Concatenation of words \( v, z \in G^* \) is written \( v \cdot z \) or just \( vz \). Thus \( \text{cat} \cdot \text{nip} = \text{catnip} \). So \( G^* \) is a semigroup under concatenation, with \( \varepsilon \) the identity element. Use \( \text{Len}(v) \) or \( |v| \) for the length of word \( v \), and have \( v > n \) mean that \( |v| > n \). For \( n \) a natnum, let \( v^n \) mean the concatenation \( vv \ldots v \). So \( v^0 = \varepsilon \).

A **language** is a subset \( L \subset G^* \). Here are six distinct languages:

\[
\emptyset = \{\} , \{\varepsilon\} , \{\text{catnip}\} , \{\text{cat} , \text{nip}\} , \{\varepsilon , \text{cat} , \text{nip}\} \\
\text{and } \{\text{bc} , \text{bac} , \text{baac} , \text{baaac} , \ldots \} = \{ba^n c\}^\infty_{n=0} .
\]

The first five are finite languages, having cardinalities \( 0, 1, 2, 3 \). Call \( \emptyset \) the **void language** and call \( \{\varepsilon\} \) the **nullword language**. The “concatenation of languages” \( K , L \subset G^* \) is

\[
K \cdot L = K \cdot L := \{v \cdot w \mid v \in K \text{ and } w \in L\} .
\]

So \( \emptyset \cdot L = \emptyset = L \cdot \emptyset \) and \( \{\varepsilon\} \cdot L = L = L \cdot \{\varepsilon\} \). Let \( L^n \) mean \( L \cdot \ldots \cdot L \). Hence \( L^0 = \{\varepsilon\} \), since the nullword language is the identity element for language-concatenation.

Define the **Kleene star** operator by

\[
L^* := \bigcup_{n=0}^\infty L^n .
\]

In particular \( \emptyset^* = \{\varepsilon\} = \{\varepsilon\}^* \). Similarly, the **Kleene plus** operator is

\[
L^+ := \bigcup_{n=1}^\infty L^n .
\]

Hence \( [\varepsilon \in L^+] \iff [\varepsilon \in L] \iff [L^+ = L^*] \). Each Kleene op is idempotent: \( [L^*]^* = L^* \) and \( [L^+]^+ = L^+ \).

**Prefix/Suffix.** For words, say “\( v \) is a *prefix* of \( w \)” if there exists a word \( z \) with \( vz = w \); write \( v \preceq w \) for this relation. If, also, \( v \neq w \), then \( v \) is a *proper prefix* of \( w \), written \( v \prec w \).

If \( \exists z \in G^* \) with \( zv = w \), then “\( v \) is a *suffix* of \( w \).” [However, we have no special symbol for the relation.]

---

**Formal languages**

Use \( \emptyset \) for the empty set. An **alphabet** \( G \) is a non-empty set, whose members are called “letters”; usually \( 2 \leq |G| < \infty \). A **word** (over an alphabet \( G \)) is a *finite* string of letters; Use \( G^* \) for the set of all words, and use \( \varepsilon \) for the **nullword**, the unique length-zero word. E.g if \( G = \{a, b\} \), then \( G^* \) equals

\[
\{\varepsilon , a , b , aa , ab , ba , bb , aaa , aab , \ldots \} .
\]

Write \( G^+ \) for \( G^* \setminus \{\varepsilon\} \).
**Codes**

For the time being, a *code* \( C \) means a non-void subset \( C \subset G^+ \); usually \( 2 \leq |C| < \infty \). Occasionally it is convenient to consider collections \( C \) which **might own \( \varepsilon \)**. So if all we know is that \( C \subset G^+ \), then we call \( C \) a *nullish code*. If we can later on prove that \( C \not\ni \varepsilon \), then we’ll have shown \( C \) to be a code.

Call \( C \) a *block code* if all its codewords have the same length. E.g., \{FBI, CIA\} is a blockcode, whereas \{Go, Gators\} is *not* a blockcode, although it is (see below) a prefixcode. [Caveat: “block code” is used with slightly different meanings in the literature. Perhaps constant-length code is a more accurate term.]

A code \( C \) is **uniquely decodable** (a UD-code) if each code-message \( z \in C^* \) has a unique decomposition w.r.t \( C \). That is, if words \( v_j, w_k \in C \) satisfy

\[
1.1: \quad \text{if } v_1v_2\ldots v_j = z = w_1w_2\ldots w_K \quad \text{then } J = K \quad \text{and } \forall i: v_i = w_i.
\]

A *prefix code* \( C \) (more accurately called a “prefix-free code”) has no codeword being a proper prefix of another. Prefix-codes are UD-codes since, stronger than (1.1), they have the **RI-UD property** (the right-infinite-UD property) that

\[
1.2: \quad \left[ v_1v_2\ldots v_j = w_1w_2\ldots w_K \right] \Rightarrow \left[ \forall i \in \mathbb{Z}^+: v_i = w_i \right].
\]

We have these non-reversible implications

\[
1.2': \quad \text{Block } \implies \text{Prefixcode } \overset{\ast}{\implies} \text{RI-UD } \overset{\ast}{\implies} \text{UD}.
\]

A code showing \((\ast2)\) non-reversible has these words

\[
1.2'': \quad v := b, \quad w := ba, \quad z := aa.
\]

It is uniquely decodable (Exer. E1), yet fails (1.2), since \( vzzz\ldots \) equals \( wzzz\ldots \). Finally, that \((\ast1)\) is non-reversible will be shown by (1.5), the “Chris code”.

A *suffix code* (no codeword is a proper suffix of another) is automatically a UD-code. Dually to (1.2') we have non-reversible implications

\[
1.3': \quad \text{Block } \implies \text{Suffixcode } \implies \text{LI-UD } \implies \text{UD},
\]

where a left-infinite-UD-code (a LI-UD-code) satisfies

\[
1.3: \quad \left[ \cdots v_2v_1 = \cdots w_2w_1 \right] \Rightarrow \left[ \forall i \in \mathbb{Z}^-: v_i = w_i \right].
\]

Note (1.2'') is an example of a suffixcode which is not a prefixcode.

### Bi-infinite

A bi-\( \infty \) \( G \)-string \( \sigma \) can be viewed as a map \( \sigma: \mathbb{Z} \rightarrow G \). A \( \mathcal{C} \)-parsing of \( \sigma \) is a sequence

\[
\cdots < k_2 < k_1 < k_0 < k_1 < k_2 < k_3 < \cdots
\]

of integers st. each substring \( \sigma|_{[k_\ell-k_{\ell+1}]} \) is a codeword, that is, lies \( C \). Write sequence \((k_\ell)_{\ell \in \mathbb{Z}} \) as \( \mathbf{k} \).

