1 Preliminaries

Prefix codes
Prefix If $y = xz$, x is a prefix of y.
Prefix free A set A is prefix-free, if any $x \in A$ is never a prefix of any of $y \in A$.
Prefix code The code whose domain is prefix-free.
Self-delimiting code Let x be a binary string, and $\ell(x)$ be its length. Then,

$$\overline{x} = 1^{\ell(x)}0x$$

is called the self-delimiting version of the string x.
Example: $\overline{xy} = 110011110010$ implies $x = 01$ and $y = 010$.

Primitive recursive function[S8]
Church's lambda notation: Let $[\cdots x \cdots]$ be an expression such that for any integer in place of x the expression has at most one corresponding value. Then, $\lambda x[\cdots x \cdots]$ denotes the associated partial function.
The class of primitive recursive functions is the smallest class $\mathcal{C}$ closed under the following schemata:
(I) The successor function: $\lambda x[x + 1]$ is in $\mathcal{C}$.
(II) The constant function: $\lambda x_1 \cdots x_n[m]$are in $\mathcal{C}$ for all $n, m \in \boldsymbol{N}$.
(III) The identity function (projections) $\lambda x_1 \cdots x_n[x_i]$ are in $\mathcal{C}$ for $n \in \boldsymbol{N}$ and $i \in \{1, \cdots, n\}$.
(IV) (Composition) If $g_1, \cdots, g_m, h \in \mathcal{C}$, then

$$f(x_1, \cdots, x_n) = h(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n)) \tag{1.1}$$

is in $\mathcal{C}$.
(V) (Primitive recursion) If $g, h \in \mathcal{C}$ and $n \geq 1$, then $f \in \mathcal{C}$ where

$$\begin{aligned} f(0, x_2, \cdots, x_n) &= g(x_2, \cdots, x_n), & (1.2)\\ f(x_1 + 1, x_2, \cdots, x_n) &= h(x_1, f(x_1, x_2, \cdots, x_n), x_2, \cdots, x_n). & (1.3) \end{aligned}$$

Partial recursive and recursive functions
The class of partial recursive functions is the least class obtained by closing under (I)-(V) above and
(VI) (Unbounded search) If $\theta(x_1, \cdots, s_n, y)$ is partial recursive and total[1]

$$\psi(x_1, \cdots, x_n) = \mu y[\theta(x_1, \cdots, x_n, y)], \tag{1.4}$$

where μ implies the minimalization.
A total partial function is called a recursive function.

[1] A more precise definition is in [S10].

Turing machine
Two states suffice to make a Turing machine (p79; however, the number of tape symbols must be increased). Therefore, state × symbol complexity could classify Turing machines (or algorithms).
Turing's thesis: an effective procedure is realized by a Turing machine.
Church's thesis: The class of algorithmically computable numerical functions (in the intuitive sense) coincides with the class of partial recursive functions.

Gödel numbering: (or effective enumeration)

Enumeration theorem: n-variable partial recursive functions $\phi_i(x_1, \cdots, x_n)$ there is a n+1-variable partial recursive function $\psi(i, x_1, \cdots, x_n) = \phi_i(x_1, \cdots, x_n)$. ($\psi$ is the universal function for n-variables.)

Recursiveness
Recursively enumerable set: a set which is a domain of some total recursive function.
Recursive set: a set whose indicator is a total recursive function. In other words, a set is recursive if it is accepted by a Turing machine that surely halts.
Examples of recursive sets:
(1) The set of primes
(2) Any finite set or any set with a finite complement.
(3) The set of all $\{e, x, y : \phi_{e,<s}(x) = y\}$ is a recursive set, where $\phi_{e,<s}$ the Turing machine e ($< s$) and $\phi_{e,<s}(x)$ is defined if it stops with the program x by s steps. $\{e, x, y, s : \phi_{e,<s}(x) = y\}$ is also recursive.
Examples of recursively enumerable sets
(1) any recursive set.
(2) The set of partial recursive functions whose range is nonempty.
(3) The set of x such that x consecutive 1's appear in π.

