $C(x_2) \leq l(x)/2 - c$.

For each c there is d such that there is no d-run of c-incompressible numbers, and conversely for each d, there is a c such that there are d-runs of c-incompressible numbers.

For the definition of complexity monotonic on prefixes, see Exercise 2.3.2 {130}. For properites of Kolmogorov complexity of infinite sequences, see Exercise 2.3.4, 2.3.5 pages $\{131 - 132\}$.

6.6 Function complexity

Definition *P2.1.14* - {115} *: (Function complexity).* We define the *function complexity* as

$$C(f|D) := \min \{l(p) : U(\langle x, p\rangle) = f(x), \forall x \in D\}, \tag{45}$$

where $D \subseteq \mathcal{N}$ and $f : \mathcal{N} \to \mathcal{N}$.

Lemma *P2.1.14* - {116}*: (Upper bound function complexity).* Let $f : \mathcal{N} \to \mathcal{N}$ be a partial computable function. Then,

$$C(f| \leq n) \leq \log n + O(1). \tag{46}$$

6.7 Complexity of infinite sequences

See E2.2.15/16 on page {125} to see definitions of complexity for infinite sequences, although not that interesting.

6.8 Incompressibility

Lemma *S2.2* - {116}*: (Existence of incompressible sequence).* Let $n \in \mathcal{N}$. There exists $x \in \mathcal{N}$, with $l(x) = n$, such that $C(x) \geq n$. This follows from a counting argument.

Definition *D2.2.1* - {116}*: (c-incompressibility).* {116} Let $c \in \mathcal{N}$. We say that $x \in \mathcal{N}$ is c-incompressible if $C(x) \geq l(x) - c$.

Lemma *S2.2* - {117}*: (Counting of incompressible sequences).* Let $n, c \in \mathcal{N}$. There are at least $2^n - 2^{n-c} + 1$ c-incompressible elements $x \in \mathcal{N}$, with $l(x) = n$.

Theorem *T2.2.1* - {117}*: (Generalized counting of incompressible sequences).* Let $n, c, y \in \mathcal{N}$, and $A \subseteq \mathcal{N}$. Then, there are at least $d(A)(1 - 2^{-c}) + 1$ elements $x \in A$ satisfying $C(x|y) \geq \log d(A) - c$.

Lemma *E2.2.1* - {117}*: (Incompressible strings contains compressible strings).* Let $x \in \mathcal{N}$ of length n be an c-incompressible string. Then, for all $v \in \mathcal{N}$ being a substring of x, $C(v) \geq l(v) - O(\log n)$ (here note that $O(logn)$ can be big compared to $l(v)$, so v might be compressible).

Lemma *E2.2.2* - {118}*: (The shortest program is incompressible).* There exists $c > 0$, such that for all $x \in \mathcal{N}$, and all shortest programs p for x by U, $C(p) \geq l(p) - c$.

{120} A set $A \subseteq \mathcal{N}$ is called *meager* if $\lim_{n\to\infty} d(A^{\leq n})/2^n = 0$, with $A^{\leq n} = \{x \in A : l(x) \leq n\}$.

Lemma {120}*: (Incompressible elements in meager sets).* If A is computable and meager, then for all $c \in \mathcal{N}$ there is only finitely many c-incompressible elements in A.

Definition *D2.2.3* - {120}*: (Randomness deficiency).* We define the *randomness deficiency of x relative to A* as

$$\delta(x|A) := l(d(A)) - C(x|A). \tag{47}$$

NOTE: how to define $C(x|A)$??? I guess it can be formalized through an oracle Turing machine !

Theorem *T2.2.2* - {120}*: (Distribution of randomness deficiency).* The quantity $d(\{x : \delta(x|A) \geq k\}) \leq d(A)/2^{k-1}$.

Theorem *E2.2.6* - {122}*: (Number of incompressible constants).* There exists $d > 0$ such that there are at least $2^n/d$ strings of length n satisfying $C(x) \geq n$ and $C(x|n) \geq n$.

6.9 Randomness of finite sequences

Definition *D2.4.1* - {135}*: (P-test of randomness).* Let $\delta : \mathcal{N} \to \mathcal{N}$ be a total function, and P be a computable probability distribution on $\mathcal{N}$ (i.e. $f(n) := P(\{n\})$ is a computable function). δ is said to be a P-test if

(a) δ is lower semicomputable,

(b) $V_{n,m} := \{x \in \mathcal{N} : l(x) = n, \delta(x) \leq m\}$ satisfies $P(V_{n,m}) \leq 2^{-m}$, for all $n, m \in \mathcal{N}$.

{136} Note that as δ is total, we can change hypothesis (a) to be δ is computable.

Definition *D2.4.2* - {137}*: (Universal P-test).* A universal P-test is a P-test δ_0 such that for every other P-test δ, there exists $c > 0$ such that $\delta_0(x) \geq \delta(x) - c$, for all $x \in \mathcal{N}$.

Definition *Implicitly defined* {140}*: (Randomness).* Let δ_0 be a universal P-test. We say that $x \in \mathcal{N}$ is c-random with respect to δ_0 and P if $\delta_0(x) \leq c$.

Theorem *L2.4.1, T2.4.1* - {137 − 138}*: (Existence of universal P-test).* There exists an effective enumeration $\delta_1, \delta_2, \ldots$ of P-tests. Moreover, define $\delta_0(x|P) := \max\{\delta_y(x) - y : y \geq 1\}$ is a universal P-test.

Theorem *T2.4.2* - {139}*: (Universal test for the uniform law).* The function $\delta_0(x|L) = l(x) - C(x|l(x)) - 1$ is a universal test for the uniform distribution.

Definition *D2.4.3* - {140}*: (Reference randomness).* We say that $x \in \mathcal{N}$ is c-random if $\delta_0(x|L) \leq c$.