Say that \( C \) has the bi-infinite-UD property (is **BI-UD**) if

\[
1.4: \quad \text{Each bi-} \infty \text{ string } \sigma \text{ which has a } \mathcal{C}-\text{parsing, has only one } \mathcal{C}-\text{parsing. I.e., with } \mathbf{j} \text{ and } \mathbf{k}, \text{ two } \mathcal{C}\text{-parsings of } \sigma, \text{ then the sets } \{j_i\}_{i \in \mathbb{Z}} \text{ and } \{k_\ell\}_{\ell \in \mathbb{Z}} \text{ are equal.}
\]

Slightly weaker, consider two parsings \( \mathbf{j} \) and \( \mathbf{k} \), and let \( v_\ell := \sigma|_{[j_\ell-\ell+1]} \) and \( w_\ell := \sigma|_{[k_\ell-k_{\ell+1}]} \). The weak-BI-UD property asserts

\[
1.4^{\text{weak}}: \quad \forall \ell \in \mathbb{Z}: \quad v_{\ell+T} = w_\ell.
\]

(I.e., one parsing may be a shift of the other, but the codeword sequences are the same.)

Immediately,

\[
1.4': \quad \text{BI-UD } \overset{\ast}{\implies} \text{weak-BI-UD } \overset{\ast}{\implies} \quad \left[ \text{Both LI-UD and RI-UD} \right].
\]

The code \( \{bbb\} \) produces \( \sigma := \cdots bbb \cdots \), which is its only bi-\( \infty \) string. This \( \sigma \) has three parsings, since the cutpoints \( j \) can all be mod-3 congruent to \(-1\) or \( 0 \) or \( 1 \). Yet each parsing yields the same codeword sequence, namely \( \cdots bbb bbb bbb bbb \cdots \). Hence \((\ast3)\) is *not* reversible.

The “Pirate code” \( \{OH, HO\} \) is trivially LI-UD and RI-UD, since it is a blockcode. Yet the Pirate code admits bi-\( \infty \) string \( \cdots HOHOHOH \cdots \), which can be parsed as \( \cdots OH OH OH \cdots \) or as \( \cdots OH HO HO \cdots \), two different codeword sequences. Yup; \((\ast4)\) *ain’t reversible either*.

The “Chris code” (evidently a cry for help)

\[
1.5: \quad \{S, SOS\}
\]

is BI-UD, since each occurrence of “\( 0 \)” must lie in \( \text{SOS} \), and every other codeword must be \( \text{S} \). Not being a prefixcode, (1.5) proves \((\ast1)\) not reversible.
Trees. Here, a (rooted) tree is a set \( T \) of nodes, equipped with two operators: \( \text{Root}(T) \) is the root-node of \( T \). For each node \( v \in T \), let \( \text{Kids}(v) \) be the set of children of \( v \). A node \( w \) is a leaf-node if: The set \( \text{Kids}(w) \) is empty. A tree has the property that, from the root-node, one can get to an arbitrary node, by applying the \( \text{Kids}(\cdot) \) operator finitely-many times. 

Trees \( T \) and \( S \) are (tree-)isomorphic if there exists a bijection \( f: T \to S \) such that:

TI 1: \( f(\text{Root}(T)) = \text{Root}(S) \). 

TI 2: For each \( v \in T \):

\[
\{ f(k) \mid k \in \text{Kids}(v) \} = \text{Kids}(f(v)).
\]

For a \( \Gamma \in \mathbb{Z}_+ \), a tree is \( \Gamma \)-bounded if each node has at most \( \Gamma \) many children. The tree is \( \Gamma \)-full if every node is either has no children [is a leaf-node], or has precisely \( \Gamma \) many children; otherwise, the tree is \( \Gamma \)-deficient.

### 2. Kraft-McMillan Inequality.

Consider a countable code \( C \) over finite alphabet \( G \). If \( C \) is a UD-code then

\[
\sum_{v \in C} 1/\Gamma^{\text{Len}(v)} \leq 1,
\]

where \( \Gamma \) is the number of letters in \( G \).

Conversely, consider posints \( \ell = (\ell_1, \ell_2, \ldots, \ell_R) \).

If \( \sum_{j=1}^R \ell_j \leq 1 \) then there exists a prefix \( G \)-code \( C = (v_1, \ldots, v_R) \) with each \( \text{Len}(v_j) = \ell_j \).

The also result holds for infinite tuples \( \ell = (\ell_1, \ell_2, \ell_3, \ldots) \) that satisfy \( \sum_{j=1}^\infty 1/\ell_j \leq 1 \).

**Exer. E2.** Give an example of a code, \( X \), that violates (2a). [So \( X \) must fail to be UD.]

**Defn.** A code \( C \) is weakly-UD if the following holds. For each posint \( N \) and words \( v_i, w_i \in C \):

\[
1.1': \quad \text{If } v_1 v_2 \ldots v_N = w_1 w_2 \ldots w_N \text{ then } \forall i : v_i = w_i \text{.}
\]

Contrast this with the (1.1) defn of UD.

**Exer. E3.** Posting race: Who can be the first to post a code which is weakly-UD, but not UD?

**Soll to E3.** Impossible. Consider a non-UD. So there exist codewords \( v_j, w_k \) and bounds \( J \neq K \) for which

\[
v_1 v_2 \ldots v_J = w_1 w_2 \ldots w_K =: z \text{.}
\]

WLOG \( v_1 \neq w_1 \). [If they are equal, discard both. Continue. You won't discard all the words, since \( J \neq K \). Let \( N := J + K \). Notice that the word \( zz \) has two decompositions, each using precisely \( N \) many codewords:

\[
v_1 \ldots v_J w_1 \ldots w_K = w_1 \ldots w_K v_1 \ldots v_J \text{.}
\]

Moreover, these decompositions differ, since \( v_1 \neq w_1 \). Hence the code turned out not be weakly-UD.

### Preliminaries for (2a).

The below proof uses \( S_{n,\ell} \), the number of length-\( \ell \) strings which are concatenations of \( n \) many codewords. E.g., consider a code \( C = \{ v, w, z \} \) with lengths 5, 7, 8, respectively.

\[
S_{1,15} = |\varnothing| = 0. \quad S_{2,15} = |\{ wz, zw \}| = ?, \quad S_{3,15} = |\{ vvv \}| = 1. \quad S_{4,15} = |\varnothing| = 0.
\]

Indeed, \( S_{n,15} \) is zero for each \( n \geq 4 \). As for \( S_{2,15} \): If \( wz = zw \) then \( S_{2,15} = 1 \), else \( S_{2,15} = 2 \).