Lemma 1.3
(i) A set A is recursive if A and A^c are both recursively enumerable.
(ii) An infinite set A is recursive, iff it is recursively enumerable in the increasing order.
[Demo of (ii)] (Only if) we have only to order A.
(If) Let A be ordered as $a_1 < a_2 < \cdots$. Given x we have only to check i elements satisfying $a_i \geq x$.
Lemma 1.4 Every infinite recursively enumerable set contains an infinite recursive set.
[Demo] Let f enumerate an infinite recursive set A. Order f according to its value, and define a recursive function g that gives these values in the increasing order as $g(1), g(2), \cdots$. This sequence does not end, so g is a total recursive function. [(ii) above is enough.] [Intuitively, collect all the inputs on which

the TM halts. Even though we cannot effectively find such set, there is such a set any way.]

Halting problem
Lemma 1.5 Let $K_0 = \{(x, y) : \phi_x(y) \text{ is defined}\}$, where $\phi_x(y)$ is the Turing machine x with input y. K_0 is called the halting set. K_0 is not recursive. [S19]
[Demo] K_0 is recursively enumerable (if necessary, see [S18]). Let $K = \{x : \phi_x(x) \text{ halts}\}$. If K_0 is recursive, then its indicator $\chi(x, y)$ is recursive. Therefore, $g(x) \equiv \chi(x, x)$ is also recursive, and K becomes a recursive set. Let $\psi(x) = \phi_x(x) + 1$ if $\phi_x(x)$ is computable, and $\psi(x) = 0$ if not. Since $\psi(x)$ is partially recursive function, so there must be z such that $\phi_z(x) = \psi(x)$ for all x. This cannot, however, be correct for $x = z$; a contradiction.

The set K_0 is the domain of a universal partial recursive function.

Reduction
$A \leq_m B$ (A is many-one reducible to B), if there is a recursive function such that $f(A) \subset B$ and $A^c \subset B^c$ (.e., $x \in A$ iff $f(x) \in B$).
If f is one-to-one, $A \leq_1 B$ (one=one reducible) [S19].
$A \equiv_m B$ if $A \leq_m B$ and $B \leq_m A$. If $A \leq_m B$ and B is recursive, then A is recursive.
Theorem [S20] Let $K = \{x : \phi_x(x) \text{ definable }\}$, and $\text{Tot} = \{x : \phi_x \text{ i a total function }\}$. Then, $K \leq_1 \text{Tot}$.
[Demo] □.

Notice that whether ϕ_x is constant, or $\text{dom}\phi_x \neq \emptyset$ is undecidable.

Index set and Rice's theorem
$A \subset \mathbf{N}$ is an *index set* if for all x and y, $x \in A$, and $\phi_x = \phi_y$ implies $y \in A$.
Theorem [S21]. If A is a nontrivial index set (i.e., $A \neq \emptyset$, $A \neq \boldsymbol{N}$), then either $A \leq_1 A$ or $K \leq_1 A^c$.

Incompleteness
True formulas: formulas 'true' according to some (nonconstructive) criterion of truth.
Provable formulas: formulas provable according to a syntactic notion of proof.

Theory T: a set T of formulas.
Axiomatizable theory: T is effectively enumerable.
Decidable theory: T is recursive.
Consistent theory: T that does not contain x and its negation. Soundness = consistency.
Lemma 1.6 Let T be an axiomatizable theory that is sound and extends

Peano Arithmetic. Then, there is a recursively enumerable set K_0 such that $n \notin K_0$ and the formula "$n \notin K_0$" is true in T but not provable in T.
This is the Church-Kleene version of the incompleteness theorem.
[Demo] K_0 introduced in Lemma 1.5 is a RENR set (Lemma 1.5). Suppose all true formulas of the form $n \notin K_0$ are provable in T. This means that there is a way to enumerate the complement of K_0. This implies that K_0 is a recursive function, a contradiction.

Language
Language: a language over a finite alphabet Σ is a subset L of Σ^*.
We say a Turing machine T accepts a language L, if T computes the indicator of L.

