Review of main concepts from MATH 474

- Finite source alphabet \( \mathcal{X} \), binary code alphabet \( \mathcal{D} = \{0, 1\} \), and \( \mathcal{D}^* = \{0, 1\}^* \) denotes the set of all finite-length binary strings.
- A variable-length source code is a mapping \( C : \mathcal{X} \to \mathcal{D}^* \). For \( x \in \mathcal{X} \), \( C(x) \) is the codeword for \( x \) having length \( l(x) = |C(x)| \).
- If \( X \) is a random variable with alphabet \( \mathcal{X} \) and pmf \( p(x) = P(X = x) \), the expected code length is
  \[
  L(C) = E l(X) = \sum_{x \in \mathcal{X}} p(x) l(x)
  \]
- \( C \) is nonsingular if it is injective as a mapping:
  \[ x \neq y \implies C(x) \neq C(y) \quad \text{for all } x, y \in \mathcal{X} \]

Example (Cover&Thomas): \( \mathcal{X} = \{1, 2, 3, 4\} \)

<table>
<thead>
<tr>
<th>( \mathcal{X} )</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
<th>( C_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>11</td>
<td>110</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

- \( C_1 \) is nonsingular
- \( C_2 \) is uniquely decodable
- \( C_3 \) is a prefix code
- If \( X \sim p(x) \) with \( p(1) = 1/2, p(2) = 1/4, p(3) = p(4) = 1/8 \), then
  \[
  L(C_3) = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = 1.75
  \]
Kraft’s inequality:

- For any uniquely decodable code \( C \), the codeword lengths \( l(x) = |C(x)| \) satisfy
  \[
  \sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1
  \]
- Conversely, if the nonnegative integers \( l(x), x \in \mathcal{X} \) satisfy Kraft’s inequality, then there exists a prefix code \( C \) on \( \mathcal{X} \) with codeword lengths \( |C(x)| = l(x) \).

We will only consider prefix codes since the (larger) class of uniquely decodable codes don’t offer any advantage in terms of their (expected) code lengths.

The entropy of a discrete random variable \( X \) on alphabet \( \mathcal{X} \) and pmf \( p(x) \) is
\[
H(p) = H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)
\]

Bounds on optimal code length

- For any prefix code \( C \)
  \[
  L(C) \geq H(X)
  \]
  and equality holds if and only if the pmf of \( X \) is such that \( p(x) = 2^{-l(x)} \) for all \( x \in \mathcal{X} \) (dyadic distribution).
- There exists a prefix code \( C \) such that
  \[
  L(C) < H(X) + 1
  \]

The proof of the lower bound is based on Kraft’s and Jensen’s inequalities.

One code that satisfies the upper bound is the Shannon-Fano code having codeword lengths
\[
l(x) = \lceil -\log p(x) \rceil
\]

Optimal lossless coding

- Let \( C \) be the collection of all (binary) prefix codes on \( \mathcal{X} \). The expected code length of an optimal code is
  \[
  L^* = \min_{C \in \mathcal{C}} L(C)
  \]
- The entropy lower bound and the Shannon-Fano code give
  \[
  H(X) \leq L^* < H(X) + 1
  \]
- Huffman’s algorithm yields an optimal code \( C^* \) such that \( L(C^*) = L^* \) (Huffman code). Gallager proved that
  \[
  L^* < H(X) + p_{\max} + c
  \]
  where \( p_{\max} = \max_x p(x) \) and \( c \approx 0.086 \). If \( p_{\max} \geq 1/2 \), then
  \[
  L^* < H(X) + p_{\max}
  \]

Coding blocks of source symbols

- Replace the alphabet \( \mathcal{X} \) with \( \mathcal{X}^n \) and the random variable \( X \) with the sequence \( X^n = (X_1, X_2, \ldots, X_n) \). Assume \( X^n \sim p(x^n) \).
- For a prefix code \( C \) on \( \mathcal{X}^n \) with codeword length \( l(x^n) = |C(x^n)| \) for \( x^n = (x_1, \ldots, x_n) \) define
  \[
  L_n(C) = \frac{1}{n} \mathbb{E} l(X^n) = \frac{1}{n} \sum_{x^n \in \mathcal{X}^n} p(x^n) l(x^n) \quad \text{(per symbol length)}
  \]
- The minimum per symbol expected codeword length for source \( X^n \) is
  \[
  L^*_n = \min_{C \in \mathcal{C}_n} L_n(C)
  \]
  where \( \mathcal{C}_n \) denotes the set of all prefix codes on \( \mathcal{X}^n \).
**Lossless source coding theorem**

- For any source $X^n$ the minimum expected codeword length per source symbol satisfies
  $$\frac{1}{n}H(X^n) \leq L_n^* \leq \frac{1}{n}H(X^n) + \frac{1}{n}$$

- If $X_1, X_2, \ldots$ is a stationary process with entropy rate $\bar{H}(X) = \lim_{n \to \infty} \frac{1}{n}H(X^n)$, then
  $$\lim_{n \to \infty} L_n^* = \bar{H}(X)$$

**Note:** If $X_1, X_2, \ldots$ is an i.i.d. source, then $\bar{H}(X) = H(X_1)$.

---

**Practical concerns**

- To build the code tree for the Shannon-Fano or Huffman codes, one has to generate all codewords $\{C(x^n) : x^n \in \mathcal{X}^n\}$.
- The size of the code tree ($\#$ of nodes $= 2|\mathcal{X}| + 1$) increases exponentially with the block length.
- For larger $n$ both the Huffman and the Shannon-Fano code (in the naive implementation) become impractical.
- For both codes one needs to look at the entire sequence $x^n$ before encoding. One would like to start generating the codeword sequentially as the source symbols $x_i$ are received.

---

**Shannon-Fano-Elias (arithmetic) coding**

- $\mathcal{X} = \{a_1, a_2, \ldots, a_m\}$ is ordered such that $a_1 < a_2 < \cdots < a_m$.
- Consider the induced *lexicographical ordering* on $\mathcal{X}^n$ for $n \geq 2$:
  $$y^n = (y_1, \ldots, y_n) < (x_1, \ldots, x_n) = x^n \text{ if and only if } y_1 < x_1$$
  or there exists $k \in \{2, \ldots, n\}$ such that
  $$y_k = x_k, \quad i = 1, \ldots, k - 1, \quad \text{and} \quad y_k < x_k.$$
- Let $p(x^n)$ be a pmf on $\mathcal{X}^n$ such that $p(x^n) > 0$, $\forall x^n \in \mathcal{X}^n$, and define
  $$\tilde{F}(x^n) = \sum_{y^n < x^n} p(y^n), \quad F(x^n) = \sum_{y^n \leq x^n} p(y^n)$$
Note that $0 \leq \hat{F}(x^n) < F(x^n) \leq 1$ for all $x^n$. Define the *tag* of $x^n$ by

$$\bar{T}(x^n) = \hat{F}(x^n) + \frac{1}{2} p(x^n)$$

so that $\bar{T}(x^n)$ is the midpoint of the interval $[\hat{F}(x^n), F(x^n)]$.