**Proof of (2a).** WLOGenity, \( C \) is finite. ( Exer. E4 )

With \( \Gamma := |G| \), our goal is

\[
2a' : \quad \sum_{v \in C} 1/\Gamma^{\text{Len}(v)} \overset{?}{\leq} 1.
\]

WELOG, suppose the shortest and longest words in \( C \) have lengths 3 and 7. For \( n = 1, 2, \ldots, \) each string in \( C^n \) has a length, \( \ell \), in \([3n, 7n]\); let \( S_{n,\ell} \) be the number of such strings. Certainly \( S_{n,\ell} \overset{?}{\leq} \Gamma^\ell \), the number of all length-\( \ell \) strings over \( G \). So the "generating function"

\[
F_n(x) := \sum_{\ell=3n}^{7n} [S_{n,\ell} \cdot x^\ell]
\]

satisfies, for \( x > 0 \), that \( F_n(x) \leq \sum_{\ell=3n}^{7n} \Gamma^\ell \cdot x^\ell \). Thus

\[
*: \quad F_n\left(\frac{1}{7}x\right) \leq \sum_{\ell=3n}^{7n} \Gamma^\ell \cdot x^\ell \overset{\text{note}}{=} 1 + 7n - 3n \overset{\text{note}}{\leq} 5n,
\]

for each posint \( n \).
**Using uniqueness.** Fix $n$ and an $\ell \in [3n..7n]$. The coefficient of $x^\ell$ in $[ F_1(x) ]^n$ is the number of $C$-n-parsings of length-$\ell$ strings, whereas $S_{n,\ell}$ is the number of length-$\ell$ strings which admit a $C$-n-parsing.

The UD-hypothesis [actually, only “weakly-UD” is being used] says these two numbers are equal. Hence our two polynomials are equal,

$$[ F_1(x) ]^n = F_n(x).$$

So ($\ast$) implies

$$[ F_1(\frac{1}{2}) ]^n \leq 5n.$$  

The LhS is exponential in $n$, whilst the RhS is linear. Thus $F_1(\frac{1}{2}) \leq 1$. Finally, observe that $F_1(\frac{1}{2})$ is a rewriting of LhS(2a).

### Proof of (2b).
We’ll show the idea for $\Gamma = 2$. Arrange the lengths as $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_R$. On the full binary-tree of depth $D : = \ell_R$, put weight $1/2^D$ on each leaf-node. All the nodes start as free; we will iteratively mark some as busy as we create words $v_1, v_2, \ldots$. Call a node very-free if it and all its children are free, i.e not busy.

Let $v_1$ be the leftmost path down to depth $\ell_1$; so $v_1 = 000..9.0$. Mark $v_1$ and all its children as busy. This action creates busy leaf-nodes of total weight,

$$2^{D-\ell_1} \cdot \frac{1}{2^D} \text{ note } 1/2^{\ell_1}.$$  

With $d := \ell_1$, note that

\[ \ast: \quad \text{Each free node at depth } \geq d \text{ is very-free.} \]

Let $v_2$ be the leftmost path to a free node at depth $\ell_2$. [So $v_2$ has $\ell_2 - 1$ many 0s, then a 1, then $\ell_2 - 1$ many 0s.] Mark $v_2$ and its descendants as busy. Now the total weight of busy leaf-nodes is

$$\frac{1}{2^{\ell_1}} + \frac{1}{2^{\ell_2}}.$$  

Moreover, with $d := \ell_2$, note ($\ast$) holds, since $\ell_2 \geq \ell_1$. We’d like to continue using depth $\ell_3$, depth $\ell_4$, ..., depth $\ell_k$, ... The only obstruction at a stage $k$, is if there is no free node at depth $\ell_k$. But the total leaf-weight we’ve used up so far, is

$$W := \sum_{j=1}^{k-1} \frac{1}{2^{\ell_j}}.$$  

Since this sum is strictly less than 1, there exists a free-node at depth $\ell_{k-1}$. (Indeed, the number of such free-nodes is precisely $[1 - W]/2^{k-1}$.). Finally, since $\ell_k \geq \ell_{k-1}$, there is certainly a free-node at depth $\ell_k$. ♦

### 2c: Defn.
For a $\Gamma$-code with lengths $\vec{\ell} = (\ell_1, \ldots, \ell_R)$, use

$$\Sigma(\vec{\ell}) := \Sigma_\Gamma(\vec{\ell}) := \sum_{j=1}^{R} 1/\Gamma^{\ell_j}$$

for its Kraft-sum. Kraft’s thm says –if the code is UD– that $\Sigma(\vec{\ell}) \leq 1$. If equality, then the code [ditto the tuple] is complete, otherwise it is redundant; more precisely, $\Gamma$-complete and $\Gamma$-redundant.

Given tuples $\vec{\ell} = (\ell_1, \ldots, \ell_N)$ and $\vec{s} = (s_1, \ldots, s_R)$, write $\vec{\ell} \lessdot \vec{s}$ if $N=R$ and $\forall j: \ell_j \leq s_j$. Write $\vec{\ell} < \vec{s}$ if $\vec{\ell} \lessdot \vec{s}$ yet $\vec{\ell} \neq \vec{s}$. [Ditto for $\infty$ tuples.] Note $\vec{\ell} \lessdot \vec{s}$ implies $\Sigma(\vec{\ell}) \geq \Sigma(\vec{s})$.

### Exer. E5.
A finite $\Gamma$-bounded tree $T$ with $R$ many leaves, yields a length-spectrum $\vec{\ell} = (\ell_1, \ldots, \ell_R)$; so terms “$\Gamma$-complete” and “$\Gamma$-redundant” makes sense for the tree. Prove:

2d: Completeness Lemma. A finite $\Gamma$-bounded tree, $T$, is $\Gamma$-complete IFF it is $\Gamma$-full. ♦

In (2c), below, we first consider only binary prefix-codes; $\Gamma = 2$.

### 2e: K-M Completeness corollary.
If finite tuple $\vec{s}$ has $\Sigma(\vec{s}) \leq 1$, then there exists a complete prefix-code with tuple $\vec{\ell} \lessdot \vec{s}$. ♦

### Proof.
We need but produce a complete $\vec{\ell} \lessdot \vec{s}$, since Kraft’s thm will hand us a prefix-code with lengths $\vec{\ell}$.

It suffices, given a redundant $\vec{s}$, to produce an $\vec{\ell} < \vec{s}$ with $\Sigma(\vec{\ell}) \leq 1$. After all, there are only finitely-many tuples $\prec \vec{s}$, so iterating will eventually halt, at a complete tuple.