Computable complexity
Computable complexity: T has a *time complexity* $t(n)$ if T halts after at most $t(n)$ steps for any length n inputs. T has a *space complexity* $s(n)$ if T halts after using at most $s(n)$ cells for any length n inputs.
k-tape Turing machine with $t(n)$ time complexity ($s(n)$ space complexity) can be emulated by a single tape TM with a time complexity $t(n)^2$ (a space complexity $s(n)$).
Thus *complexity classes* such as DTIME[$t(n)$], NSPACE[$s(n)$] may be consistently defined.
P $\subset$ NP $\subset$ PSPACE.

Oracle
Oracle machine: If a TM has a special oracle state in which it can ask whether $x \in A$ or not for the temporal output x as many times as required, the TM is called an oracle TM with a oracle set A. A TM T with an oracle set A is denoted as T^A.

Reduction of language: A language A is polynomial time Turing reducible to a language B (denoted as $A \leq_T^P B$), if T^B accepts A in polynomial time.

NP-complete language: an NP-hard language A is said to be NP-complete, if T^A can reduce any NP problem to P.
Polynomial hierarchy:
$\Delta_1^p = P$, $\Sigma_1^p = NP$.
$\Sigma_{n+1}^p = NP^{\Sigma_n^p}$, $\Delta_{n+1}^p = P^{\Sigma_n^p}$.

Randomness
p48 *Von Mises collective*: LLN + place selection rule yielding sequences with the same LLN.
According to Kolmogorov: "Generally speaking, there is no ground to believe that random phenomena should possess any definite probability. Therefore, we

should distinguish between randomness proper (as absence of any regularity) and stochastic randomness (which is the subject of probability theory). There emerges the problem of finding reasons for the applicability of the mathematical theory of probability to the real world."

Definitions in terms of initial segments are the effectively meaningful approaches.

Information theory
Shannon-Fano coding: Let the frequency of symbols be $p_1 \geq p_2 \geq \cdots$. The code length $\ell(r)$ for symbol r is chosen to satisfy

$$2^{-\ell(r)} \leq p_r \leq 2^{1-\ell(r)}. \tag{1.5}$$

In this case the entropy per symbol is asymptotically H.

p71 **Theorem 1.2 [noiseless coding theorem]**. Let H be the entropy of the source. Then, $H \leq L \leq H+1$ for the optimal code, where $L = \sum_x P(x)\ell(x)$. [Demo is straightforward with the aid of the Shannon-Fano code and Kraft's inequality (Th 1.1).]

Codes
Prefix code: the code consisting of prefix free codewords. This is instantaneously decodable uniquely.
Unique decodability does not require the prefix property (cf. 0, 01, 011, 0111, $\cdots$ for 0, 1, 2, $\cdots$, respectively). The prefix codes are characterized by the property that the end of a code is always recognized as such.

p69 In terms of the tree representation of the codes, a prefix code corresponds to the one without any internal node corresponding to a word.
Theorem 1.1 [Kraft's inequality] A prefix code exists if and only if

$$\sum_r 2^{-\ell(r)} \leq 1. \tag{1.6}$$

Here, $\ell(r)$ is the length of the binary code word encoding the r-th word.
[Demo] 'Only if' is easy, because the prefix codes are longer than the original (a graphical way is to illustrate the code sequence as an subset of $[0,1)$; prefix codes are a collection of disjoint such subsets). To demonstrate 'If' we have only to construct a prefix code with the aid of the tree.
NB: The above theorem says that (1.6) implies there is *a* prefix code; Not all the codes satisfying (1.6) are prefix. Example: $\{0, 00, 11\}$.
NB: Uniquely decodable codes must satisfy (1.6) (mentioned in Ex 1.35).

Optimal code: The code with the shortest codeword length expectation value. Decoding procedures should be effective. If the number of words is finite, we

have only to make a table, but if there are infinitely many, we need a method to recognize a code word.
Self-delimiting code If there is a Turing machine that can recognize code words and decode them, then we call the code self-delimiting.
Universal code: The code such that there is a constant c satisfying $L/\max(H,1) \leq$
p74 c for any source. If $c=1$ asymptotically in the $H \to \infty$ limit, then the code is said to be *asymptotically optimal*.
For any binary sequence x $\overline{\ell(x)}x$ gives asymptotically optimal universal coding.

