CHAPTER 1

Information Theory behind Source Coding

FRANS M. J. WILLEMS
TJALLING J. TJALKENS

1.1 INTRODUCTION

Information theory forms the mathematical basis of both lossy and lossless compression. In this chapter we will look at those aspects of information which are most relevant to lossless compression.

In this section we discuss the concept of entropy. We define entropy, discuss some of its properties, and demonstrate that these properties make it a reasonable information measure. In the next sections we will show that entropy is also an information measure in a more rigorous sense. We do this by proving a "source coding theorem" for discrete memoryless sources.

1.1.1 Definition of Entropy

Assume that X is a discrete random variable that takes values in a finite alphabet $\mathcal{X}$. Let $|\mathcal{X}|$ denote the cardinality of the alphabet $\mathcal{X}$ and suppose that $p(x) := \Pr\{X = x\}$ for $x \in \mathcal{X}$.

Definition 1.1. *The entropy of X is defined by*

$$H(X) := \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}. \tag{1.1}$$

There are two aspects of this definition that need further explanation. First, note that the base of the logarithm is unspecified. Usually, and throughout these chapters, we assume that this base is 2, and we say that the entropy is being measured in *bits* (from binary digits). It is also possible to assume some other base. (If we take e as the base of the logarithm, we obtain the entropy in *nats* (from natural digits).) Second, if $p(x) = 0$, then the term $p(x) \log \frac{1}{p(x)}$ in (1.1) is indeterminate; however, then we define it to be 0. This makes $H(X)$ continuous in the probability distribution, $\{p(x) : x \in \mathcal{X}\}$.

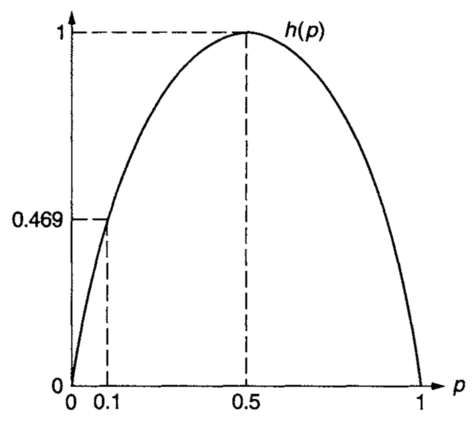

FIGURE 1.1
The binary entropy function.

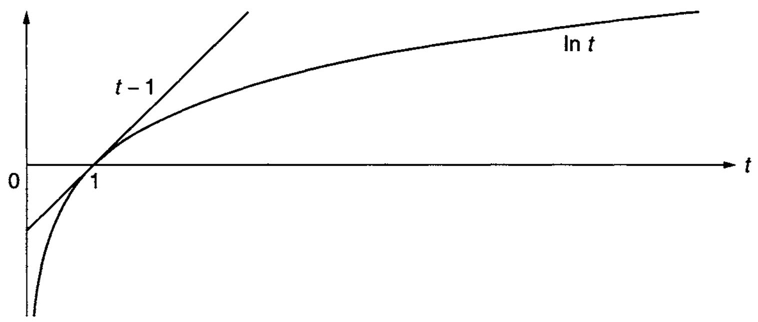

FIGURE 1.2
Geometrical illustration of the inequality $\ln t \leq t - 1$.

Example 1.1. Let X represent the outcome of a single roll with a fair die. Then $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$ and $p(x) = \frac{1}{6}$ for all $x \in \mathcal{X}$. Then, $H(X) = \log 6 = 2.585$ bits.

Example 1.2. Let $X = \{0, 1\}$ and suppose that $p(0) = 1-p$ and $p(1) = p$ for some $0 \leq p \leq 1$. Then $H(X) = h(p)$, where $h(p) := -(1-p)\log(1-p) - p \log p$. The function $h(\cdot)$, which is called the *binary entropy function*, is plotted in Fig. 1.1. It can be seen that $h(0.1) = 0.469$ bits.

1.1.2 Properties of Entropy

If the random variable X takes values in the alphabet $\mathcal{X}$, then

Theorem 1.1. $0 \leq H(X) \leq \log |\mathcal{X}|$. *Furthermore* $H(X) = 0$ *if and only if* $p(x) = 1$ *for some* $x \in \mathcal{X}$, *and* $H(X) = \log |\mathcal{X}|$ *if and only if* $p(x) = \frac{1}{|\mathcal{X}|}$ *for all* $x \in \mathcal{X}$.

Proof. Since each $p(x)$ is ≤ 1 it follows that each term $p(x) \log \frac{1}{p(x)}$ is ≥ 0. So $H(X) \geq 0$. Furthermore $H(X) = 0$ implies that all terms $p(x) \log \frac{1}{p(x)}$ are 0. A term is equal to 0 only if $p(x) = 0$ or $p(x) = 1$. Therefore $H(X) = 0$ only if one $p(x) = 1$ and all the rest are 0.