WLOG, $T := s_1$ is a max-length in $\vec{s}$, so each $1/2^{s_j}$ is a multiple of $1/2^T$, hence so is $\Sigma(\vec{s})$. As $\vec{s}$ is redundant, the gap $1 - \Sigma(\vec{s})$ dominates $1/2^T$. So define $\vec{\ell}$ by $\ell_2 := s_2, \ldots, \ell_R := s_R$, and $\ell_1 := s_1 - 1$. ♦

### Exer. E6. POSTING RACE: Does (2e) hold for larger alphabet-sizes? If so, how does the proof need to be modified?
Exer. E7. Posting race: A block code is an example of a prefix/suffix-code, i.e., both. (Dis)Prove: There exists a complete prefix/suffix-code \( C \) whose length-spectrum is not constant.

Sardinas-Patterson Algorithm. An example of a UD-code [indeed, it is a suffixcode], for which the SarPat algorithm eventually cycles (as it must), but not with the empty prefix-list, is

\[ \{bc, b, Xc, cX\}. \]

(On hold…)

Decoding-delay for UD-codes. Consider a long word \( w \) which is the initial part…

(On hold…)

Cryptography

Affine codes. Breaking affine codes with known/chosen plaintext.

Diffie-Hellman and El Gamal.

RSA. Pollard-\( \rho \) algorithm and Floyd cycle-finding alg.
Data compression


Expected coding-length

The binary numeral for posint \( K \) has form \( 1 \text{Bits}(K) \), where \( \text{Bits}(K) \) is a \( \{0, 1\} \)-word. E.g., \( \text{Bits}(23) = 0111 \) because Binary(23) = 10111. Also \( \text{Bits}(3) = 1 \) and \( \text{Bits}(2) = 0 \) and \( \text{Bits}(1) = \varepsilon \), the nullword. Let

\[
|K|_{\text{Bit}} := |\text{Bits}(K)|.
\]

So \( |23|_{\text{Bit}} = 4 \), \( |2|_{\text{Bit}} = 1 \) and \( |1|_{\text{Bit}} = 0 \).

With \( n := |K|_{\text{Bit}} \), then, \( 2^{n+1} > K \geq 2^n \).

**Exer. E8.** Posting race: **Produce an infinite prefix-code**

\( C = \{v_1, v_2, v_3, \ldots \} \) such that \( \lim_{K \to \infty} |v_K|/|K|_{\text{Bit}} = 1. \)

\[\Box\]

**Elias-delta.** Set \( n := |K|_{\text{Bit}} \), and \( b := |n+1|_{\text{Bit}} \). Define codeword

\[ v_K := 0^b 1 \text{Bits}(n+1) \text{Bits}(K) \; . \]

So \( |v_K| = b + 1 + b + n \approx 2\log_2(n) + n. \)

**Example.** For \( K := 75 \), note \( \text{Bits}(75) = 001011 \); so \( |75|_{\text{Bit}} = 6 \). Now \( \text{Bits}(6+1) = 11 \), so \( |7|_{\text{Bit}} = 2 \). Thus

\[ v_{75} = \underbrace{00}_{7} \underbrace{11}_{75} \underbrace{001011}_{75} \; . \]

\[\Box\]

**Exer. E8.1.** Infinite prefix-code \( C = \{w_1, w_2, \ldots \} \) has the property that each

\[ |w_K| \leq |K|_{\text{Bit}} + f(|K|_{\text{Bit}}), \]

where \( f: \mathbb{Z}_+ \to \mathbb{N} \). Prove that \( \lim_{n \to \infty} f(n) = \infty \), using that \( C \) satisfies the Kraft inequality.

\[\Box\]

**Being Krafty.** An \( n \) has \( 2^n \) values of \( K \) with \( 2^n \leq K < 2^{n+1} \); each \( K \) has \( |w_K| \leq n + f(n) \). Summing over these \( K \),

\[
\sum_K \frac{1}{2|w_K|} \geq \sum_K \frac{1}{2^{n+f(n)}} = 2^n \cdot \frac{1}{2^{n+f(n)}} = \frac{1}{2f(n)}.
\]

But now we facepalm, because

\[
\uparrow: \quad 1 \geq \sum_{K=1}^{\infty} \frac{1}{2|w_K|} \geq \sum_{n=1}^{\infty} \frac{1}{2f(n)}. \]

And Rhs would be \( \infty \), were \( f \) bnded-above on a sub-seq.

**Aside:** With \( \mathcal{L} := \log_2 \), Elias-code has \( f(n) \approx 1 + 2\mathcal{L}(n) \). So \( 2f(n) \approx 2n^2 \). Thus

\[
\text{Rhs} \left( \uparrow \right) \approx \frac{1}{2} \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{1}{2} \cdot \frac{\pi^2}{6} \approx 0.82;
\]

consistent with Kraft. In contrast, there is no code with \( f(n) \leq \text{Const} + \mathcal{L}(n) \), since \( \sum_{n=1}^{\infty} \frac{1}{n} \) is infinite.

**Exer. E8.2.** (Dis)Prove: \( \exists \) prefix code \( C = \{w_1, w_2, \ldots \} \) satisfying \( (\uparrow) \), with \( f(n) \leq \text{Const} + [1.007] \cdot \log(n) \).

\[\Box\]

**Exer. E8.3.** (Dis)Prove: \( \exists \) prefix-code \( \{w_1, w_2, \ldots \} \) with \( \lim_{K \to \infty} |w_K|/|K|_{\text{Bit}} = 1 \) and subseq \( K_1 < K_2 < \ldots \) with each \( |w_{K_i}| \leq |K_i|_{\text{Bit}} + 99. \)

\[\Box\]
**Probability distr.** A probability distribution on a codeword-set $\mathcal{C}$ is a map $P: \mathcal{C} \to [0,1]$ st.

$$\sum_{v \in \mathcal{C}} P(v) = 1.$$  

We will usually discard from the code all probability-zero words. In practice, then, a “probability distribution” is a map $P: \mathcal{C} \to (0,1)$ fulfilling (3a)

The expected\(^{2a}\) coding length of $\mathcal{C}$ is

$$ECL(\mathcal{C}) := \sum_{v \in \mathcal{C}} P(v) \cdot \text{Len}(v).$$

E.g., consider code $\mathcal{C} := \{w_1, \ldots, w_4\}$ where

$$w_1 := 00, \ w_2 := 010, \ w_3 := 011, \ w_4 := 1,$$

where $P(w_4) = \frac{1}{2}$, and the other three words have probability $\frac{1}{6}$. Then $ECL(\mathcal{C})$ is then

$$\frac{1}{2} \cdot 1 + \frac{1}{6} \cdot (2 + 3 + 3) = \frac{11}{6}.$$

**Codemap.** A source alphabet $\Omega$, also called a “message set”, might be

$$\{a, b, \ldots, z, \text{ Space}\},$$

or might be $\{\text{ tank, ship, \ldots, plane}\}$. Fixing a code-alphabet $G$, a map $f: \Omega \to G^+$ is a codemap (or cipher) if

**i:** $f$ is injective, and

**ii:** $\mathcal{C} := \text{Range}(f)$ is a code. [Phrased this way, so that if we change our defn of “code” for a given context, then the defn of codemap changes with it.]