2 Complexity

Complexity
Conditional complexity: Let ϕ be a partial recursive function, and $x = \phi(\langle p, y\rangle)$, where $\langle p, y\rangle$ is a bijection between the pair (p, y) and the singleton $\langle p, y\rangle$ (say, $= \overline{p}y$). The complexity $C_\phi(x \,|\, y)$ of x conditional to y is defined by

$$C_\phi(x \,|\, y) = \min\{\ell(p) \,:\, x = \phi(\langle p, y\rangle)\}. \tag{2.1}$$

(If there is no such p, C_ϕ is interpreted as ∞.)

As usual we can introduce a universal partial recursive function, and can define the *conditional Kolmogorov complexity*, fixing a universal function:

$$C(x \,|\, y) = C_{\phi_0}(x \,|\, y). \tag{2.2}$$

If $y = \epsilon$, this defines the unconditional complexity.
Theorem 2.2. There is a constant c such that

$$C(x) \leq \ell(x) + c, \tag{2.3}$$
$$C(x \,|\, y) \leq C(x) + c. \tag{2.4}$$

[Demo] (2.3) follows from a copying machine. (2.4) follows because y need not be used (or should be used only when printing x becomes easier).

It is easy to show

$$C(x, y) \leq C(x) + C(y) + O[\log(\min\{C(x), C(y)\})]. \tag{2.5}$$

[Demo] Use a TM to print x and y and separate them. To separate x and y the TM must be able to tell when the program for x (or y) ends. Therefore, the program length must be specified. This requires $\log C(x)$. This argument tells us

$$C(x, y \,|\, C(x)) \leq C(x) + C(y) + O[1]. \tag{2.6}$$

Incompressibility
x is *c-incompressible*, if

$$C(x) \geq \ell(x) - c. \tag{2.7}$$

If $\ell(x) = n$, there are 2^n such strings. It is very likely that $C(x) \sim n$. Or more precisely, the number of the strings of length n with $C(x) \leq n - c$ is about 2^{n-c}. Therefore, c-compressible strings are very rare.
NB: Notice that here x is not an infinite sequence. If x_n is the n-length prefix of an infinite sequence x, then the above formula does not hold for all n (see Theorem 2.10).
Notice that the very incompressibility of a sequence requires that it has compressible substrings; if not, then the sequence is classified as a sequence without certain set of subsequences. This allows the sequence to be compressed.

Theorem 2.4. Let $A \subset \boldsymbol{N} \times \boldsymbol{N}$ be recursively enumerable, and $Y = \{x : (x, y) \in A\}$ be finite. Then, for all $x \in Y$

$$C(x \,|\, y) \leq \ell(Y^\circ) + c \tag{2.8}$$

for some constant c that depends only on A.
Randomness deficiency: Let $x \in A$. Then, we can define $C(x \,|\, A)$. The randomness deficiency of x relative to A is defined by

$$\delta(x \,|\, A) \equiv \ell(A^\circ) - C(x \,|\, A). \tag{2.9}$$

Th 2.4 implies that $\delta(x \,|\, A) \geq -c$. The deficiency measures the discrepancy of maximal complexity of a string in A and the complexity of x.
If $\delta(x \,|\, n) = O[1]$ ($A = \{x : \ell(x) = n\}$), we say x is a random finite string (= Martin-Löf's notion of randomness). In this case $\delta(x \,|\, n) = n - C(x \,|\, n) + O[1]$.