6.10 Randomness of infinite sequences

Theorem *T2.5.1* - {143}*: (No naïve way of defining random infinite sequences).* For any f satisfying $\sum_{n=1}^{\infty} 2^{-f(n)} = \infty$, and any $\omega \in \{0,1\}^{\omega}$, there are infinitely many $n \in \mathcal{N}$ such that

$$C(\omega_{1:n}|n) \leq n - \log n. \tag{48}$$

Definition *D2.5.1* - {148}*: (Sequential test).* Let μ be a computable probability distribution on the space $\{0,1\}^{\infty}$. A total function $\delta : \{0,1\}^{\infty} \to \mathcal{N} \cup \{\infty\}$ is a sequential μ-test if

(a) $\delta : \omega \mapsto \sup_{n \in \mathcal{N}}\{\gamma(\omega_{1:n})\}$, where $\gamma : \mathcal{N} \to \mathcal{N}$ is a total lower semicomputable function, i.e. $V = \{(m, y) : \gamma(y) \geq m\}$ is a c.e. set,

(b) $\mu\{\omega : \delta(\omega) \geq m\} \leq 2^{-m}$ for all $m \geq 0$.

{148} One can require γ to be computable without changing the definition, see the book for more details.

{149} Note that $V_m := \{\omega : \delta(\omega) \geq m\} = \cup\{\Gamma_y : (m, y) \in V\}$. We have

$$\delta(\omega) < \infty \iff \omega \notin \bigcap_{m=1}^{\infty} V_m. \tag{49}$$

Definition *D2.5.2* - {149}*: (Randomness).* Let $\mathbf{V}$ be the set of all sequential μ-tests. ω is said to be μ-random if

$$\omega \notin \bigcup_{V \in \mathbf{V}} \bigcap_{m=1}^{\infty} V_m. \tag{50}$$

Note that $\mu\left(U := \bigcup_{V \in \mathbf{V}} \bigcap_{m=1}^{\infty} V_m\right) = 0$. U is said to be the maximal constructive μ-null set.

Definition *D2.5.3* - {149}*: (Universal sequential test).* A universal sequential μ-test f is such that $f(\omega) \geq \delta(\omega) - c$, for all δ sequential μ-test.

Theorem *T2.5.2* - {150}*: (Existence of universal sequential test).* There is a universal sequential μ-test.

Theorem *T2.5.3* - {152}*: (Miller-Yu theorem).* (i) Let $f : \mathcal{N} \to \mathcal{N}$ be a computable function with $\sum_n 2^{-f(n)} < \infty$. For every λ-random ω there is a constant such that $C(\omega_{1:n}) \geq n - f(n) - c$.

(ii) There exists $f : \mathcal{N} \to \mathcal{N}$ be a computable function with $\sum_n 2^{-f(n)} < \infty$ such that for all λ-nonrandom ω there exists n such that $C(\omega_{1:n}) < n - f(n) - c$.

Definition *D2.5.5* - {153}*: (Computably convergent function).* $\sum_{i=1}^{\infty} a_i$ is computably convergent if there exists a computable sequence $n_1, \ldots, n_m, \ldots$ such that

$$\sum_{i=n_m}^{\infty} a_i \leq 2^{-m}. \tag{51}$$

Theorem *T2.5.5* - {153}*: (Proper supset of random sequences).* Let $f(n)$ be computable such that $\sum_{i=1}^{\infty} 2^{-f(i)}$ is computably convergent. Let ω be random, then $C(\omega_{1:n}|n) \geq n - f(n)$ for some n onward.

Theorem *T2.5.6* - {154}*: (Proper subset of random sequences).* Let $\omega \in \{0,1\}^{\infty}$.

(i) If $\exists c > 0$ s.t. $\exists^{\infty} n \in \mathcal{N}$ $C(\omega_{1:n}) \geq n - c$, then ω is random.

(ii) The set of sequences satisfying the above condition is of measure 1.

{157 – 159} Discussion on Mises-Wald-Church and Kolmogorov-Loveland stochastic sequences.

6.11 Statistical properties of random sequences

Here, the statistical properties can never be established "sharply" for random sequences, but weakly, by introducing deficiency functions.

Definition *D2.6.2* - {169}*: (Deficiency functions).* Let $c_1 \geq 0$ be a constant. We define the the class of c_1-deficiency functions $\delta : \mathcal{N} \to \mathcal{N}$ as satisfying $K[n, \delta(n)|n - \delta(n)] \leq c_1$. We choose a c_1 so large that all the common sublinear functions are c_1-deficient, and we refer to this class as *deficiency functions*.

The proofs in this chapter are mostly by contradiction.

Lemma *L2.6.1* - {169}*: (Statistical properties of random sequences).* There exists $c > 0$, s.t. if $C(x) \geq n - \delta(n)$, then

$$\left| \#\{x_i = 1\} - \frac{n}{2} \right| \leq \sqrt{\frac{3(\delta(n) + c)n}{2 \log e}}. \tag{52}$$

Lemma *L2.6.2* - {170}*: (Even ones and zeros remove randomness).* There exists $c > 0$, s.t. if

$$\left| \#\{x_i = 1\} - \frac{n}{2} \right| \leq 2^{-\delta(n)-c} \sqrt{n}, \tag{53}$$

then $C(x) < n - \delta(n)$.

Lemma *L2.6.2* - {171}*: (Fixed frequency of ones remove randomness).* There exists $c > 0$, s.t. if

$$\left| \#\{x_i = 1\} - \frac{n}{2} \right| = j, \tag{54}$$

then $C(x|n) < n - \frac{1}{2} \log n + K[j|n] + c$.