Every adjective applying to a code, also applies to a codemap; e.g., a block/prefix/UD codemap.

**ECL.** Consider a [finite or countably-infinite] message set $\Omega$ and a probability distribution $P: \Omega \to [0,1]$. A codemap $f: \Omega \to G^+$ puts a probability-distribution on $\mathcal{C} := \text{Range}(f)$ by assigning, for $w \in \mathcal{C}$,

$$P(w) := P(f^{-1}(w)).$$

Thus the code has an expected coding-length, which we may write as

$$ECL(\mathcal{C}) \text{ or } ECL(f).$$

**MECL.** Use $\text{MECL}$ for Minimum ECL. Consider a finite prob-vector $\vec{p} = (p_1, \ldots, p_L)$. A code $\mathcal{C}$ for the moment, assume a binary code $\mathcal{C} = \{v_1, \ldots, v_L\}$ has

$$\mathcal{ECL}(\mathcal{C}) = \sum_{j=1}^{L} p_j \cdot \text{Len}(v_j).$$

The minimum of $(3b')$ taken over all prefix-codes, or over all UD-codes, we will call

$$\text{PC-MECL}(\vec{p}) \text{ and } \text{UD-MECL}(\vec{p}),$$

respectively. Evidently

$$\text{PC-MECL}(\vec{p}) \geq \text{UD-MECL}(\vec{p})$$

since, for UD-codes, we are taking a minimum over the larger collection of codes. By the way, I'll sometimes use MECL($\vec{p}$) as a synonym for UD-MECL($\vec{p}$).

The minimum in $(3b')$ depends on $\Gamma := |\Gamma|$, the number of letters in our code alphabet. [We can compress English more by coding into a 3-letter alphabet, rather than a 2-letter alphabet.] To indicate the dependency on cardinality $\Gamma$, we may write

$$\text{PC-MECL}_{\Gamma}(\vec{p}) \text{ and } \text{UD-MECL}_{\Gamma}(\vec{p}).$$

**Huffman codes**

(Binary HCs will be described in class.)

Interpret a tuple such as $(3: A \ 1:B \ 5:C)$ as putting prob-distribution $(\frac{3}{9}, \frac{1}{9}, \frac{5}{9})$ on letters $(A, B, C)$; the 9 is the sum of the weights, $3 + 1 + 5$.

Our convention is that the branch going up-right is labeled with bit 0, and the down-right with bit 1.

**Non-uniqueness of Huffman Codes.** Frequency-tuple $F := (1:A \ 1:B \ 1:C \ 1:D)$ admits HC

$$\begin{array}{ll}
4 & \downarrow 2 \leftarrow 1 \downarrow 4 \leftarrow 2 \leftarrow 1 \\
& 1 - A: \ 00 \\
& 1 - B: \ 01 \\
& 1 - C: \ 10 \\
& 1 - D: \ 11
\end{array}$$

But $F$ also admits each other permutation of $\{A, B, C, D\}$ being attached to those leaves. So this Freq-tuple admits several HCs.
For a more interesting example, consider Frequency-tuple \( F' := (1:A; 1:B; 2:C; 2:D; 14:E) \). This admits HC \( C_1 \):

\[
\begin{array}{c}
5b: & 14 \quad E: 0 \\
 & 2 \quad D: 10 \\
 & 2 \quad C: 110 \\
 & 1 \quad B: 1110 \\
 & 1 \quad A: 1111 \\
\end{array}
\]

So \( 20 \cdot \text{ECL}(C_1) \) equals \([\text{Weight} \cdot \text{WordLen} \cdot \text{Count}]\)

\[
\begin{array}{c}
\text{R.A} + 2 \cdot \text{C} + 2 \cdot \text{D} + 14 \cdot \text{E} \\
\text{1}\cdot \text{A} + \text{2}\cdot \text{C} + \text{2}\cdot \text{D} + \text{1}\cdot \text{A} + \text{1}\cdot \text{E} \\
\end{array}
= 32.
\]

Thus \( \text{ECL}(C_1) = \frac{32}{20} = \frac{8}{5} \) bits-per-letter.

Our \( F' \) also admits HC \( C_2 \):

\[
\begin{array}{c}
5c: & 14 \quad E: 0 \\
 & 2 \quad D: 100 \\
 & 2 \quad C: 101 \\
 & 1 \quad B: 110 \\
 & 1 \quad A: 111 \\
\end{array}
\]

Thus \( 20 \cdot \text{ECL}(C_2) \) equals

\[
\begin{array}{c}
\text{R.A} + 2 \cdot \text{C} + 2 \cdot \text{D} + 14 \cdot \text{E} \\
\text{1}\cdot \text{A} + \text{2}\cdot \text{C} + \text{2}\cdot \text{D} + \text{1}\cdot \text{A} + \text{1}\cdot \text{E} \\
\end{array}
= 32.
\]

We see that \( \text{ECL}(C_2) = \text{ECL}(C_1) \). It is worth noticing that codes \( C_1 \) and \( C_2 \) are not only different, they are not even tree-isomorphic. \( \square \)

**Induction step.** Fix an \( L \geq 3 \) st. \( R(L-1) \).

Let \( J := L-2 \). Given \( \mathbf{p} \), let \( \alpha, \beta \) denote its two lowest probabilities.\(^\dagger\) and write \( \mathbf{p} \) as \((\alpha, \beta, p_1, \ldots, p_J)\).