For $x \in [0, 1)$ we let $b(z) = b_1(z)b_2(z)b_3(z)\cdots$ be its infinite binary fraction representation: $b_i(z) \in \{0, 1\}$ and

$$z = \sum_{i=1}^{\infty} b_i(z)2^{-i}$$

Make representation *unique* by excluding representations where for some $i \geq 1$, $b_i(z) = 0$ and $b_j(z) = 1$ for all $j > i$.

Let $b^l(z)$ denote the binary representation of $z$ truncated to $l$ bits:

$$b^l(z) = b_1(z)\cdots b_l(z)$$

### Definition (Shannon-Fano-Elias code)

The Shannon-Fano-Elias (SFE) code $C$ on $X^n$ is defined by

$$C(x^n) = b^l(x^n)(\bar{T}(x^n))$$

where

$$l(x^n) = \lceil -\log p(x^n) \rceil + 1$$

That is, the codeword associated with $x^n$ is the binary representation of $\bar{T}(x^n)$ truncated to $\lceil -\log p(x^n) \rceil + 1$ bits.

### Theorem 1

The Shannon-Fano-Elias code is a prefix code with expected code length satisfying

$$L(C) < H(X^n) + 2$$

### Remark:

For the per symbol expected codeword length we have

$$L_n(C) \leq \frac{1}{n} H(X^n) + \frac{2}{n}$$

If $X_1, X_2, \ldots$ is a stationary source,

$$\lim_{n \to \infty} L_n(C) = \bar{H}(X).$$

(code is asymptotically optimal).

### Proof:

The bound on $L(C)$ is trivial since

$$|C(x^n)| = l(x^n) = \lceil -\log p(x^n) \rceil + 1 < -\log p(x^n) + 2$$

so

$$\sum_{x^n} p(x^n) l(x^n) < \sum_{x^n} p(x^n) \log p(x^n) + 2 = H(X^n) + 2$$

We have to prove that $C$ is a prefix code. Let $\lfloor z \rfloor$ be $z \in [0, 1)$ truncated to $l$ bits:

$$\lfloor z \rfloor = \sum_{i=1}^{l} b_i(z)2^{-i}$$

Note that

$$z - \lfloor z \rfloor = \sum_{i=l+1}^{\infty} b_i(z)2^{-i} < 2^{-l}$$

since representations with $b_i = 1$ for all $i \geq l + 1$ are excluded.
As described, the SFE code can be implemented as follows:

**Encoding**

1. Given $x^n = (x_1, \ldots, x_n)$, calculate
   \[
   \hat{T}(x^n) = \sum_{y^n < x^n} p(y^n) + \frac{p(x^n)}{2}
   \]

2. Let $l = \lceil -\log p(x^n) \rceil + 1$ and let $C(x^n)$ be the binary representation of $\hat{T}(x^n)$ truncated to $l$ bits.

**Decoding**

3. Given $C(x^n) = b_1 b_2 \cdots b_l$, compute $z = \sum_{i=1}^{l} b_i 2^{-i}$.

4. Find the unique $x^n$ such that $\hat{F}(x^n) < z < F(x^n)$.
   - This procedure does not require the storage of all codewords in advance.
   - The naive implementation of (1) and (4) requires computing sums with an exponential number of terms.

**Proof cont’d:** Thus

\[
\hat{T}(x^n) - \lceil \hat{T}(x^n) \rceil (x^n) < 2^{-l(x^n)} \tag{1}
\]

Since $l(x^n) = \lceil -\log p(x^n) \rceil + 1 \geq -\log p(x^n) + 1,$

\[
2^{-l(x^n)} \leq \frac{p(x^n)}{2} = \hat{T}(x^n) - F(x^n) \tag{2}
\]

(1) and (2) give

\[
[\hat{T}(x^n)]_{l(x^n)} > \hat{T}(x^n) - 2^{-l(x^n)}
\]
\[
\geq \hat{T}(x^n) - \frac{p(x^n)}{2}
\]
\[
= F(x^n)
\]

and (2) implies

\[
[\hat{T}(x^n)]_{l(x^n)} + 2^{-l(x^n)} \leq \hat{T}(x^n) + 2^{-l(x^n)}
\]
\[
\leq \hat{T}(x^n) + \frac{p(x^n)}{2}
\]
\[
= F(x^n)
\]

**Proof cont’d:** Define the interval

\[
I(x^n) = \left[\hat{T}(x^n)]_{l(x^n)}, \hat{T}(x^n)]_{l(x^n)} + 2^{-l(x^n)}\right)
\]

We just proved that $I(x^n) \subseteq [\hat{F}(x^n), F(x^n))$. Since $[\hat{F}(x^n), F(x^n))$ and $[\hat{F}(y^n), F(y^n))$ are disjoint if $x^n \neq y^n$,

\[
I(x^n) \cap I(y^n) \text{ if } x^n \neq y^n
\]

Let $l = l(x^n)$ and $[\hat{T}(x^n)]_l = 0.b_1 b_2 \cdots b_l$. Then

\[
I(x^n) = \left[0.b_1 b_2 \cdots b_l, 0.b_1 b_2 \cdots b_l + 2^{-l}\right)
\]

contains all numbers whose binary representation starts with $C(x^n) = b_1 b_2 \cdots b_l$. If $y^n \neq x^n$, then $\hat{T}(y^n) \notin I(x^n)$, so $C(x^n)$ cannot be the prefix of $C(y^n)$.

\[\square\]

**Example:**

\[\mathcal{X} = \{0,1\}, n = 2, \text{ source is i.i.d. with pmf } p(0) = 0.8, p(1) = 0.2.\]

<table>
<thead>
<tr>
<th>$x^n$</th>
<th>$p(x^n)$</th>
<th>$\hat{F}(x^n)$</th>
<th>$\hat{T}(x^n)$</th>
<th>$l(x^n)$</th>
<th>$C(x^n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0.64</td>
<td>0</td>
<td>0.32</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>0.16</td>
<td>0.64</td>
<td>0.72</td>
<td>4</td>
<td>1110</td>
</tr>
<tr>
<td>10</td>
<td>0.16</td>
<td>0.8</td>
<td>0.88</td>
<td>4</td>
<td>1110</td>
</tr>
<tr>
<td>11</td>
<td>0.04</td>
<td>0.96</td>
<td>0.98</td>
<td>6</td>
<td>111110</td>
</tr>
</tbody>
</table>

Some details:

\[
\hat{F}(10) = p(00) + p(01) = 0.8 \cdot 0.8 + 0.8 \cdot 0.2 = 0.8
\]
\[
\hat{T}(10) = 0.8 + \frac{1}{2} p(01) = 0.8 + \frac{0.16}{2} = 0.88 = 0.1100\ldots
\]
\[
l(10) = \lceil -\log p(01) \rceil + 1 = [2.643] + 1 = 4, \quad C(01) = 1110
\]

**Note:** $L_2(C) = \frac{2.8}{2} = 1.4$ and $H(X) = 0.72 \implies$ need to use larger $n$ for better efficiency.
Arithmetic coding implementation of SFE code (high-level description)

Let \( X_1, X_2, \ldots \) be a source with alphabet \( \mathcal{X} \). For any \( k \geq 1 \) let

\[
 p(x^k) = P(X^k = x^k) = P(X_1 = x_1, \ldots, X_k = x_k)
\]

The following hold for \( k \geq 2 \) and \( x^k \in \mathcal{X}^k \):

- \( p(x^{k-1}) = \sum_{y \in \mathcal{X}} p(x^{k-1}|y) \) (law of total probability)
- \( p(x_k|x^{k-1}) = P(X_k = x_k|X^{k-1} = x^{k-1}) = \frac{p(x^k)}{p(x^{k-1})} \)
- \( p(x^k) = \prod_{i=1}^k p(x_i|x^{i-1}) \) (product rule)

Here we used the convention that \( p(x_i|x^{i-1}) = p(x_i) \) if \( i = 1 \).