Theorem *L2.6.1* - {172}*: (Fixed frequency of ones remove randomness).* Let $y \in \{0,1\}^l$, with $l < \log n$. There exists $c > 0$, s.t. for all $x \in \{0,1\}^n$, if $C(x) \geq n - \delta(n)$, then

$$\left| \#\{x_i \dots x_{i+l} = y\} - \frac{n}{2^l} \right| \leq \sqrt{\alpha 2^{-l} n}, \tag{55}$$

with $\alpha = \frac{3l}{\log e} \left(K(y|n) + \log l + \delta(n) + c \right)$.

Theorem *L2.6.2* - {174}*: (Fixed frequency of ones remove randomness).* Let $x \in \{0,1\}^n$, s.t. $C(x) \geq n - \delta(n)$. Then, for sufficiently large n, each block of length

$$l = \log n - \log\log n - \log(\delta(n) + \log(n)) + O(1) \tag{56}$$

occurs at least one time.

6.12 Algorithmic properties of C

Definition *T2.7.1* - {177}*: (Post's simple set).* A simple set is a computably enumerable set such that its complement is infinite and there is no infinite computably enumerable subset.

Definition *C2.7.1* - {178}*: (Immune set).* An immune set is a set that complement is simple.

Theorem *T2.7.1* - {177}*: (Uncomputability).* See the book for details.

Theorem *C2.7.1* - {178}*: (RAND is immune).* $\{C(x) \geq l(x)\}$ is immune.

Lemma *T2.7.2* - {181}*: (Barzdins's lemma).* (a) Any characteristic sequence χ of a computably enumerable set A satisfies $C(\chi_{1:n}|n) \leq \log n + c$ for all n, where c is a constant dependant on A (but not on n).

(b) Moreover, there is a computably enumerable set such that its characteristic sequence χ satisfies $C(\chi_{1:n}) \geq n$ for all n.

6.13 Algorithmic information theory

Definition *D2.8.1* - {189}*: (Algorithmic information).* We define the algorithmic information about y contained in x as

$$I_C(x:y) = C(y) - C(y|x). \tag{57}$$

Theorem *T2.8.2/C2.8.1* - {192 − 193}*: (Almost symmetry of algorithmic information).*

$$|I_C(x:y) - I_C(y:x)| = O(\log C(x,y)). \tag{58}$$

This follows from

$$C(x,y) = C(x) + C(y|x) + O(\log C(x,y)). \tag{59}$$

Theorem *T2.8.1* - {190}*: (Entropy upper bounds complexity).* Let $x \in \{0,1\}^{nm}$, and denote by p_y be the frequency of $y \in \{0,1\}^n$ as a block of x. Denote by $H = -\sum_{y\in\{0,1\}^n} p_y \log p_y$. Then,

$$C(x) \leq m(H + 2^{n+1} l(m)/m). \tag{60}$$

See exercise 2.8.4 page {195} for interesting relations between $C(x)$ and the entropy.

7 Self-delimiting complexity $K(x)$

Definition *D3.1.1* - {204}*: (Computable prefix function).* A partial computable prefix function is a partial computable function ϕ such that $\mathrm{dom}(\phi)$ is prefix-free.

Theorem *D3.1.2* - {204}*: (Computable prefix functions are enumerable).* There is computable way to enumerate partial computable prefix functions from partial computable functions.

Definition {206}*: (Universal computable prefix function).* A universal computable prefix function is a partial computable prefix function U such that for all partial computable prefix function ψ, there exists $n \in \mathcal{N}$ such that $U(\langle n, x\rangle) = T(x)$, for all $x \in \mathcal{N}$.

Definition {206}*: (Additively optimal niversal computable prefix function).* An additively optimal universal computable prefix function is a universal computable prefix function U such that for all partial computable prefix function ψ, there exists $c > 0$ such that

$$C_U(x|y) \leq C_\psi(x|y) + c. \tag{61}$$

Theorem *T3.1.1* - {206}*: (Existence theorem).* There exists an additively optimal universal computable prefix function. We fix one additively optimal universal computable prefix function ψ_0, and we let

$$K(x|y) := C_{\psi_0}(x|y), \text{ and } K(x) := K(x|\epsilon). \tag{62}$$

Definition *D3.1.3* - {205}*: (Self-delimiting strings).* A string is said to be self-delimiting with respect to T conditional on y if $T(\langle x, y\rangle) < \infty$. A string is said to be self-delimiting if $U(x) < \infty$.

7.1 Main results

Lemma *Example 3.1.3-4* - {207}*: (Additivity).*

$$K(x, y) = K(xy) + O(1) = K(x) + K(y) + O(1) \tag{63}$$
$$C(x, y) = C(xy) + O(1) = K(x) + C(y) + O(1) \tag{64}$$

Definition $\{207 - 8\}$*: (Links with plain complexity).*

$$K(x) \leq C(x) + K(C(x)) + O(1) \tag{65}$$
$$C(x|y) \leq K(x|y) \leq C(x|y) + l^*(C(x|y)) + O(1) \tag{66}$$
$$K(x) = C(x) + C(C(x) + O(C(C(C(x)))) \tag{67}$$
$$C(x) = K(x) - K(K(x) + O(K(K(K(x)))) \tag{68}$$
$$C(x) = K(x|C(x)) + O(1) \tag{69}$$

Theorem *T3.2.1* - {212}*: (Tight upper bound).*

$$K(x) \leq l(x) + K(l(x)) + O(1), \tag{70}$$

and this upper bound is attained for every length. Moreover, there are at most $2^{n-r+O(1)}$ strings of length n such that $K(x) \leq n + K(n) - r$.

7.2 Incompressibility

Definition *D3.2.1* - {213}*: (Incompressible string).* A self-delimiting string is said to be c-incompressible if $K(x) \geq l(x) - c$, and said to be incompressible if it is 0-incompressible.