Consider two HCs, \( C \) and \( X \), with length-spectra that I have written above and below \( \mathbf{p} \) here.

\[
\begin{align*}
C &: \quad D \quad D \quad d_1 \quad d_2 \quad \ldots \quad d_J \\
(\alpha, \beta, p_1, p_2, \ldots, p_J) \\
X &: \quad Y \quad y_1 \quad y_2 \quad \ldots \quad y_J.
\end{align*}
\]

So code \( C \) assigns length-\( D \) codewords to the first two nodes it joins, which have probs \( \alpha \) and \( \beta \). Computing

\[
\begin{align*}
\text{ECL}(C) &= D \cdot \alpha + D \cdot \beta + \sum_{i=1}^{J} [d_i \cdot p_i] \\
\text{ECL}(X) &= Y \cdot \alpha + Y \cdot \beta + \sum_{i=1}^{J} [y_i \cdot p_i].
\end{align*}
\]

After joining two nodes, the codes now recursively act on \( \mathbf{q} := (\alpha + \beta, p_1, p_2, \ldots, p_J) \) and assign length-spectra as follows:

\[
\begin{align*}
C &: \quad D-1 \quad d_1 \quad d_2 \quad \ldots \quad d_J \\
(\alpha+\beta, p_1, p_2, \ldots, p_J) \\
X &: \quad Y-1 \quad y_1 \quad y_2 \quad \ldots \quad y_J.
\end{align*}
\]

Since \( \mathbf{q} \) is an \( [L-1] \)-vector, proposition \( R(L-1) \) says that the above two ECLs are equal, i.e

\[
\begin{align*}
[D-1] \cdot [\alpha + \beta] + \sum_{i=1}^{J} [d_i \cdot p_i] \\
&= [Y-1] \cdot [\alpha + \beta] + \sum_{i=1}^{J} [y_i \cdot p_i].
\end{align*}
\]

And this implies equality in the two RhSs of \((\dagger)\). \( \blacklozenge \)

**7a: Depth Lemma.** Fix a probability \( L \)-vector \( \mathbf{p} \), and a \( \mathbf{p} \)-PC-MECL. Consider two leaf-nodes with probabilities \( \alpha \) and \( \alpha' \), at depths \( D \) and \( D' \), respectively. If \( \alpha > \alpha' \), then necessarily \( D \leq D' \). \( \checkmark \)

**Exer. E9.** Prove the above Depth Lemma. \( \square \)

**7b: Huffman’s theorem.**

\( i \): HCs are PC-MECLs.

\( ii \): HCs are UD-MECLs. \( \checkmark \)

\( \dagger \)They might be equal; indeed, perhaps \( \beta = \alpha \), with 8 nodes all having probability \( \alpha \). We are not picking two nodes; we are picking two **probabilities**. In particular, I am not assuming that HCs \( C \) and \( X \) join the same two nodes, at the first step.
Pf of (i). We induct on $L$, with proposition

\[ \text{HUFF}(L): \text{ Each probability } L\text{-vector } \vec{q}, \text{ admits a Huffman Code which is a PC-MECL.} \]

The base $L=2$ case is immediate, since the only tree is Root \(\frac{\text{Prob.}}{\text{Prob.}}\), which is a Huffman-tree.

**Induction step.** Fix an $L \geq 3$ st. \text{HUFF}(L−1). Fix $\vec{p}$, a prob. $L$-vector, and consider a $\vec{p}$–PC-MECL, viewed as a tree.

Let $\alpha \leq \beta$ denote the two smallest probabilities of $\vec{p}$. At the tree’s deepest level, $D$, consider two joined leaf-nodes, and call their probabilities $x$ and $y$. It suffices to show:

We can permute the probabilities of the leaves, \*:

* without changing the ECL, so that, now, these two nodes have probabilities $\alpha$ and $\beta$.

For then, we collapse these two into a single node, producing prob.-vec $\vec{q} := (\alpha+\beta, p_2, p_3, \ldots, p_{L-1})$. By the induction hypothesis, there is a $\vec{q}$-HC which is a $\vec{q}$-PC-MECL. Expanding the collapsed node back into $\leq \alpha \beta$ automatically produces a Huffman-tree which is a $\vec{p}$-PC-MECL. And all HCs have the same ECL, by (6).

**Establishing (\*)**. If $x = \alpha$, then leave that leaf-node alone. Otherwise, $x > \alpha$. Now the Depth Lemma, (7a), says that $\alpha$ can’t be shallower than $x$, so [since $x$ is at max depth], every $\alpha$-node has to be at $D$, the deepest level. Switch some $\alpha$-leaf with our $x$-leaf.

This does not change the ECL, since the nodes are at the same depth.

Now our joined-pair is $\leq \alpha \beta$. Do the same operation with $y$ w.r.t $\beta$. Now our joined-pair is $\leq \alpha \beta$, as desired.

**Exer. E10.** Prove (ii), that every HC is a UD-MECL.

\[ \text{Pf of (ii), (E10). Fix } \vec{p} \text{ and a } \vec{p}-\text{UD-MECL; write its length-spectrum as } \ell = (\ell_1, \ldots, \ell_R). \text{ By Kraft’s thm, there is a PC-code with the same spectrum hence, when assigned to the same probabilities, has the same ECL. And part (i) shows there is a HC with the same ECL.} \]

\[ \text{Exer. E11. Posting race: (Dis)Prove: If prefix code } C \text{ is a PC-MECL, then } C \text{ is a Huffman code.} \]

**Solution to E11.** False. Consider frequency-tuple (\(2;A 2;B 3;C 3;D\)). Its only Huffman-tree is

\[ 7c: \]

\[ 10 \]

\[ 4 \]

\[ 2 - A \]

\[ 2 - B \]

\[ 6 \]

\[ 3 - C \]

\[ 3 - D \]

(This admits eight HCs, since at each of the three nodes we can choose which edge is labeled 0 and which is 1.) This codetree has ECL = 2. But so does this tree,

\[ 7d: \]

\[ 10 \]

\[ 5 \]

\[ 2 - A \]

\[ 2 - B \]

\[ 5 \]

\[ 3 - C \]

\[ 3 - D \]

which is not a Huffman code.

\[ \text{ Exer. E11. False. Consider frequency-tuple (2;A 2;B 3;C 3;D). Its only Huffman-tree is} \]

\[ \text{This admits eight HCs, since at each of the three nodes we can choose which edge is labeled 0 and which is 1.) This codetree has ECL = 2. But so does this tree,} \]

\[ \text{which is not a Huffman code.} \]

\[ \text{Exer. E10. Prove (ii), that every HC is a UD-MECL.} \]

\[ \text{Exer. E11. Posting race: (Dis)Prove: If prefix code } C \text{ is a PC-MECL, then } C \text{ is a Huffman code.} \]

\[ \text{Solution to E11. False. Consider frequency-tuple (2;A 2;B 3;C 3;D). Its only Huffman-tree is} \]

\[ \text{This admits eight HCs, since at each of the three nodes we can choose which edge is labeled 0 and which is 1.) This codetree has ECL = 2. But so does this tree,} \]

\[ \text{which is not a Huffman code.} \]

\[ \text{Exer. E11. Posting race: (Dis)Prove: If prefix code } C \text{ is a PC-MECL, then } C \text{ is a Huffman code.} \]
Entropy/Distropy

Define \( \eta : [0, 1] \to [0, \infty) \) by \( \eta(x) := x \cdot \log_2(1/x) \), and extend by continuity, so that \( \eta(0) = 0 \). (Use l’Hôpital’s rule, if you like.)