**Encoding**

For \( k \geq 2 \) we have

\[
 \hat{F}(x^k) = \sum_{y_k < x_k} p(y_k)
\]

\[
 = \sum_{y_{k-1} < x_{k-1}} \sum_{y_k \in \mathcal{X}} p(y_{k-1}|y_k) + \sum_{y_k < x_k} p(x_{k-1}|y_k)
\]

\[
 = \hat{F}(x^{k-1}) + p(x^{k-1}) \sum_{y_k < x_k} p(y_k|x^{k-1})
\]

Suppose the conditional probabilities \( p(y|x^k) \) are easy to compute. Then

- We can sequentially compute the intervals \( [\hat{F}(x^k), F(x^k)] \), \( k = 1, \ldots, n \).
- The number of additions in the encoding operation is reduced from \( O(|\mathcal{X}|^n) \) to \( O(n|\mathcal{X}|) \).

**Decoding**

- Note that

\[
 \hat{F}(x^k) = \hat{F}(x^{k-1}) + p(x^{k-1}) \sum_{y_k < x_k} p(y_k|x^{k-1})
\]

implies \( \hat{F}(x^k) \geq \hat{F}(x^{k-1}) \).

- Similarly,

\[
 F(x^k) = \sum_{y_k \leq x_k} p(y_k)
\]

\[
 = \sum_{y_{k-1} \leq x_{k-1}} \sum_{y_k \in \mathcal{X}} p(y_{k-1}|y_k) + \sum_{y_k \leq x_k} p(x_{k-1}|y_k)
\]

\[
 = \hat{F}(x^{k-1}) + p(x^{k-1}) \sum_{y_k \leq x_k} p(y_k|x^{k-1})
\]

\[
 \leq \hat{F}(x^{k-1}) + p(x^{k-1}) = F(x^k)
\]

- Thus \( [\hat{F}(x^k), F(x^k)] \subset [\hat{F}(x^{k-1}), F(x^{k-1})] \), \( k = 2, \ldots, n \)

Given the codeword \( b_1 b_2 \cdots b_l \), compute \( z = \sum_{i=1}^l b_i 2^{-i} \).

- Recall that \( x^n \) is the unique string in \( \mathcal{X}^n \) such that \( \hat{F}(x^n) < z < F(x^n) \). Since \( [\hat{F}(x^k), F(x^k)] \subset [\hat{F}(x^{k-1}), F(x^{k-1})] \) for all \( k \), \( x^k \) is the unique string in \( \mathcal{X}^k \) satisfying \( \hat{F}(x^k) < z < F(x^k) \). Thus we can decode using the following procedure:

  - Starting from \( k = 1 \), find sequentially the largest \( y \in \mathcal{X} \) such that

\[
 \hat{F}(y^k) < z
\]

  - and let \( x_k = y \). Repeat this for \( k = 2, \ldots, n \). The resulting \( x^n \) will satisfy \( \hat{F}(x^n) < z < F(x^n) \), so we decode \( x^n \).

  - Since only the values \( \hat{F}(x^{k-1}y), y \in \mathcal{X}, k = 1, \ldots, n \) are needed, the number of additions and comparisons is again \( O(n|\mathcal{X}|) \).
Another look at the encoding process

- Define the intervals \( J(x^k) = [\hat{F}(x^k), F(x^k)] \), \( k = 1, \ldots, n \)
- We showed that \( J(x^k) \subset J(x^{k-1}) \), \( k = 2, \ldots, n \). Each \( J(x^k) \) has length \( p(x^k) \), so the lengths shrink exponentially with \( k \).
- The tag satisfies \( I(x^n) \in J(x^k) \) for each \( k \).
- Consider a dyadic interval \([0,b_1 b_2 \ldots b_j, 0,b_1 b_2 \ldots b_j + 2^{-j}]\). This contains all numbers in \([0,1]\) whose binary representation starts with the bits \( b_1 b_2 \ldots b_j \). Now choose \( b_1 b_2 \ldots b_j \) such that

\[
J(x^k) \subset \left[0,b_1 b_2 \ldots b_j, 0,b_1 b_2 \ldots b_j + 2^{-j}\right]
\]

Since \( I(x^n) = [0.C(x^n), 0.C(x^n) + 2^{-l(x^n)}] \subset J(x^n) \), we have

\[
I(x^n) \subset \left[0,b_1 b_2 \ldots b_j, 0,b_1 b_2 \ldots b_j + 2^{-j}\right].
\]

- Thus the binary string \( b_1 b_2 \ldots b_j \) is a prefix of \( C(x^n) \) and the encoder can release the bits \( b_1 b_2 \ldots b_j \).

**Example:**

\( \mathcal{X} = \{0,1\}, \) i.i.d. source with pmf \( p(0) = 0.8, p(1) = 0.2, x^4 = 0110 \).

Simplify notation: \( \hat{F}(x^k) = \hat{F}_k, F(x^k) = F_k, p(x^k) = p_k \), so that

\[
\hat{F}_k = \hat{F}_{k-1} + p_{k-1} p(y), \quad p_k = p_{k-1} \cdot p(x^k)
\]

Always choose the smallest dyadic interval (of length \( \leq 1/2 \)) containing \( J_n = [\hat{F}_n, F_n] \) and produce corresponding bit(s):

- \( x_1 = 0 \):

\[
\hat{F}_1 = 0, \quad F_1 = 0 + p(0) = 0.8, \quad J_1 = [\hat{F}_1, F_1] = [0, 0.8]
\]

\( J_1 \) is not included in a dyadic interval \( \implies \) no bits produced.

- \( x_2 = 1 \):

\[
\hat{F}_2 = 0 + p_1 \cdot p(0) = 0.64, \quad F_2 = 0.64 + p_2 = 0.8, \quad J_2 = [0.64, 0.8)
\]

\([0.64, 0.8) \subset [1/2, 1) = [0.1, 1.0] \implies b_1 = 1\)

**Sequential encoding**

- The codeword bits can be generated sequentially: the \( x_k \) are read one-by-one, and as soon as \( J(x^k) \) is contained in a dyadic interval, the corresponding (previously unsent) bits are released in an incremental fashion. Then the process starts over until \( k = n \).
- The decoder can reverse the above procedure to sequentially generate the symbols of \( x^n \) as the bits of the codeword \( C(x^n) \) are received.