7.3 Random sequences

Theorem *T3.5.1* - {223}*: (Characterization of Martin-Löf random sequences).* An infinite binary sequence ω is Martin-Löf random w.r.t the uniform distribution iff there is a constant c such that for all $n \in \mathcal{N}$, $K(\omega_{1:n}) \geq n - c$, i.e. $\rho_0(\omega|\lambda)$ is a universal sequential Martin-Löf test for the uniform distribution.

Definition *D3.5.1* - {226}*: (Halting probability).* The halting probability is the real number defined as

$$\Omega = \sum_{U(p)<\infty} 2^{\ell(p)}. \tag{71}$$

Theorem *C3.5.2* - {228}*: (Randomness of Ω).* Ω is Martin-Löf random. The proof is effected through the characterization via prefix Kolmogorov complexity.

7.4 Algorithmic information theory

{250} We define

$$I(x : y) := K(y) - K(y|\langle x, K(x)\rangle). \tag{72}$$

Theorem *T3.8.2* - {250}*: (Symmetry of information).*

$$I(x : y) = I(y : x) + O(1). \tag{73}$$

8 Algorithmic Probability

8.1 Lower semicomputable functions

Recall that we define lower semicomputable functions according to Definition D1.7.4.

Definition *D4.1.1* - {263}*: (Universal lower semicomputable function).* A lower semicomputable function f is universal if there is an enumeration of lower semicomputable functions $f_1, f_2, \ldots$, such that $f(\langle i, x\rangle) = f_i(x)$, for all $i, x \in \mathcal{N}$.

Lemma *L4.1.1* - {263}*: (Existence).* There is a universal lower semicomputable function.

Definition *D4.1.2* - {264}*: (Lower semicomputable real number).* A real number x is lower semicomputable if the set of rational numbers below it is computably enumerable.

8.2 Measure Theory

{265} Let $\mathcal{B}$ be a finite or countably infinite set.

Definition *D4.2.1* - {265}*: (Measure, semimeasure).* A function $\mu : \mathcal{B}^* \to \mathcal{R}$ is a probability measure (in short measure) if

$$\mu(\epsilon) = 1, \tag{74}$$

$$\mu(\Gamma_x) = \sum_{b \in \mathcal{B}} \mu(\Gamma_{xb}). \tag{75}$$

and is a semimeasure if

$$\mu(\epsilon) \leq 1, \tag{76}$$

$$\mu(\Gamma_x) \geq \sum_{b \in \mathcal{B}} \mu(\Gamma_{xb}). \tag{77}$$

{266} One can transform a semimeasure into a measure by adding an element u not in $\mathcal{B}$.

Definition *D4.2.2* - {266}*: (Semicomputable semimeasure).* A semimeasure μ is semicomputable (resp. computable) if the function μ is semicomputable (resp. computable).

8.3 Discrete sample space

{267} A discrete semimeasure is a semimeasure P with $\mathcal{B} = \mathcal{N}$, and $P(w) = 0$ for all $w \in \mathcal{B}^*$ such that $\ell(w) \geq 2$. This engenders the rather simple equivalent definition D4.3.1.

Definition *D4.3.2* - {268}*: (Universal discrete semimeasure).* Let $\mathcal{M}$ be a class of discrete semimeasures. A semimeasure P_0 is universal for $\mathcal{M}$ if $P_0 \in \mathcal{M}$ and for all $P \in \mathcal{M}$ there exists a constant c_P such that $c_P P_0(x) \geq P(x)$, for all $x \in \mathcal{N}$.

Theorem *T4.3.1* - {269}*: (Existence universal lower semicomputable discrete semimeasure).* There is a semimeasure $\mathbf{m}$ that universal for the class of lower semicomputable semimeasures. In particular, we take

$$\mathbf{m}(x) := \sum_{j \geq 1} 2^{-K(j)} P_j(x). \tag{78}$$

The above semimeasure is basically defined as computable convex sum of all the lower semicomputable semimeasures.

Definition *D4.3.3* - {271}*: (Conditional probability mass functions).* Let $f(x, y)$ be a lower semicomputable function such that for all y, $\sum_x f(x, y) \leq 1$. Then, $P(x|y) := f(x, y)$ is called a lower semicomputable conditional probability mass function.

Theorem *D4.3.4,T4.3.2* - {272}*: (Universal conditional probability).* Let $P_1, P_2, \ldots$ be an enumerate of all lower semicomputable conditional probability mass functions. We define the conditional version of $\mathbf{m}(x)$ as

$$\mathbf{m}(x|y) := \sum_{j \leq 1} 2^{-K(j)} P_j(x|y). \tag{79}$$

Lemma *L4.3.2* - {272}*: (Negative results on* $\mathbf{m}$*).* $\mathbf{m}$ is incomputable and $\sum_x \mathbf{m}(x) < 1$.

Definition *D4.3.5* - {274}*: (Universal a priori probability).* For T a prefix Turing machine, we define

$$Q_T(x) = \sum_{T(p)=x} 2^{-\ell(p)}. \tag{80}$$

$Q_T(x)$ is called the *halting probability of* T *on* x , but this is not really the haltin probability, as it would be more fair to use $2^{-2\ell(p)-1}$, so it is called "probability" just because the sum $\sum_x Q_T(x) \leq 1$, but one can have $\sum_x Q_T(x) = 1$ even though T does not halt all the time. We define the universal a priori probability as being Q_U, where U is a universal prefix machine.

Definition *D4.3.3* - {275}*: (Algorithmic probability).* We define the algorithmic probability as

$$R(x) = 2^{-K(x)}. \tag{81}$$

Theorem *T4.3.3* - {276}*: (Coding theorem).* There us a constant c such that for every x,

$$\log \frac{1}{\mathbf{m}(x)} = \log \frac{1}{Q_U(x)} = K(x), \tag{82}$$

with equality up to a constant c.