Properties of C
Theorem 2.5. Let $m(x) = \min_{y \geq x} C(y)$ (We know that C is essentially majorized by $\log x$ from $C(x) \leq \ell(x) + c$).
(i) C is unbounded.
(ii) m is unbounded.
(iii) m is bounded by any unbounded monotone increasing recursive function for all $x \geq x_0$ for some x_0. That is, m increases slower than any unbounded increasing recursive function).
NB: (ii) and (iii) do not hold for $C(x \,|\, \ell(x))$, because it drops to a constant for infinitely many x. [This is because there is a special $x = n2^{n-\ell(n)}$ for each positive integer n (called *n-string*).]
[Demo of (i) and (ii)] Since there are only finitely many programs of a given length, m must increase without bound. Thus, (ii) that implies (i).
[Demo of (iii)] Assume the contrary. There must be a recursive function ψ such that there are infinitely many x where $\psi \leq m$. $F(a) = \max\{a : \psi(x) \leq a + 1\}$

is a recursive function, so $C(F(a)) \leq \ell(a) + O[1]$ (use a universal recursive function to emulate F). On the other hand, $C(F(a)) > a$ by definition of F, we must conclude that $a \leq \ell(a)+c$ for infinitely many a, but this is impossible.
p105 We can estimate the frequency of the dips in C. n-strings appear roughly $\log x$ up to x, so there are at least $\log x$ dips.

Theorem 2.6.
(i) C is not computable.
(ii) There is no computable function that agrees with C at infinitely many points.
A demo of (ii) is quite similar to the above (iii).

Theorem 2.7. There is a monotone decreasing sequence of total recursive functions $\{\psi_n\}$ such that $\lim_n \psi_n(x) = C(x)$.
NB: However, for a given x it is impossible to decide the needed n.
C is continuous in the sense that there is a constant c such that

$$|C(x) - C(x \pm h)| \leq 2\ell(h) + c. \tag{2.10}$$

C hugs $\log x$. [This is due to Th2.2 and (2.7).]
C fluctuates rapidly. [By changing the last one half sequence of x (that is, within $\pm\sqrt{x}$ range of x) C can be changed by $\ell(x)/2$.]
For each c there is r such that there is no length r run of c-incompressible numbers. [To show this, we use the fact that there are special $x = p2^q$ such that $C(p2^q) \sim \ell(p) + \ell(q) + c \sim \ell(p2^q) - r$ (that is, we can choose p and q for each c and r. Here, precisely speaking $\sim$ should be interpreted as $\leq$.]
For each r there is c such that there is a length r run of c-incompressible numbers.

Test
If incompressible sequences have various properties or randomness known from the theory of probability (= stochasticity), equating incompressibility and randomness may be admissible. We invent statistical tests to reject non-random numbers.
P-text or *Martin Löf test*: Let P be a recursive probability on $\boldsymbol{N}$. If a total recursively enumerable function δ satisfies

$$P(\{x : \delta(x) \geq m, \ell(x) = n\}) \leq 2^{-m} \tag{2.11}$$

for all n, then δ is called a P-test.
Example: Uniform distribution L_n for length n string is $L_n(x) = 2^{-n}$, if $\ell(x) = n$ and 0 otherwise. In this case (2.11) may be written as $\{x : \delta(x) \geq m, \ell(x) = n\}^\circ \leq 2^{n-m}$.
Lemma P-tests are effectively enumerable.

Universal test
Universal P-text δ_P is a test such that

$$\delta_P(x) \geq \delta(x) - c \tag{2.12}$$

for some c for each δ. Its existence is a major result.
Theorem 2.8. $\max\{\delta_n(x) - n : n \in \boldsymbol{N}^+\}$ is a universal P-test, if δ_n is an enumeration of all the P-tests.
To demonstrate this, first we must show that all the P-tests are enumerable.
Theorem 2.9. For a uniform distribution $\delta_U(x) = \ell(x) - C(x \,|\, \ell(x) - 1$ is a universal test.
c-randomness: If $\delta_U(x) \leq c$, we say x is c-random.
Notice that if x is incompressible $\ell(x) \sim C(x) \sim C(x \,|\, \ell(x))$, so randomness is related to incompressibility.
p111 The fluctuation of the number of 1 in a Martin-Löf random number must be $O[\sqrt{n}]$.