**Example cont’d:**

- \( x_3 = 1 \):

\[
\hat{F}_3 = 0.64 + p_2 \cdot p(0) = 0.768, \quad F_3 = 0.768 + p_3 = 0.8, \quad J_3 = [0.768, 0.8)
\]

\([0.768, 0.8) \subset [3/4, 3/4 + 1/16) = [0.1100, 0.1101] \implies b_2 b_3 b_4 = 100\)

- \( x_4 = 0 \): Encoding terminates since we know that \( n = 4 \).

\[
\hat{F}_4 = 0.768 + p_3 \cdot 0 = 0.768, \quad T_4 = 0.768 + \frac{1}{2} p_4 = 0.7808
\]

We have \( 0.7808 = 0.110001111110 \ldots \) and

\[
l(0110) = \lceil - \log p_4 \rceil + 1 = 7
\]

so \( C(0110) = b_1 b_2 b_3 b_4 b_5 b_6 b_7 = 1100011\)

**Note:** The first 4 of the 7 bits were produced before the entire source string was processed.
Practical issues

- The described (simplified) procedure requires floating point operations with infinite precision. The interval \( J(x^k) \) shrinks very fast as \( k \) increases. The procedure quickly becomes impractical for larger block length \( n \).
- Encoding and decoding delay: How many input letters should be processed for the output to be decodable?
- The above problems are solved by practical algorithms for arithmetic coding (Rissanen (1976), Pasco (1976)).

Universal Coding

- All codes we studied so far (Huffman, Shannon-Fano) assumed the source distribution (pmf) is known.
- In practice we often have limited information about the true distribution of the source. Instead, all we know (or assume) is that the source distribution is one from a given family (class) of source distributions.
- A sequence of codes that (asymptotically) compresses any source in a given source class to its entropy is called universal for that source class.
- Examples of source classes: stationary and memoryless (i.i.d.) sources, stationary Markov chains, stationary Markov processes of a given order \( k \); stationary and ergodic sources.

Conclusion

- Arithmetic coding gives a practical method for lossless coding having codeword length

\[
l(x^n) = - \log p(x^n) + O(1)
\]

for all \( x^n \in \mathcal{X}^n \) when designed for pmf \( p(x^n) \).
- If the source \( X^n \) has pmf \( p(x^n) \) and the code design uses pmf \( q(x^n) \), then

\[
L_n(C) = \frac{1}{n} \sum x^n l(x^n)p(x^n) = \frac{1}{n} H(X^n) + \frac{1}{n} D(p||q) + O(1/n)
\]

- Arithmetic coding reduces the problem of lossless coding to finding a good coding distribution \( q(x^n) \) for the given data (more on this later).

Idea for universal coding:

- Use the data \( x^n \) to form an estimate of the source distribution.
- Construct optimal code for the estimated distribution.
- Use code to compress \( x^n \) and transmit the compressed data along with the description of the code (or the distribution)
- The decoder can reconstruct \( x^n \) since it knows (or can generate) the code used to compress it.

Note: This is not the only way to construct universal codes.

To show an example for this general procedure we need some new definitions from information theory.
Assume $\mathcal{X} = \{1, 2, \ldots, m\}$.

**Definition (Type)**

For $x^n \in \mathcal{X}^n$ and $a \in \mathcal{X}$ let

$$n(a|x^n) = |\{i : x_i = a, i = 1, 2, \ldots, n\}|$$

be the number of times $a$ occurs in $x^n$. Define the type (empirical distribution) of $x^n$ as the pmf on $\mathcal{X}$ given by

$$P_{x^n}(a) = \frac{n(1|x^n)}{n}, \frac{n(2|x^n)}{n}, \ldots, \frac{n(m|x^n)}{n}$$

**Note:** $P_{x^n}$ is indeed a pmf on $\mathcal{X}$ since

$$\sum_{a \in \mathcal{X}} P_{x^n}(a) = \frac{1}{n} \sum_{a \in \mathcal{X}} n(a|x^n) = \frac{n}{n} = 1$$

**Enumerative universal code**

Define $C : \mathcal{X}^n \rightarrow \{0, 1\}^*$ using the following construction:

1. Given $x^n$, describe its type $P_{x^n}$ using a fixed-length nonsingular code $C_1 : \mathcal{P}_n \rightarrow \{0, 1\}^{l_1}$, where

   $$l_1 = \lceil \log |\mathcal{P}_n| \rceil$$

2. Given $P_{x^n}$, use a fixed-length nonsingular code $C_2 : T(P_{x^n}) \rightarrow \{0, 1\}^{l_2}$ to identify $x^n$ in $T(P_{x^n})$, where

   $$l_2 = \lceil \log |T(P_{x^n})| \rceil$$

3. Define the code $C$ by

   $$C(x^n) = C_1(P_{x^n})C_2(x^n)$$

**Note:** $C$ is a variable-length code since $l_2$ depends on the type of $x^n$.

**Definition (Type class)**

Let $\mathcal{P}_n$ denote the set of all types of sequences in $\mathcal{X}^n$. If $P \in \mathcal{P}_n$, then $T(P)$ denotes the set of sequences of type $P$:

$$T(P) = \{x^n \in \mathcal{X}^n : P_{x^n} = P\}$$

**Combinatorial facts** (recall that $|\mathcal{X}| = m$):

- The number of types of sequences in $\mathcal{X}^n$ is

  $$|\mathcal{P}_n| = \binom{n + m - 1}{m - 1}$$

- For any $P \in \mathcal{P}_n$, the size of the associated type class is

  $$|T(P)| = \frac{n!}{(nP(1)!(nP(2))! \cdots (nP(m))!}$$

- $C$ is clearly nonsingular.
- $C$ is a prefix code: if $C(y^n)$ is a prefix of $C(x^n)$, then
  $C_1(y^n) = C_1(x^n)$, and so $P_{x^n} = P_{y^n}$. But then $|C(y^n)| = |C(x^n)|$, so we must have $C(y^n) = C(x^n)$, which implies $y^n = x^n$.
- The length of $C(x^n)$ is

  $$l(x^n) = \lceil \log |\mathcal{P}_n| \rceil + \lceil \log |T(P_{x^n})| \rceil$$

- We know $|\mathcal{P}_n|$ and $|T(P_{x^n})|$ exactly, but these expressions are hard to work with. Instead, we will develop some useful estimates.
**Theorem 2 (Number of types)**

\[ |\mathcal{P}_n| \leq (n + 1)^m \]

**Proof:** For each \( a \in \mathcal{X} \), \( n(a|x^n) \) can take at most \( n + 1 \) values \( 0, 1, \ldots, n \). Thus the vector \( (n(1|x^n), n(2|x^n), \ldots, n(m|x^n)) \) can take at most \( (n + 1)^m \) different values. \( \square \)

For any pmf \( Q \) on \( \mathcal{X} \), let \( Q^n \) be the product distribution on \( \mathcal{X}^n \):

\[ Q^n(x^n) = \prod_{i=1}^{n} Q(x_i) \]

Recall the definition of the relative entropy between two pmfs \( P \) and \( Q \):

\[ D(P||Q) = \sum_{a \in \mathcal{X}} P(a) \log \frac{P(a)}{Q(a)} \]

and the fact that \( D(P||Q) \geq 0 \) with equality if and only of \( P = Q \).