Proof. Obviously, $Q_U(x) \geq R(x)$, and as Q_U is lower semicomputable, $c_{Q_U}\mathbf{m}(x) \geq Q_U(x)$. Finally, with an extended version of Kraft inequality, we can prove that there is a computable prefix code E such that $2^{-\ell(E(x))} \geq \mathbf{m}(x)$, then $K(x) \leq \ell(E(x))$. □

Theorem *T4.3.4* - {278}*: (Conditional coding theorem).* There us a constant c such that for every x,

$$\log \frac{1}{\mathbf{m}(x|y)} = \log \frac{1}{Q_U(x|y)} = K(x|y), \tag{83}$$

with equality up to a constant c.

Lemma *L4.3.4* - {278}*: (Equivalence semimeasures and halting probabilities).* Let $P_1, P_2, \ldots$ the enuemration of lower semicomputable semimeasures, and $Q_1, Q_2, \ldots$ the enumeration of halting probabilities. Then, there exists computable functions f, g such that $Q_j = \Theta(P_{f(j)})$ and $P_j = \Theta(Q_{g(j)})$.

{279} In this page, there is an amazing remark that if an outcome has many long descriptions, then it will have a short Kolmogorov complexity.

Definition *D4.3.8* - {281}*: (Sum-test).* Let P be a computable discrete semimeasure on $\mathcal{N}$. A sum P-test is a lower semicomputable function δ satisfying

$$\sum_x P(x)2^{\delta(x)} \leq 1. \tag{84}$$

A universal sum P-test is a sum P-test that additively dominates each other.

Lemma *L4.3.5* - {281}*: (Sum P-tests are stronger than Martin-Löf).* Sum P-tests are P-tests, but the converse is not true.

Theorem *T4.3.5* - {282}*: (Universal sum P-test).* (i) Let P be a computable discrete probability distribution. Then, $\kappa_0(x|P) := \log(\mathbf{m}(x)/P(x))$ is a universal sum P-test.

(ii) Let $P(\cdot|y)$ be a computable conditional discrete probability distribution. Then, $\kappa_0(x|P(\cdot,y)) := \log(\mathbf{m}(x|y)/P(x|y))$ is a universal sum $P(\cdot|y)$-test.

Definition *D4.3.9* - {283}*: (Randomness deficiency).* We let the randomness deficiency of x wrt P be

$$\delta(x|P) = \left\lfloor \log \frac{1}{P(x)} \right\rfloor - K(x) = \log \frac{m(x)}{P(x)} + O(1). \tag{85}$$

{184} Sum P-tests are then really related to randomness deficiency measures.
{286} Cool fact $E_P[\kappa_0(x|P)] = D(P||\mathbf{m})$.
{287 − 290} Universal gambling is defined via the sum-P tests, and its mind-blowing properties are studied.

Definition *D4.4.1* - {296}*: (Worst-case and average-case time complexity).* Let $P : \mathcal{N} \to \mathcal{R}_+$ be a density function, and $t(x)$ be the running time of an algorithm A on inputs $x \in \mathcal{N}$. The worst-case time complexity is defined as

$$T(n) := \max\{t(x) : l(x) = n\}, \tag{86}$$

and the P-average time complexity is

$$T(n|P) = \frac{\sum_{l(x)=n} P(x)t(x)}{\sum_{l(x)=n} P(x)}. \tag{87}$$

Theorem *T4.4.1* - {296}*: (m-Average complexity).*

$$T(n|\mathbf{m}) = \Omega(T(n)). \tag{88}$$

{297} Same holds with space complexity. On page {298}m there are techniques to sample from $\mathbf{m}$.

8.4 Continuous sample space

8.4.1 Universal continuous lower semicomputable semimeasure

Definition *D4.5.1* - {299}*: (Universal continuous semimeasure).* Let $\mathcal{M}$ be a class of continuous semimeasures. A semimeasure μ_0 is universal for $\mathcal{M}$ if $\mu_0 \in \mathcal{M}$ and for all $\mu \in \mathcal{M}$ there exists a constant c_μ such that $c_\mu \mu_0(x) \geq \mu(x)$, for all $x \in \mathcal{B}^*$.

Theorem *T4.5.1* - {299}*: (Existence universal lower semicomputable continuous semimeasure).* There is a semimeasure $\mathbf{M}$ that universal for the class of lower semicomputable semimeasures. In particular, we take

$$\mathbf{M}(x) := \sum_{j \geq 1} 2^{-K(j)} \mu_j(x). \tag{89}$$

{302 – 303} We have the same incomputability and unprobability results as for discrete measures.

8.4.2 Universal continuous prior probability

Definition *D4.5.2* - {303}*: (Monotone machines).* A monotone machine has two one-way tapes, one read-only and one write-only, and several two-way read/write tapes. The input tape contains a one-way infinite binary sequence. The monotone machine can never halt, and hence generate an infinite sequence on its output tape.

{303} $S_{\mathcal{B}} := \mathcal{B}^* \cup \mathcal{B}^\infty$.

Definition *D4.5.3* - {303}*: (Monotone functions).* Monotone machines compute monotone functions $\psi : \{0,1\}^* \to S_{\mathcal{B}}$. Each such partial function induces a mapping $\psi' : \{0,1\}^\infty \to S_{\mathcal{B}}$.

Theorem *T4.5.2* - {306}*: (Pushforward theorem).* Each semimeasure μ is lower semicomputable if and only if there is a monotone function ψ such that $\mu = \lambda_\psi$, where λ is the uniform measure.

Definition *D4.5.6* - {307}*: (Universal a priori probability).* We define the universal a priori probability as λ_U, where U is the reference universal monotone machine.