The distribution entropy, which I call distropy, of a probability-vector \( \vec{v} \) is

\[ H(\vec{v}) := \sum_{p \in \vec{v}} \eta(p). \]

For a probability-distribution \( P() \) on a code \( C \), then, \( H(P) \) equals \( \sum_{v \in C} \eta(P(v)) \).

8: Distropy UD-code Inequality. Fix a binary code \( C \) and probability distribution \( P: C \to (0, 1) \). If \( C \) is uniquely decodable, then

8a: \( \text{ECL}(C) \geq H(P) \).

There is equality in (8a) IFF

8b: \( \forall v \in C: P(v) = 1/2^{\hat{v}} \).

\( Pf \) of (8b). Suppose \( \text{ECL}(C) = H(P) \). This forces equality in Kraft, so \( \sum_{v} 1/2^{\hat{v}} = 1 \), and in Jensen’s, so the map \( v \mapsto \frac{1}{P(v)} \cdot \frac{1}{2^{\hat{v}}} \) is constant; say \( \kappa \). Thus \( P(v) \cdot \kappa = 1/2^{\hat{v}} \), for each \( v \). Summing over all \( v \in C \) implies that \( 1 \cdot \kappa = 1 \). Hence \( \kappa = 1 \).

For comparison with (binary) distropy/entropy, we will usually be examining a binary code; a code over a 2-symbol alphabet, \( \mathbb{B} \). (Typically, \( \mathbb{B} = \{0, 1\} \).) So a binary code is a subset \( C \subset \mathbb{B}^* \).

\( Pf \) of (8a). Let “\( \sum_v \)” mean “\( \sum_{v \in C} \)” and \( \hat{v} \) mean \( \text{Len}(v) \).

With \( L() := \log_2() \), note \( \text{ECL}(C) \) equals \( \sum_{v} P(v)\hat{v} \), which equals \( \sum_{v} P(v)L(2^{\hat{v}}) \). Consequently, we can write \( H(P) - \text{ECL}(C) \) as

\[
\left[ \sum_{v} P(v)L\left( \frac{1}{P(v)} \right) \right] - \left[ \sum_{v} P(v)L(2^{\hat{v}}) \right] = \sum_{v} P(v)L\left( \frac{1}{P(v)} \cdot \frac{1}{2^{\hat{v}}} \right).
\]

Since \( L() \) is strictly convex-down, Jensen’s inequality, (12), applies to say

\[
H(P) - \text{ECL}(C) \leq L\left( \sum_{v} P(v) \cdot \frac{1}{P(v)} \cdot \frac{1}{2^{\hat{v}}} \right)
\]

\[ \overset{\dagger}{=} L\left( \sum_{v} 1/2^{\hat{v}} \right). \]

By (2a) the Kraft-McMillan inequality, \( \sum_{v} 1/2^{\hat{v}} \leq 1 \). And \( L() \) is order-preserving. Thus the above yields

\[
H(P) - \text{ECL}(C) \leq L(1) = 0,
\]

as desired.

\( \overset{\dagger}{=} \) For comparison with (binary) distropy/entropy, we will usually be examining a binary code; a code over a 2-symbol alphabet, \( \mathbb{B} \). (Typically, \( \mathbb{B} = \{0, 1\} \).) So a binary code is a subset \( C \subset \mathbb{B}^* \).
**Convection.** For $p \in [0,1]$, let $p^c$ mean $1-p$, in analogy with $P(B^c)$ equaling $1-P(B)$ on a probability space. [See Appendix for independence, $\perp$, defns.]

9: **Distropy fact.** For partitions $P, Q, R$ on probability space.

9a: $\mathcal{H}(P) \leq \log(#P)$, with equality $\text{iff}$ $P$ is an equi-mass partition.

b: $\mathcal{H}(Q \lor R) \leq \mathcal{H}(Q) + \mathcal{H}(R)$, with equality $\text{iff}$ $Q \perp R$.

c: For $p \in [0,\frac{1}{2}]$, the function $p \mapsto \mathcal{H}(p, p^c)$ is strictly increasing. \hfill \blacklozenge

**Proof.** Use the strict concavity of $\eta()$, together with Jensen’s Inequality. \hfill ∆

10: **Binomial Lem.** Fix $p \in [0, \frac{1}{2}]$ and let $H := \mathcal{H}(p, p^c)$. Then for each $n \in \mathbb{Z}_+$:

10a: $\sum_{j \in [0..m]} \binom{n}{j} \leq 2^{Hn}$. \hfill ∆

**Proof.** Let $X \subset \{0,1\}^n$ be the set of $x$ with $\#\{i \in [1..n] \mid x_i = 1\} \leq p \cdot n$. On $X$, let $P_1, P_2, \ldots$ be the coordinate partitions; e.g. $P_7 = (A_7, A_7^c)$, where $A_7 := \{x \mid x_7 = 1\}$. Weighting each point by $\frac{1}{|X|}$, the uniform distribution $\mu()$ on $X$, gives that $\mu(A_7) \leq p$. So $\mathcal{H}(P_7) \leq H$, by (9c). Finally, the join $P_1 \lor \ldots \lor P_n$ separates the points of $X$. So

$$
\log(\#X) = \mathcal{H}(P_1 \lor \ldots \lor P_n) \\
\leq \mathcal{H}(P_1) + \ldots + \mathcal{H}(P_n) \leq Hn,
$$

making use of (9a,b). And $\#X$ equals LHS(10a). \hfill ∆

**Note:** Below, several quantities need to be natnums, and so some floor or ceiling symbols are needed. I have omitted them, to show the overall idea of the proof.

11: **Shannon source-coding thm.** Fix probability $p \in (0,1)$, and set $H := \mathcal{H}(p, p^c)$. Fix $\varepsilon > 0$. Then for large $N$, there exists a block-code, mapping

$$N \text{ bits } \rightarrow [H + \varepsilon] \cdot N \text{ bits},$$

with error-probability $< \varepsilon$. \hfill ∆
Error-correcting codes

Hamming codes, distance, weight, bound.

Shannon’s Noisy-channel Thm . . .
\section{Appendix}

Various general tools.

\textbf{12: Jensen’s inequality.} On an interval \( J \subset \mathbb{R} \), consider points \( Q_v \in J \), for each \( v \) in a countable indexing-set \( C \). We have a probability-distr \( P() \) on \( C \). Then for each convex-down fnc \( \mathcal{L} : J \to \mathbb{R} \)

\[ \mathcal{L} \left( \sum_{v \in C} P(v) \cdot Q_v \right) \geq \sum_{v \in C} P(v) \cdot \mathcal{L}(Q_v) . \]

Now suppose \( \mathcal{L} \) is strictly convex-down. Then:

\textbf{12b:} Equality in (12a) \textit{iff} the probability-distr is concentrated on a single point.