**Theorem 4 (size of type class)**

**For any type \( P \in \mathcal{P}_n \)**

\[ |T(P)| \leq 2^n H(P) \]

**Proof:** The previous lemma implies that if \( x^n \in T(P) \) (i.e., \( P_{x^n} = P \)), then

\[ P^n(x^n) = 2^{-n \left[ D(P_{x^n}||P) + H(P_{x^n}) \right]} = 2^{-n H(P)} \]

Thus

\[ 1 \geq \sum_{x^n \in T(P)} P^n(x^n) \]

\[ = \sum_{x^n \in T(P)} 2^{-n H(P)} \]

\[ = |T(P)| 2^{-n H(P)} \]

\( \square \)

**Lemma 3**

**For any pmf \( Q \) on \( \mathcal{X} \) and \( x^n \in \mathcal{X}^n \)**

\[ Q^n(x^n) = 2^{-n \left[ D(P_{x^n}||Q) + H(P_{x^n}) \right]} \]

**Proof:**

\[ Q^n(x^n) = \prod_{i=1}^{n} Q(x_i) = \prod_{a \in \mathcal{X}} Q(a)^n(a|x^n) \]

\[ = \prod_{a \in \mathcal{X}} Q(a)^n P_{x^n}(a) = \prod_{a \in \mathcal{X}} 2^n P_{x^n}(a) \log Q(a) \]

\[ = \prod_{a \in \mathcal{X}} 2^n \left[ P_{x^n}(a) \log \frac{Q(a)}{P_{x^n}(a)} + P_{x^n}(a) \log P_{x^n}(a) \right] \]

\[ = 2^n \sum_{a \in \mathcal{X}} \left[ P_{x^n}(a) \log \frac{Q(a)}{P_{x^n}(a)} + P_{x^n}(a) \log P_{x^n}(a) \right] \]

\[ = 2^{-n \left[ D(P_{x^n}||Q) + H(P_{x^n}) \right]} \]

\( \square \)

**Analysis of expected codeword length**

Using the upper bounds \( |\mathcal{P}_n| \leq (n + 1)^m \) and \( |T(P_{x^n})| \leq 2^n H(P_{x^n}) \), we have

\[ l(x^n) = \left[ \log |\mathcal{P}_n| \right] + \left[ \log |T(P_{x^n})| \right] \leq m \log(n + 1) + n H(P_{x^n}) + 2 \]

Assume \( X_1, X_2, \ldots \) is an i.i.d. source with an arbitrary pmf \( Q \). Then Lemma 3 implies

\[ \log Q^n(x^n) = -n D(P_{x^n}||Q) - n H(P_{x^n}) \]

so that

\[ n H(P_{x^n}) = -\log Q^n(x^n) - n D(P_{x^n}||Q) \]

and

\[ nE[H(P_{X^n})] = -\sum_{x^n} Q^n(x^n) \log Q^n(x^n) - \sum_{x^n} Q(x^n) n D(P_{x^n}||Q) \]

\[ \geq 0 \]
We obtain
\[ El(X^n) \leq m \log(n + 1) + nE[H(P_{X^n})] + 2 \]
\[ \leq m \log(n + 1) + H(X^n) + 2 \]
Thus
\[ \frac{1}{n} El(X^n) \leq \frac{m \log(n + 1)}{n} + \frac{H(X^n)}{n} + \frac{2}{n} \]
Since \( H(X^n) = nH(Q) \),
\[ \lim_{n \to \infty} \frac{1}{n} El(X^n) = H(X) \]
where \( H(X) = H(Q) \) is the entropy of the i.i.d. source \( X_1, X_2, \ldots \).
The same code works for all \( Q \): The code is universal for the class of i.i.d. sources!

**Remarks:**
- With more work, the upper bound
  \[ \frac{1}{n} El(X^n) - H(X) \leq \frac{m \log(n + 1)}{n} + \frac{2}{n} \]
can be improved to
  \[ \frac{1}{n} El(X^n) - H(X) \leq \frac{(m - 1) \log n}{2n} + \frac{c}{n} \]
where \( c > 0 \) depends on \( m \) only.
- The universal code we constructed is a so-called two-pass code: the entire string needs to be processed (first pass) before encoding (second pass) can start.
- With the aid of arithmetic coding we will construct one-pass (online) universal codes.

**Source classes and universal codes**
- Assume \( \mathcal{P} \) is a class of distributions of sources \( X_1, X_2, \ldots \) with alphabet \( \mathcal{X} \).
- The redundancy of a prefix \( n \)-code \( C_n : \mathcal{X}^n \to \{0, 1\}^* \) with respect to the source distribution \( p \in \mathcal{P} \) is
  \[ R(C_n, p) = \frac{1}{n} \left[ El(X^n) - H_P(X^n) \right] \]
where \( l(x^n) = |C_n(x^n)| \).

**Definition (Universal code)**
A sequence of prefix \( n \)-codes \( \{C_n\} \) is universal with respect to the source class \( \mathcal{P} \) if for any \( p \in \mathcal{P} \)
\[ \lim_{n \to \infty} R(C_n, p) = 0 \]

**Examples of source classes**

*Stationary and memoryless (i.i.d.) sources:*
- Let \( \mathcal{X} = \{1, \ldots, m\} \) and define the parameter set
  \[ \Theta_0 = \{(p_1, p_2, \ldots, p_m) : \sum_{i=1}^{m} p_i = 1, p_i \geq 0 \text{ all } i\} \]
Any \( \theta = (p_1, \ldots, p_m) \) defines a pmf \( p_\theta(x^n) \) on \( \mathcal{X}^n \) given by
  \[ p_\theta(x^n) = \prod_{i=1}^{m} p_{x_i} \]
i.e., if \( X^n \sim p_\theta(x^n) \), then \( X_1, X_2, \ldots, X_n \) are i.i.d. with pmf \( P(X_i = j) = p_j, j \in \mathcal{X} \).
- \( \Theta_0 \) parametrizes the set of all i.i.d. distributions \( \mathcal{P}_{\Theta_0} \).
**Note:** Recall the bound for the redundancy of the enumerative code $C_n$: for all memoryless sources $p_\theta \in \mathcal{P}_\Theta$

$$R(C_n, p_\theta) = \frac{1}{n} \left[ E_{p_\theta} I(X^n) - H_{p_\theta}(X^n) \right] \leq \frac{m \log(n+1) + 2}{n}$$

Thus this code is universal for $\mathcal{P}_\Theta$, in the strong sense that its worst-case redundancy asymptotically vanishes:

$$\lim_{n \rightarrow \infty} \max_{\theta \in \Theta} R(C_n, p_\theta) = 0$$

---

**Redundancy and coding distributions**

- For a prefix $n$-code $C$ with codeword length $l(x^n)$ define
  $$c = \left( \sum_{x^n \in \mathcal{X}^n} 2^{-l(x^n)} \right)^{-1}$$

  Then $q(x^n) = c 2^{-l(x^n)}$ is a valid pmf on $\mathcal{X}^n$ and since $c \geq 1$ by Kraft’s inequality,
  $$l(x^n) \geq - \log q(x^n)$$

- Conversely, if $q(x^n)$ is a pmf on $\mathcal{X}^n$, then the Shannon-Fano code with codeword lengths $l(x^n) = \lceil - \log q(x^n) \rceil$ is a prefix code such that
  $$l(x^n) < - \log q(x^n) + 1$$

- Thus every pmf $q(x^n)$ can serve as a coding distribution (model) and every prefix code gives rise to a coding distribution.