Theorem *T4.5.3* - {307}*: (Continuous Coding theorem).*

$$\log 1/\lambda_U(x) = \log 1/\mathbf{M}(x) + O(1). \tag{90}$$

8.4.3 Solomonoff normalization

Definition *D4.5.7* - {308}*: (Solomonoff normalization).* We can define measures that multiplicatively dominates all lower semicomputable semimeasures. These measures are obviously not semicomputable, but Solomonoff argues we

should still consider them. The Solomonoff normalization is the a normalization process which leaves invariant the ratio

$$P(x) = \mu(x1)/\mu(x0). \tag{91}$$

It is defined as follows: for a semimeasure μ, consider μ_{norm} as

$$\mu_{\text{norm}}(\epsilon) = 1 \tag{92}$$

$$\mu_{\text{norm}}(xb) = \mu_{\text{norm}}(x)\frac{\mu(xb)}{\sum_{a\in\mathcal{B}}\mu(xa)}. \tag{93}$$

8.4.4 Monotone complexity

Definition *D4.5.8/9 -* {310}*: (KM- and Km-complexities).*

$$KM(x) := \log\frac{1}{\mathbf{M}(x)}. \tag{94}$$

Let U be a reference universal monotone machine.

$$Km(x) := \min\{l(p) : U(p) = x\omega, \omega \in S_{\mathcal{B}}\}. \tag{95}$$

On pages $\{310 - 314\}$, the authors discuss the fact that KM and Km do not agree, and so there is an impossibility to assess a coding theorem in this case.

8.4.5 Integral tests

Integral test are an analogue to sum tests but in the continuous case.

Definition *D4.5.10 -* {315}*: (Unit integrable function).* A function $f : \mathcal{B}^\infty \to \mathcal{R}_+$ is unit integrable over X w.r.t μ if it is measurable and

$$\int_X f d\mu \leq 1. \tag{96}$$

Definition *D4.5.11 -* {315}*: (Lower-semicomputable continuous-domain function).* A function $f : \mathcal{B}^\infty to\mathcal{R}_+$ is lower semicomputable if there exists $g : \mathcal{N} \to \mathcal{N}$ which is left prefix-nondecreasing and right-nondecreasing such that

$$f(\omega) = \sup_{\omega\in\Gamma_x, k\in\mathcal{N}}\{g(x,k)\}. \tag{97}$$

Theorem *L4.5.7 -* {316}*: ().* Equivalence unit integrable and sequential test If f is lower semicomputable unit integrable function on $\mathcal{B}^\infty$ w.r.t μ then $\log f$ is a sequential μ-test. Conversely, if δ is a sequential μ-test, then $\delta - 2\log\delta - c$ is a lower semicomputable unit integrable function on $\mathcal{B}^\infty$ w.r.t μ.

Theorem *D4.5.12/T4.5.6* - {317}*: (Universal integral μ-test (μ computable)).* An integral μ-test is $\log f$ where f is lower semicomputable unit integrable wrt μ. A universal integral μ-test is additively dominating all other integral μ-tests. There is a universal integral μ-test, defined as

$$\log f_0(\omega) = \sup_{n\in\mathcal{N}} \left\{ \log \frac{1}{\mu(\omega_{1:n})} - K(\omega_{1:n}|\mu) \right\}, \tag{98}$$

whatever $K(\cdot|\mu)$ means...

8.4.6 Martingale tests

Definition *D4.5.14* - {321}*: (Martingale test).* Let $\mu : \mathcal{B}^* \to \mathcal{R}$ be a computable measure and $\gamma : \mathcal{B}^* \to \mathcal{R}$ be nonnegative, lower semicomputable such that

$$\mu(\epsilon)2^{\gamma(\epsilon)} \leq 1, \text{ i.e. } \gamma(\epsilon) \leq 0 \tag{99}$$

$$\mu(x)2^{\gamma(x)} \geq \sum_{b\in\mathcal{B}} \mu(xb)2^{\gamma(xb)}. \tag{100}$$

Then, γ is martingale μ-test. A martingale μ-test is universal if it additively dominates all others.

As usual, martingale μ-tests are μ-tests, and μ-tests are almost martingale μ-tests (L4.5.8 - {322}).

Definition *D4.5.15* - {322}*: (Sequential martingale test).* A sequential martingale μ-test is obatined from a martingale μ-test γ by defining

$$\delta(\omega) := \sup_{n\in\mathcal{N}} \gamma(\omega_{1:n}). \tag{101}$$

Theorem *T4.5.7* - {322}*: (Universal martingale tests).* Let $\mu : \mathcal{B}^* \to \mathcal{R}$ be a computable measure. Then,

(i) $\gamma_0(x|\mu) = \log \frac{\mathbf{M}(x)}{\mu(x)}$ is a universal martingale μ-test,

(ii) $\sigma_0(\omega|\mu) = \sup_n \log \frac{\mathbf{M}(\omega_{1:n})}{\mu(\omega_{1:n})}$ is a universal sequential martingale μ-test.

{323} Now there is a probably very important theorem: An infinite sequence ω is μ-random iff

$$KM(\omega_{1:n}) = Km(\omega_{1:n}) + O(1) = \log \frac{1}{\mu(\omega_{1:n})} + O(1). \tag{102}$$

8.4.7 Supermartingales

Definition *D4.5.16* - {324}*: (Supermartingales).* Let $t : \mathcal{B}^* \to \mathcal{R}_+$ such that

$$\mu(\epsilon)t(\epsilon) \leq 1 \tag{103}$$

$$\mu(x)t(x) \geq \sum_{b\in\mathcal{B}} \mu(xb)t(xb). \tag{104}$$

Then, t is called a μ-supermartingale, and a μ-martingale if there is equality in the conditions.

9 Inductive reasonning

{345} Deduction vs. induction: deduction is about drawing exact conclusions from observed data. Induction tries to generalize the observed data to finding rules that govers it. In particular, deduction is a subpart of induction, as if one finds the correct generalization, then one can draw the same conclusions from induction that from deduction.