\textbf{IOWords}, having removed all zero-probability elements from \( C \), the map \( v \mapsto Q_v \) is constant.

\textbf{Proof.} Exercise. [Or see picture on blackboard.]

\section*{Probability}

A \textit{random variable} \([r.var]\) is a measurable map \( Y : \Omega \to \mathbb{R} \) where \( \Omega \) is a probability space. \([\text{Can take } \Omega \text{ to be } [0,1].]\) Unless both the positive and negative parts of \( Y \) have infinite integral, the “\textit{expectation} of \( Y \)”, \( E(Y) := \int_\Omega Y \), is a value in \([-\infty, +\infty] \).

When finite, it is common to call \( \mu := E(Y) \) the \textit{mean} of \( Y \). Then \textit{variance} \( \text{Var}(Y) := E((Y - \mu)^2) \) is well-defined, and could be \( +\infty \).

\textbf{Independence.} Events \( A, B \) are \textit{independent}, written \( A \perp B \), if \( P(A \cap B) = P(A)P(B) \). A \textit{family} \( \mathcal{C} \) of events is independent, written \( \perp (\mathcal{C}) \) or \( \perp (\{A\}_{A \in \mathcal{C}}) \), if each finite subset \( A_1, \ldots, A_N \) has \( P(A_1 \cap \ldots \cap A_N) \) equalling \( \prod_{j=1}^N P(A_j) \). This property of \( \mathcal{C} \) is much stronger than \textit{pairwise independence}, where each pair of events in \( \mathcal{C} \) is independent.

Random variables \( X, Y \) are \textit{independent}, \( X \perp Y \), if for each pair of measurable sets \( S,T \subset \mathbb{R} \), events \( \{X \in S\} \) and \( \{Y \in T\} \) are independent. It turns out that this is equivalent to saying, for each pair \( x,y \in \mathbb{R} \), that events \( \{X \leq x\} \perp \{Y \leq y\} \). When \( X \perp Y \) have finite expectations, then \( E(X \cdot Y) = E(X) \cdot E(Y) \).

Extend notions of \textit{independence} and \textit{pairwise independence} to collections of random variables.

\textbf{13a: Markov Lemma.} Consider posint \( n \) and random variable \( Y \). For each \( \varepsilon \in \mathbb{R}_+ \):

\[ P(|Y| \geq \varepsilon) \leq \frac{E(|Y|^n)}{\varepsilon^n} ; \quad \text{Markov Inequality.} \]

When \( n \) is even,

\[ P(|Y| \geq \varepsilon) \leq \frac{E(Y^n)}{\varepsilon^n} . \]

In particular, if \( Y \) has finite mean \( \mu := E(Y) \), then

\[ P(|Y - \mu| \geq \varepsilon) \leq \frac{\text{Var}(Y)}{\varepsilon^2} ; \quad \text{Chebyshev Inequality.} \]

\textbf{Proof.} Exercise.

\textbf{13b: Weak Law of Large Numbers (WLLN).} Consider an identically-distributed pairwise-independent sequence \( X_1, X_2, \ldots \) where both mean \( \mu := E(X) \) and variance \( \nu := \text{Var}(X) \) are finite. Then

\[ \lim_{N \to \infty} P \left( \left| \frac{\sum_{j=1}^N X_j}{N} - \mu \right| \geq \varepsilon \right) = 0 , \]

where \( \bar{X}_N := \frac{1}{N} \sum_{j=1}^N X_j \).

\textbf{Proof.} WLOG \( \mu = 0 \). Then \( N^2 \cdot \text{Var}(\bar{X}_N) \) equals

\[ E \left( \left( \sum_{j=1}^N X_j \right)^2 \right) = \sum_{i=1}^N E(X_i^2) + \sum_{j \neq k} E(X_j X_k) = N \nu + \sum_{j \neq k} E(X_j) \cdot E(X_k) = N \nu , \]

since each \( E(X_j) = 0 \). Thus \( \text{Var}(\bar{X}_N) = \frac{\nu}{N} \). Hence

\[ P \left( \left| \bar{X}_N \right| \geq \varepsilon \right) \leq \frac{\text{Var}(\bar{X}_N)}{\varepsilon^2} = \frac{1}{N} \frac{\nu}{\varepsilon^2} , \]

by the Chebyshev Inequality.
### §Index for “JK Codes notes”

<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Posting race</td>
<td>3–6, 9</td>
</tr>
<tr>
<td>prefix code</td>
<td>2</td>
</tr>
<tr>
<td>prefix/suffix-code</td>
<td>5</td>
</tr>
<tr>
<td>probability distribution</td>
<td>7</td>
</tr>
<tr>
<td>redundant code</td>
<td>4</td>
</tr>
<tr>
<td>RI-UD property</td>
<td>2</td>
</tr>
<tr>
<td>source alphabet</td>
<td>7</td>
</tr>
<tr>
<td>spectrum</td>
<td>4, 9</td>
</tr>
<tr>
<td>suffix code</td>
<td>2</td>
</tr>
<tr>
<td>tree</td>
<td>3</td>
</tr>
<tr>
<td>( \Gamma )-complete/redundant</td>
<td>4</td>
</tr>
<tr>
<td>( \Gamma )-full/deficient</td>
<td>3</td>
</tr>
<tr>
<td>( \Gamma )-bounded</td>
<td>3</td>
</tr>
<tr>
<td>isomorphism</td>
<td>3</td>
</tr>
<tr>
<td>surgery</td>
<td>9</td>
</tr>
<tr>
<td>UD, uniquely decodable</td>
<td>2</td>
</tr>
<tr>
<td>void language</td>
<td>1</td>
</tr>
<tr>
<td>weak-BI-UD</td>
<td>2</td>
</tr>
<tr>
<td>weakly-UD</td>
<td>3</td>
</tr>
<tr>
<td>word</td>
<td>1</td>
</tr>
</tbody>
</table>

This is a test of the pre-note.

C-parsing, 2
alphabet, 1
bi-infinite-UD property, 2
BI-UD, 2
block code (constant-length), 2
cipher, 7
code, 2
codemap, 7
complete code, 4
distribution entropy, 10
distropy, 10
entropy, 10
expected coding-length, ECL, 7
full tree, see tree
HC, Huffman code, 7
idempotent, 1
Kleene star/plus, 1
Kraft-sum, 4
language, 1
leaf-node, 3
left-infinite-UD-code, 2
LI-UD-code, 2
MECL, 7
nullishcode, 2
nullword, 1
nullword language, 1