  Coding distributions and prefix codes are essentially *equivalent* (within 1 bit)!

---

**First-order Markov sources:**

- Let
  $$\Theta_1 = \{(p_{ij})_{i,j=1}^n : \sum_{j=1}^m p_{ij} = 1, p_{ij} \geq 0 \text{ all } i \text{ and } j\}$$

- Let $(q_1, \ldots, q_m)$ be a pmf on $\mathcal{X}$ (initial distribution) and for $\theta = (p_{ij})$ define the pmf $p_\theta(x^n)$ by
  $$p_\theta(x^n) = q_z \prod_{i=2}^m p_{x_{i-1}, x_i}$$

  Thus, if $X^n \sim p_\theta(x^n)$, then $X_1, \ldots, X_n$ is a Markov chain with initial distribution $P(X_1 = j) = q_j$ and transition probabilities
  $$P(X_k = j | X_{k-1} = i) = p_{ij}$$

- $\Theta_1$ parametrizes the set of *all* first-order Markov chain distributions (up to the initial distribution).

---

**Lemma 5**

*If $C$ is the Shannon-Fano code for distribution $q(x^n)$ and $X^n$ has pmf $p(x^n)$, then*

$$D(p\|q) \leq \left[ E_p I(X^n) - H_p(X^n) \right] < D(p\|q) + 1$$

**Proof:** Since $l(x^n) = \lceil - \log q(x^n) \rceil$,

$$\log \frac{p(x^n)}{q(x^n)} \leq l(x^n) + \log p(x^n) < \log \frac{p(x^n)}{q(x^n)} + 1$$

Thus

$$\sum_{x^n \in \mathcal{X}^n} p(x^n) \log \frac{p(x^n)}{q(x^n)} \leq \sum_{x^n \in \mathcal{X}^n} p(x^n) \left( l(x^n) + \log p(x^n) \right) < \sum_{x^n \in \mathcal{X}^n} p(x^n) \left( \log \frac{p(x^n)}{q(x^n)} + 1 \right)$$

$\square$
Note: Normalizing by the block length we get the bound
\[
\frac{1}{n} D(p\|q) \leq R(C_n, p) < \frac{1}{n} D(p\|q) + \frac{1}{n}
\]
The 1/n term becomes 2/n if arithmetic coding is used instead of Shannon-Fano.

Corollary 6

A sequence of Shannon-Fano (or arithmetic) codes \(\{C_n\}\) obtained from a sequence of coding distributions \(\{q_n\}\) is universal for a source class \(\mathcal{P}\) if and only if
\[
\lim_{n \to \infty} \frac{1}{n} D(p^n\|q_n) = 0 \quad \text{for all} \quad p \in \mathcal{P}
\]

How can we obtain such universal coding distributions?

- Let \(f\) be a probability density on \(\Theta\). Then
  \[
  q(x^n) = \int_{\Theta} p_\theta(x^n) f(\theta) d\theta
  \]
is a pmf on \(X^n\).
- \(q\) is a mixture of i.i.d. distributions on \(X^n\). The idea is that if we appropriately “mix” i.i.d. sources, then we obtain a coding distribution \(q\) that works very well for each i.i.d. source.
- We consider Dirichlet densities with parameters \(\alpha_i > 0\), \(i = 1, \ldots, m\) on \(\Theta\):
  \[
  f_{\alpha_1, \ldots, \alpha_m}(\theta) = \frac{\Gamma(\sum_{i=1}^{m} \alpha_i)}{\prod_{i=1}^{m} \Gamma(\alpha_i)} \prod_{i=1}^{m} p_i^{\alpha_i-1}
  \]
  where \(\Gamma\) denotes the gamma function. (For \(m = 2\) this is the pdf of the Beta(\(\alpha_1, \alpha_2\)) distribution.)

Mixture distributions

- Let \(\mathcal{X} = \{1, \ldots, m\}\) and define the parameter set \(\Theta \subset \mathbb{R}^{m-1}\) by
  \[
  \Theta = \left\{ (p_1, \ldots, p_{m-1}) : \sum_{i=1}^{m-1} p_i \leq 1; \ p_i \geq 0, \ i = 1, \ldots, m-1 \right\}
  \]
  For \(\theta = (p_1, \ldots, p_{m-1}) \in \Theta\) and \(x^n = (x_1, \ldots, x_n) \in \mathcal{X}^n\) let
  \[
  p_\theta(x^n) = \prod_{j=1}^{n} p_{x_j} = \prod_{i=1}^{m} p_i^{n(i|x^n)}
  \]
  where \(p_m = 1 - \sum_{i=1}^{m-1} p_i\).
- Then \(p_\theta(\cdot)\) is a pmf on \(\mathcal{X}^n\) for any fixed \(\theta = (p_1, \ldots, p_{m-1})\), corresponding to an i.i.d. source \(X_1, X_2, \ldots, X_n\) with distribution
  \[
  P(X_i = j) = p_j, \quad j \in \mathcal{X}
  \]

Lemma 7 (Dirichlet mixture distribution)

The coding distribution corresponding to the Dirichlet \((\alpha_1, \ldots, \alpha_m)\) pdf is given by
\[
q(x^n) = \int_{\Theta} p_\theta(x^n) f_{\alpha_1, \ldots, \alpha_m}(\theta) d\theta = \frac{\prod_{i=1}^{m} \prod_{j=1}^{n} \left(n(i|x^n) + \alpha_i - j\right)}{\prod_{j=1}^{n} (n + \sum_{i=1}^{m} \alpha_i - j)}
\]
where \(\prod_{j=1}^{n} \left(n(i|x^n) + \alpha_i - j\right) = 1\) if \(n(i|x^n) = 0\).

We will consider two special cases: \(\alpha_i = 1, \ i = 1, \ldots, m\), and \(\alpha_i = 1/2, \ i = 1, \ldots, m\).
Dirichlet\((1, \ldots, 1)\) mixture

Setting \(\alpha_i = 1, i = 1, \ldots, m\) in

\[
\int p_\theta(x^n) f_{\alpha_1, \ldots, \alpha_m}(\theta) d\theta = \frac{\prod_{i=1}^m \prod_{j=1}^n (n(i|x^n) + \alpha_i - j)}{\prod_{j=1}^n (n + \sum_{i=1}^m \alpha_i - j)}
\]

we obtain

\[
\hat{q}(x^n) = \frac{\prod_{i=1}^m n(i|x^n)!}{\prod_{j=1}^n (n + m - j)}
\]

Note that \(f_{\alpha_1, \ldots, \alpha_m}(\theta) = \frac{\Gamma\left(\sum_{i=1}^m \alpha_i\right)}{\prod_{i=1}^m \Gamma(\alpha_i)} \prod_{i=1}^m p_i^{\alpha_i-1}\) becomes

\[
f_{1, \ldots, 1}(\theta) = \frac{\Gamma(m)}{\prod_{i=1}^m \Gamma(1)} \prod_{i=1}^m p_i^0 = (m - 1)!, \quad \theta \in \Theta
\]

Thus \(\hat{q}\) is the uniform mixture of i.i.d. sources.