{345} Inference vs Reasonning: inference is the fact of drawing conclusions (answer yes or no) to a question without total justification. In reasonning, we do not say yes or no in the end, but we change our belief in the different alternative answers.

Inductive reasonning is then the fact of generalizing data to a set of theories, that we each associate with a certain degree of belief.

There are guiding principles for inductive reasonning that have been proposed by philosophers over the years:

(a) Epicurus: Principle of Multiple Explanations. If more than one theory is consistent with the observations, keep all theories.

(b) Principle of indifference. A priori, we should associate to each theory an equal belief.

(c) Occam's Razor Principle. Entities should not be multiplied beyong necessity.

(d) Newton: Nature does nothing in vain.

(e) Einstein: Removing one parameter was right for sure.

(f) Bayes's rule: the belief we should put in a theory is proportional to our initial prior belief in the theory times the likelihood of the data given this theory.

(g) Hume: induction is impossible.

9.1 Universal prediction

We show that the best possible general initial belief is $\mathbf{M}(\cdot)$.

Theorem *C5.2.1* - {363}*: (Convergence in difference).* Let μ be a computable measure. Then, for a set A of μ-measure 1, one has

$$\lim_{n\to\infty} |\mathbf{M}(a|\omega_{1:n-1}) - \mu(a|\omega_{1:n-1})| = 0, \tag{105}$$

for all $\omega \in A$ and $a \in \mathcal{B}$.

Definition *D5.2.3* - {366}*: (Conditionally bounded away measure).* A measure is conditionally bounded away from zero if there exists $c > 0$ such that $\mu(a|x) > c$ for all $a \in \mathcal{B}$ and all $x \in \mathcal{B}^*$.

Theorem *T5.2.2* - {366}*: (Convergence in ratio).* Let μ be a computable measure. Then, for a set A of μ-measure 1, one has

(a) In-sequence convergence

$$\lim_{n\to\infty} \frac{\mathbf{M}(\omega_n|\omega_{1:n-1})}{\mu(\omega_n|\omega_{1:n-1})} = 1, \tag{106}$$

for all $\omega \in A$.

(b) Off-sequence convergence: suppose that μ is conditionally bounded away from zero. Then,

$$\lim_{n\to\infty} \frac{\mathbf{M}(x|\omega_{1:n-1})}{\mu(x|\omega_{1:n-1})} = 1, \tag{107}$$

for all $\omega \in A$ and all $x \in \mathcal{B}$.

We have prove that the likelihood with respect to any prior computable measure μ is very well estimated by the likelihood with respect to $\mathbf{M}$.

9.1.1 Prediction by data compression

We can prove that to chose the next chunk of the sequence ω, one can - almost all the time - choose it with low monotone Kolmogorov complexity.

Theorem *T5.2.3* - {368}*: (New prediction with low Kolmogorov complexity).* Let μ be a semimeasure that is conditionally bounded away from zero. There is a subset A of μ-measure 1 such that for every $\omega \in A$, and $y \in \{0,1\}^*$, one has

$$\log \frac{1}{\mu(y|\omega_{1:n})} = Km(y\omega_{1:n}) - Km(\omega_{1:n}) + \underset{n\to\infty}{O}(1). \tag{108}$$

Interpretation: there is a constant $p > 0$ such that considering an extrapolation $y \in \{0,1\}^*$ of $\omega_{1:n}$, $\mu(y|\omega_{1:n}) \geq p/2^c$ if $Km(y\omega_{1:n}) - Km(\omega_{1:n})$, when n is very large. Then, compressible extrapolation are associated with a bigger belief.

9.1.2 Inductive inference

Inference is the fact of finding parameters out of observed data. The model that we use here identifies the parameters in the limit, and is put in place by learning by enumeration, or learning in the limit.

9.2 Pac-learning

Consider a sample space S (discrete or continuous). Its elements are called examples. A concept c is a subset of S. We interchangeably speak about a concept and its characteristic function f. A concept class $\mathcal{C}$ is a set of concepts. The learning algorithm draws examples from the sample space S according to a fixed unknown distribution P.

Definition *D5.3.1* - {379}*: (Pac-learnability).* A concept class $\mathcal{C}$ is pac-learnable if there exists a possibly randomized learning algorithm A such that for each $f \in \mathcal{C}$ and $\varepsilon > 0$, algorithm A halts in a finite number of steps and examples, and outputs a concept $h \in \mathcal{C}$ that satisfies the following: With probability at least $1 - \varepsilon$,

$$\sum_{f(v) \neq h(v)} P(v) < \varepsilon. \tag{109}$$

A concept class is polynomially pac-learnable if it is pac-learnable and the learning algorithm always halts within time and number of examples $p(\ell(f), 1/\varepsilon)$.

Definition *D5.3.2* - {379}*: (Occam algorithm).* Let $\alpha \in [0, 1)$, $\beta \geq 1$, m a number of sample and s the length of the smallest concept in $\mathcal{C}$ consitent with the examples. An Occam algorithm is a polynomial-time algorithm that finds a hypothesis $h \in \mathcal{C}$ consistent with the examples and satisfying $C(h) \leq s^{\beta} m^{\alpha}$.

Theorem *T5.3.1* - {379}*: (Occam's razor theorem).* A concept class $\mathcal{C}$ is polynomially pac-leanable if there is an Occam algorithm for it.

Interpretation: concepts that are learnable are the ones that have a Kolmogorov complexity that is close to the lowest.

Many problems are intractable without having certain hypotheses on the sampling distribution.

Definition {382}*: (Simple distribution).* A distribution is said to be **simple** if it is multiplicatively dominated by **m**. All lower semicomputable distributions are simple, but some simple distributions are not lower semicomputable.