Infinite random sequence
Let ω be an infinite binary sequence, and ω_n be its length n prefix.
There is no ω such that for all n $C(\omega_n) \geq n - c$ for some constant c, because $C(\omega_n)$ can be far less than $n = \ell(\omega_n)$:
Theorem 2.10. Let f be a total recursive function satisfying

$$\sum_{n=1}^{\infty} e^{-f(n)} = \infty \tag{2.13}$$

(i.e., f should not grow too fast; $\log n$ satisfies the condition). Then, for infinitely many n

$$C(\omega_n \,|\, n) \leq n - f(n). \tag{2.14}$$

If f satisfies $C(n \,|\, n - f(n)) = O[1]$, then for infinitely many n $C(\omega_n \,|\, n) \leq n - f(n)$.
For highly random sequence we have $C(\omega_n) \geq n - 2\log n \log\log n$, so the above estimate is very accurate, but to make this statement we must define highly random sequences.

p117 Let γ be a total enumerable function such that $\{(m, x) : \gamma(x) \geq m\}$ is a recursively enumerable set. Let $\delta(\omega) = \sup_n \gamma(\omega_n)$ and μ be a recursive probability measure on $\{0,1\}^\infty$. δ is a *sequential μ-test*, if

$$\mu(\{\omega : \delta(\omega) \geq m\}) \leq 2^{-m} \tag{2.15}$$

for each $n \geq 0$. (In short, γ is a function that detect a regularity, and gives a large number for rare regularity.) If $\delta(\omega) = \infty$ we say ω fails δ (δ rejects ω). The set of δ-rejected sequences has μ-measure zero.
An infinite binary sequence ω is called μ-random, if ω passes all the sequential μ-tests.
Theorem 2.11. There is a *universal sequential μ-test* δ_μ such that for any sequential μ-test δ there is a constant c such that $\delta_\mu(\omega) \geq \delta(\omega) - c$.

ω is called *μ-random in the sense of Martin-Löf* (or simply *random*), if $\delta_\mu(\omega) < \infty$.
Theorem 2.12. The totality of the μ-random sequences are μ-measure one.
Theorem 2.13. Let f be a recursive function and $\sum_n 2^{-f(n)}$ be a computable convergent sequence. If ω is random, $C(\omega_n \mid n) \geq n - f(n)$ for all n larger than some fixed number.
In this sense Theorem 2.10 is optimal.

Theorem 2.14. If $C(\omega_n) \geq n - c$ for some constant c infinitely often, then
(A) ω is random.
(B) ω is uniform measure one.
Let U be the set characterized by 2.13, and V by 2.14. Then,

$$U \subset \{\text{random sequences}\} \subset V. \tag{2.16}$$

Statistical properties of finite sequences
This is more complicated than the infinite sequences.

Algorithmic property of C
Theorem 2.17.
(i) The set $\{(x, a) : C(x) \leq a\}$ is a RENR set.
(ii) Let f be a total recursive function such that it is bounded from below by a monotone increasing unbounded function, but bounded from above with $\log x$. Then,

$$B = \{x : C(x) \leq f(x)\} \tag{2.17}$$

is recursively enumerable and B^c is infinite but does not contain an infinite recursively enumerable subset (that is, B is *simple*).

Since an axiomatizable theory T is a set of formulas that can be a range of a partially recursive function function, there is number k such that $C(T) \leq k$. T cannot be used to demonstrate the randomness of any number much longer than k bits (demo on p135).
C2.5. There is a recursively enumerable set B ($(B^c)^\circ = \infty$) such that for every axiomatizable sound theory T only finitely many formulas outside B is both true and provable in T.

Characteristic sequence
Let $A \subset \boldsymbol{N}$. χ is the *characteristic sequence* of A, if $\chi(i) = 1$ if $i \in A$; otherwise $\chi(i) = 0$.
If A is recursive, then $C(\chi_n \mid n)$ is bounded by a fixed constant.
Theorem 2.18 [Barzdin's lemma]
(i) For any recursively enumerable set A, its characteristic sequence satisfies $C(\chi_n \mid n) \leq \log n + c$ for all n and for some$c$.
(ii) There is a recursive enumerable set whose characteristic sequence satisfies

$C(\chi_n) \geq \log n$ for all n.