Krichevsky and Trofimov (KT) coding distribution

The following lemma is of great practical importance.

\[\text{Lemma 8}\]

For all \(1 \leq t \leq n\) define

\[
q(i|x^{t-1}) = \frac{n(i|x^{t-1}) + 1/2}{t - 1 + m/2}
\]

Then

(i) \(q(i|x^{t-1})\) is a conditional pmf on \(X\) given \(x^{t-1}\).

(ii) \(\prod_{i=1}^n q(x_t|x^{t-1}) = \frac{\prod_{i=1}^n n(i|x^{t-1}) + 1/2}{\prod_{j=1}^n (n+m/2-j)} = q(x^n)\)

The conditional probabilities \(q(x_t|x^{t-1})\) can be fed to an arithmetic coder \(\implies\) practical implementation!
Remarks:

- The conditional probability
  \[ q(i|x^{n-1}) = \frac{n(i|x^{n-1}) + 1/2}{n - 1 + m/2} \]
  is a biased estimate (because of the term 1/2) of the probability of letter \( i \) occurring next based on \( x^{n-1} \).
- The natural unbiased estimate
  \[ \hat{q}(i|x^{n-1}) = \frac{n(i|x^{n-1})}{n - 1} \]
  does not work because a coding distribution must satisfy \( q(x^n) > 0 \) for all \( x^n \).
- It is not clear at this point (but can be proved) that the optimal bias term is indeed 1/2.

Example:

For \( X = \{0, 1\} \) we have

\[ q(0|x^{n-1}) = \frac{n(0|x^{n-1}) + 1/2}{n}, \quad q(1|x^{n-1}) = \frac{n(1|x^{n-1}) + 1/2}{n} \]

where \( n(0|x^{n-1}) \) is the number of zeroes in the binary string \( x^{n-1} = x_1x_2\ldots x_{n-1} \).

If \( n = 5 \) and \( x^5 = 01100 \) the KT probability is

\[ q(01100) = \prod_{t=1}^{5} q(x_t|x_{t-1}) = \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{3}{8} \cdot \frac{1}{2} = \frac{3}{256} \]
Lempel-Ziv coding

- A family of lossless coding algorithms based on the seminal work of J. Ziv and A. Lempel.
- The algorithms are based on dictionaries and string matching: no explicit probability modeling is needed.
- Simple, elegant, and universal for the class of stationary and ergodic sources.
- Variants widely used in practice.
- We follow the treatment of Lempel-Ziv coding in Cover&Thomas.

Notation:

\[ x^k = x_{i+1} \cdots x_k, \quad x^k = x^k_1 = x_1 x_2 \cdots x_k \]

**Example:**

- Let \( X = \{a, b, c, \ldots, z\} \) and \( w = 6 \).
- The source string
  
  \[ a a b a b a a c a b \]
  
  is parsed by LZ77 as follows (substrings between two commas are the phrases):
  
  \[ a, a, b, a a b a a, c, a b, \]
  
  and the corresponding pointers are
  
  \( (0, a) (1, 1) (0, b) (1, 3) (0, c) (1, 5, 2) \)

- Note that the matched phrase starting in the window can extend beyond the window.
- Decoding is obvious.

**LZ77: sliding window Lempel-Ziv algorithm**

- Set the positive integer window length \( w \). A string \( x^n = x_1 x_2 \cdots x_n \) from the finite alphabet \( X \) is to be encoded.
- Assume \( x_1 \cdots x_{i-1} \) has been encoded. Find largest \( k \) such that for some \( j \) in the range \( 1 \leq j \leq w \),
  
  \[ x_{i-j+k-1} = x_i^{i+k-1} \]
  
  i.e., the longest match of a string of \( k \) not yet encoded symbols (phrase) with a string starting in the window (search buffer) consisting of the last \( w \) symbols \( x_{i-w}^i = x_{i-w} x_{i-w+1} \cdots x_{i-1} \).
- Represent the phrase \( x_i^{i+k-1} \) with the pair \( (j, k) \), i.e., the location where the match starts in the window and the length of the match.
- In no match is found, send \( x_i \) uncompressed.
- The encoded phrase is represented by the pointer \( (F,j,k) \) or \( (F,x_i) \) where \( F = 1 \) if a match is found and \( F = 0 \) if there is no match.

- The dictionary maintained by LZ77 consists of all substrings of the string in the window, a portion of the previously encoded sequence.
- The window length \( w \) is quite large in practice (say \( w = 2^{12} \)).
- If the maximum length of the matched phrase is restricted by \( L \) (as is always the case in practice), then the pointers can be encoded with a binary code of fixed length
  
  \[ \lceil \log w \rceil + \lceil \log L \rceil \]

- The pointers can further be compressed using a variable-length binary codes.
- Practical versions used in the compression packages PKZIP, ZIP, gzip, etc., and in PNG.
Optimality of LZ77

- The proof of the practical (finite-window) version is involved.
- We assume an idealized setup: the window contains the entire infinite past \( \ldots, X_{-2}, X_{-1}, X_0 \), of the source.
- To encode the block \( X^n = X^n_1 = (X_1, \ldots, X_n) \), find the last time \( X^n \) appeared in the past
  \[
  R_n(X^n) = \min\{j \geq 0 : X^n_{-j} \neq X^n\}
  \]
- The decoder is also assumed to have access to the entire past, so having received \( R_n \) can produce \( X^n \).
- To encode \( R_n \) losslessly we need an efficient prefix code on the set of positive integers \( \mathbb{N} \).

The lemma states that there is a prefix code that can encode \( R_n \) using \( l_n = \log R_n + 2 \log \log R_n + O(1) \) bits.

**Theorem 12**

Assume \( \{X_i\} = \ldots, X_{-1}, X_0, X_1, \ldots \) is a stationary and ergodic source having entropy rate \( H(X) \). Then the expected codeword length of the simplified version of the LZ77 algorithm satisfies

\[
\lim_{n \to \infty} \frac{1}{n} E l_n(X^n) = \bar{H}(X)
\]

- The theorem will follow from the fact that given \( X^n = x^n \), the expected length of the time we have to wait to see the pattern \( x^n \) again is \( 1/p(x^n) \).

\[\text{Lemma 11}\]

There is a prefix code \( C : \mathbb{N} \to \{0, 1\}^* \) with codeword length satisfying

\[
|C(k)| = \log k + 2 \log \log k + O(1)
\]

**Proof:** Define first a code \( C_1 : \mathbb{N} \to \{0, 1\}^* \) as follows: For \( k \in \mathbb{N} \) let \( l_k = \lceil \log k \rceil \) and

\[
C_1(k) = 0 \cdots 0 b_1 \cdots b_k
\]

where \( b_j \) is the \( j \)-th digit in \( k \) in binary.