Theorem *T5.3.2* - {382}*: (Universal simple pac-learning).* A concept class $\mathcal{C}$ is polynomially pac-learnable under **m** iff it is polynomially learnable under every simple distribution ¶, provided that in the learning phase the set of examples are drawn according to **m**.

$\{386 - 388\}$ Similar considerations apply for the continuous case, with however the assumption of "polynomially" being removed.

9.3 Minimum Description Length principle

The MDL principle is described as follows. Given samples of data, and theories to explain this data, the best theory is the one that minimizes the sum of

- the length of the description of the theory,

- the length of the data when encoded with the help of the theory.

Definition *D5.4.1* - {391}*: (Ideal MDL).* The ideal MDL principle selects the hypothesis H that minimizes

$$K(H) + K(D|H). \tag{110}$$

{392} An alternative version of the MDL principle could be to minimize $K(H|D)$.

$\{392 - 393\}$ Logarithmic Bayes's rule: maximizing $Pr(H|D)$ is equivalent to minimize $\log \frac{1}{Pr(D|H)} + \log \frac{1}{P(D)}$. Generally, we assume $P(\cdot) = \mathbf{m}(\cdot)$, but we cannot assume $Pr(\cdot|H) = \mathbf{m}(\cdot|H)$, so the logarithmic Bayes's rule does not correspond to the ideal MDL principle. However, under some assumptions, we can do better.

Lemma *L5.4.1* - {394}*: (Ideal MDL is equivalent to Logarithmic Bayes's rule).* (i) If $P = \mathbf{m}$ then $\log 1/P(H) = K(H)$ up to an $O(1)$ additive term.

(ii) If P is computable and H is P-random, then $\log 1/P(H) = K(H)$ up to an additive term $K(P)$.

(iii) If $Pr(\cdot|H)$ is computable and D is $Pr(\cdot|H)$-random, then $\log 1/Pr(D|H) = K(D|H)$ up to an additive term $K(Pr(\cdot|H)) + O(1) \leq K(H) + O(1)$.

Motivated by previous lemma, we make the following definition.

Definition *D5.4.2* - {396}*: (Admissible hypothesis).* Let D be a data sample and P a prior probability distrobution. A hypothesis is said to be admissible if H is P-random and D is $Pr(\cdot|H)$-random.

Theorem *T5.4.1* - {397}*: (Bayes and MDL agree).* If H is admissible for D and P, then H_{bayes} and H_{mdl} are roughly equal.

Pages $\{398 - 405\}$ are dedicated to many examples of application, that are a bit obscure.

In $\{405-408\}$, the authors address the maximum of likelihood and maximum of entropy, showing that the later it roughly equal to MDL for some simple cases.

9.4 Nonparametric statistics

Definition *D5.5.2* - {410}*: (Algorithmic statistic).* A finite set S containing x is called an algorithmic statistic por x.

Definition *D5.5.3* - {410}*: (Typical element).* A string x is said to be typical element of S, and that S is a fitting model for x, if the randomness deficiency $\delta(x|S) := \log d(S) - K(x|S)$ is $O(1)$.

Definition *D5.5.4* - {411}*: (Algorithmic sufficient statistic).* A finite set S containing x is called an algorithmic sufficient statistic for x if $K(S) + \log d(S) = K(x) + O(1)$ (where $O(1)$ is independant of S and x)

Lemma *L5.5.1* - {411}*: (Sufficient implies typical).* If S is an algorithmic sufficient statistic for x, then x is a typical element of S.

Definition {412}*: (Model selection).* Let x a string. For S a set, we define

(a) the simplicity $K(S)$,

(b) the typicality $\delta(x|S)$,

(c) the shortness $\Lambda(S) := K(S) + \log d(S)$.

We define an order relation of sets that contain x as

$$S_0 \leq S_1 \tag{111}$$

if $K(S_0) \leq K(S_1)$, $\delta(x|S_0) \leq \delta(x|S_1)$ and $\Lambda(S_0) \leq \Lambda(S_1)$.

The author now proposes estimators based on the different tools provided by nonprobabilistic statistics.

Definition *D5.5.5* - {412}*: (Best-fit estimator).* Best-fit refers to the precise notion of fittingness of the model (i.e. the data is typical w.r.t. the model).

$$\beta_x(i) = \min_S \{\delta(x|S) : S \ni x, K(S) \leq i\} \tag{112}$$

Definition *D5.5.6* - {413}*: (Maximum likelihood estimator).* We define it as

$$h_x(i) = \min_S \{\log d(S) : S \ni x, K(S) \leq i\}. \tag{113}$$

Definition *D5.5.6* - {415}*: (MDL estimator).* We define it as

$$\lambda_x(i) = \min_S \{\Lambda(S) : S \ni x, K(S) \leq i\}. \tag{114}$$

The authors then establish different relations between the functions.

9.5 MDL principle

Definition *D5.5.9* - {422}*: (MDL algorithm).* Let x be a data sample. Let A be an algorithm that, given x and $i \leq l(x) + O(\log l(x))$ produces a finite sequence of pairs (p_t, S_t) such that $x \in S_t$ and p_t is a program for S_t. If $l(p_t) + \log d(S_t) < l(p_{t-1}) + \log d(S_{t-1})$, then A is an MDL algorithm.

There are then many discussions on why the MDL algorithm can be implemented in practice, but will not necessarily converge to a model that witnesses the best-fit estimator, at least in terms of a certain approximation measure. They justify that we do not care because: "We want a good model rather than a number that measures its goodness" Basically, they just want a model that always improves rather than a model that is expected to improve towards a good model.

References

[LV19] Ming Li and Paul Vitányi. *An Introduction to Kolmgorov complexity and its Applications.* Springer, fourth edition, 2019.