Algorithmic information theory
Algorithmic information $I_C(x;y)$ of y contained in x is defined as

$$I_C(x:y) = C(y) - C(y \,|\, x). \tag{2.18}$$

$I_C(x:x) = C(x)$, because $C(x \,|\, x) = 0$.
Theorem 2.19. Let y_i be a binary sequence of length r, and $x = y_1 y_2 \cdots y_m$. p_k be the empirical probability of k-th length r binary sequence appearing in x. Then,

$$C(x) \leq m(H + \epsilon(m)), \tag{2.19}$$

where $H = -\sum_k p_k \log p_k$ and $\epsilon(m) = 2^{r+1}\ell(m)/m$.
A tighter result will be found in Section 4.

Deficiency of plain complexity
(1) Not subadditive: $C(x,y) \leq C(x) + C(y)$ is not always correct (cf. (2.5)).
(2) $C(\omega_n) \geq n - c$ cannot characterize an infinite random sequence.
(3) Algorithmic probability cannot be defined as $P = 2^{-C}$, because it cannot be normalized.
(3) Relation to Shannon's formula has an additional term.

These deficiencies may be removed with the aid of prefix machines.

3 Prefix Complexity

$\phi : \{0,1\}^* \to \boldsymbol{N}$ is a *Partial recursive prefix function* if $\phi(x)$ and $\phi(y)$ are defined, then x is not a proper prefix of y. Prefix functions are enumerable. The TM computing a prefix function is called a *prefix machine*.
Theorem 3.1. There exists a universal prefix machine (universal partial recursive prefix function).
Prefix complexity of x: $K(x \,|\, y) = C_{\psi_0}(x \,|\, y)$, where ψ_0 is a universal prefix machine.
K is subadditive:

$$K(x,y) \leq K(x) + K(y) + O[1]. \tag{3.1}$$

C and K are asymptotically equal in the following sense:

$$C(x \,|\, y) \leq K(x \,|\, y) \leq C(x \,|\, y) + 2\log C(x \,|\, y). \tag{3.2}$$

According to Solovay a more precise relation is

$$K(x) = C(x) + C(C(x) + O[C(C(C(x)))]. \tag{3.3}$$

Therefore, there must be many K incompressible strings.
Theorem 3.2.
(i) For each n $\max\{K(x) : \ell(x) = n\} = n + K(n) + O[1]$.
(ii) For each r, the number of x of length n with $K(x) \leq n + K(n) - r$ does not exceed $2^{n-r+O[1]}$.

It is sensible to define x to be *incompressible*, if $K(x) \geq \ell(x)$.
Let x^* be the shortest program for x. Then, $K(x^*) = \ell(x^*) + O[1]$. Therefore, the fraction of the shortest programs among length n strings is at most $2^{-K(n)+O[1]}$, which goes to zero as $n \to \infty$.
p177

As a function K and C behave similarly. However, K is *co-enumerable*; that is,

$$\{(m, x) : K(x) \leq m\} \tag{3.4}$$

is recursively enumerable.

Random infinite sequence
$K(\omega_n)$ is not monotonic in n: for a random sequence it oscillates in the wedge

$$n + O[1] \leq K(\omega_n) \leq n + K(n) + O[1] \tag{3.5}$$

as seen in
Theorem 3.3. ω is random (defined in terms of C) with respect to the uniform measure iff $K(\omega_n) \geq n - c$ for some constant c.
As to oscillation of $K(\omega_n)$ see p184-5.

Halting probability
Let U be a universal prefix machine.

$$\Omega = \sum_{U(p)<\infty} 2^{-\ell(p)} \tag{3.6}$$

is the halting probability of U when the programs are sampled uniformly. Ω is random. If we know Ω_n, then for all programs of length not more than n, we can decide the halting problem.
p186

4 Algorithmic Probability

Simplicity = low K.
Ockam's razor implies that we a priori consider objects with short descriptions more plausible than the objects with only long descriptions. The probability 2^{-K} seems to be in accord with this.