\( C_1 \) is clearly a prefix code such that \( |C_1(k)| \leq 2\lceil \log k \rceil + 1 \). Now define \( C \) by

\[
C(k) = C_1(\lceil \log k \rceil) b_1 \cdots b_{l_k}
\]

Then \( C \) is a prefix code with code length

\[
|C(k)| = |C_1(\lceil \log k \rceil)| + \lceil \log k \rceil
\]

\[
= 2\lceil \log \lceil \log k \rceil \rceil + 1 + \lceil \log k \rceil
\]

\[
= \log k + 2 \log \log k + O(1) \quad \square
\]

- Let us define a new source alphabet \( \mathcal{U} = X^n \) and source \( \{U_i\} \) by setting

\[
U_i = X_{i+n-1}^i, \quad i = 0, \pm 1, \pm 2, \ldots
\]

- Clearly, the new process \( \{U_i\} \) is also stationary, i.e., \( (U_1, \ldots, U_k) \) and \( (U_j, \ldots, U_{j+k-1}) \) have the same distribution for all \( k = 1, 2, \ldots \) and \( j = 0, \pm 1, \pm 2, \ldots \).
- For simplicity, in the next lemma we assume that \( \{X_i\} \) is an i.i.d. process, but the lemma holds for all stationary and ergodic \( \{X_i\} \).

**Lemma 13 (Kac)**

For any \( u \in \mathcal{U} \) such that \( P(U_1 = u) > 0 \) let for \( i = 1, 2, \ldots \)

\[
Q_u(i) = P(U_{i-1} = u, U_j \neq u \text{ for } -i+1 < j < i | U_1 = u)
\]

Then

\[
E[R_1(U_1) | U_1 = u] = \sum_{i=1}^\infty iQ_u(i) = \frac{1}{P(U_1 = u)}
\]
Remark: It follows from the definition of \{U_i\} that
\[ R_1(U_1) = R_n(X^n) \]

Proof of lemma: Define the events \(A_{jk}\) for \(j = 0, 1, 2, \ldots\) and \(k = 1, 2, \ldots\) by
\[ A_{jk} = \{U_j = u, U_{l} \neq u \text{ for } 0 \leq l < k, U_k = u\} \]
Then
\[ \bigcup_{j,k} A_{jk} \]
is the event that the letter \(u\) occurs at least once both in the sequence
\[ \ldots, U_{-2}, U_{-1}, U_0, \text{ and in the sequence } U_1, U_2, U_3, \ldots \]
Using the fact that \(\{U_m\}_{m = -\infty}^{\infty}\) is an i.i.d. process, it is easy to show (homework) that
\[ P\left( \bigcup_{j,k} A_{jk} \right) = 1 \]

Proof of theorem: One can show that
\[ \frac{1}{n} E l_n(X^n) \geq \bar{H}(X) \]
(this does not immediately follow from the entropy lower bound since the code is “random” in that it depends on the past of the source).

Recall that \(l_n = \log R_n + 2\log \log R_n + O(1)\). Using the concavity of the logarithm and Jensen’s inequality
\[
E \log R_n(X^n) = \sum_{x^n} P(X^n = x^n) E[\log R_n(X^n)|X^n = x^n] \\
\leq \sum_{x^n} P(X^n = x^n) \log E[R_n(X^n)|X^n = x^n] \\
= \sum_{x^n} P(X^n = x^n) \log \frac{1}{P(X^n = x^n)} \quad \text{(Kac’s lemma)} \\
= H(X^n)
\]

Proof of lemma cont’d: Since the \(A_{jk}\) are disjoint
\[
1 = P\left( \bigcup_{j,k} A_{jk} \right) = \sum_{j=0}^{\infty} \sum_{k=1}^{\infty} P(A_{jk}) \\
= \sum_{j=0}^{\infty} \sum_{k=1}^{\infty} P(U_k = u)P(U_{-j} = u, U_l \neq u \text{ for } 0 \leq l < k | U_k = u) \\
= \sum_{j=0}^{\infty} \sum_{k=1}^{\infty} P(U_k = u)Q_u(j + k) \quad \text{(since } \{U_i\} \text{ is stationary)} \\
= P(U_1 = u) \sum_{j=0}^{\infty} \sum_{k=1}^{\infty} Q_u(j + k) \\
= P(U_1 = u) \sum_{i=1}^{\infty} iQ_u(i) \quad \text{(by collecting terms with } j + k = i) \\
Thus
\[ E[R_1(U_1)|U_1 = u] = \sum_{i=1}^{\infty} iQ_u(i) = \frac{1}{P(U_1 = u)} \quad \blacksquare \]

Proof of theorem cont’d: A very similar application of Jensen’s inequality shows that
\[ E \log \log R_n(X^n) \leq \log H(X^n) \]
Since
\[ \log H(X^n) \leq \log \log |X|^n = \log n + \log \log |X| \]
the expected codeword length is bounded as
\[
\bar{H}(X) \leq \frac{1}{n} E l_n(X^n) \leq \frac{1}{n} H(X^n) + \frac{2\log n}{n} + O\left(\frac{1}{n}\right) \\
\]
The upper bound converges to \(\bar{H}(X)\) as \(n \to \infty\) and we get
\[
\lim_{n \to \infty} \frac{1}{n} E l_n(X^n) = \bar{H}(X) \quad \blacksquare
\]
LZ78: Tree-structured Lempel-Ziv compression

- Builds explicit dictionary by incrementally parsing the input sequence into shortest phrases that have not been seen so far.
- The prefix consisting of all but the last symbol of a new phrase must have appeared before.
- Each phrase is represented by a pair \((i, s)\) where \(i\) is the index of this prefix in the current dictionary (0 if the phrase is a symbol not in the dictionary yet) and \(s\) is the last symbol (uncoded).
- For example, \(a b b a a a c b b a a c b a a\) is parsed by LZ78 as \(a, b, b a, a a, c, b b, a a c, b a a\)

\[
(0, a) \ (0, b) \ (2, a) \ (1, a) \ (0, c) \ (2, b) \ (4, c) \ (3, a)
\]

Decoding is obvious as the decoder can build the same dictionary.
- The dictionary has a natural tree structure where the nodes of the tree are the phrases seen so far.
- Let \(c(x^n)\) denote the number of phrases in the dictionary obtained by parsing \(x^n\).
- Each location pointer requires about \(d \log c(x^n)\) bits to encode, a phrase requires \(d \log c(x^n) + \log |X|\) bits, so the codeword length for the string \(x^n\) is

\[
l_n(x^n) = c(x^n)(\log c(x^n) + O(1))
\]

An ingenious proof by Wyner and Ziv shows that for any stationary and ergodic source having entropy rate \(\bar{H}(X)\),

\[
\lim_{n \to \infty} \frac{E[c(X^n)(\log c(X^n) + O(1))]}{n} = \bar{H}(X)
\]

Theorem 14

The LZ78 algorithm is universal in the class of all stationary and ergodic sources on alphabet \(X\), i.e.,

\[
\lim_{n \to \infty} \frac{1}{n} E l_n(X^n) = \bar{H}(X)
\]

for any stationary and ergodic source \(X_1, X_2, \ldots\) with entropy rate \(\bar{H}(X)\).

Remarks:

- There are mathematically stronger versions of the above theorem as well as rate of convergence results.
- Practical versions of LZ78 are widely used, e.g., in Unix compress, GIF, and TIFF.