Before we continue with the second part of the proof we will state the perhaps most frequently used inequality in information theory, namely, $\ln t \leq t - 1$, where $t > 0$. Note that equality occurs only for $t = 1$ (see Fig. 1.2).

The proof of this inequality follows directly from the Taylor expansion around $t = 1$.

$$\ln t = \ln 1 + \frac{d}{d\tau}\ln\tau\bigg|_{\tau=1}(t-1) + \frac{1}{2}\frac{d^2}{d\xi^2}\ln\xi\bigg|_{\xi\in[1,t]}(t-1)^2$$

and with

$$\frac{d}{d\tau}\ln\tau\bigg|_{\tau=1} = \frac{1}{\tau}\bigg|_{\tau=1} = 1,$$

$$\frac{d^2}{d\xi^2}\ln\xi\bigg|_{\xi\in[1,t]} = \frac{-1}{\xi^2}\bigg|_{\xi\in[1,t]} < 0,$$

we find

$$\ln t \le t - 1 \quad \text{with equality only if } t = 1. \tag{1.2}$$

Now let $\mathcal{X}'$ be the set of elements $x \in \mathcal{X}$ for which $p(x) > 0$. Using the inequality $\ln t \le t - 1$ we then obtain

$$\begin{aligned} H(X) - \log|\mathcal{X}'| &= \sum_{x\in\mathcal{X}'} p(x)\log\frac{1}{p(x)} - \sum_{x\in\mathcal{X}'} p(x)\log|\mathcal{X}'| \\ &= \frac{1}{\ln 2}\sum_{x\in\mathcal{X}'} p(x)\ln\frac{1}{|\mathcal{X}'|p(x)} \\ &\le \frac{1}{\ln 2}\sum_{x\in\mathcal{X}'} p(x)\left[\frac{1}{|\mathcal{X}'|p(x)} - 1\right] \\ &= \frac{1}{\ln 2}\left[\sum_{x\in\mathcal{X}'}\frac{1}{|\mathcal{X}'|} - \sum_{x\in\mathcal{X}'} p(x)\right] = 0. \end{aligned} \tag{1.3}$$

From this and

$$\log|\mathcal{X}'| \le \log|\mathcal{X}|, \tag{1.4}$$

it follows easily that $H(X) \le \log|\mathcal{X}|$. Note that the two inequalities (1.3) and (1.4) hold with equality only if $p(x) = \frac{1}{|\mathcal{X}|}$ for all $x \in \mathcal{X}$. ∎

1.1.3 Entropy as an Information Measure

Information should be regarded as the entity that can resolve uncertainty. To see what this means consider a random experiment X that generates a random outcome x. Before starting this experiment we are uncertain about its actual result. The generated output x contains the information that takes away this initial uncertainty. We would like the entropy $H(X)$ to be a measure of the initial uncertainty, or equivalently, of the generated information.

Entropy as defined in (1.1) appears to have some properties that a useful measure of information must have:

- If $p(x) = 1$ for some $x \in \mathcal{X}$, then we know in advance what the output of the experiment will be. There is no initial uncertainty; in other words, the experiment generates no information. Indeed in this case the entropy $H(X)$ is equal to 0.
- If $p(x) = \frac{1}{|\mathcal{X}|}$, for all $x \in \mathcal{X}$, i.e., when all outputs are equally likely, the entropy $H(X) = \log|\mathcal{X}|$. It was pointed out by Hartley in 1928 [5] that for T equiprobable outcomes $\log T$

would be the most natural choice. The reason for this is simple. Suppose we have two random experiments that generate T_1 and T_2 equiprobable outcomes. Then the information generated by the two experiments should be equal to the information generated by the first experiment plus the information generated by the second. The logarithmic function satisfies this requirement since $\log T_1T_2 = \log T_1 + \log T_2$.

- As we have seen in Theorem 1.1 entropy is always greater than or equal to 0 and less than or equal to $\log|\mathcal{X}|$. Indeed uncertainty can be absent, i.e., $H(X) = 0$, but it is rather difficult to think of negative uncertainty. Also it seems reasonable that only an equiprobable random experiment generates the maximal amount of information.

1.1.4 Joint Entropy and Conditional Entropy

Assume that X and Y are discrete random variables that take values in the finite alphabet $\mathcal{X}$ and $\mathcal{Y}$, respectively. Suppose for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ that

$$\begin{aligned} p(x,y) &\triangleq \Pr\{X = x \wedge Y = y\}, \\ p(x) &\triangleq \Pr\{X = x\} = \sum_{y\in\mathcal{Y}} \Pr\{X = x \wedge Y = y\}, \\ p(y) &\triangleq \Pr\{Y = y\} = \sum_{x\in\mathcal{X}} \Pr\{X = x \wedge Y = y\}. \end{aligned} \tag{1.5}$$

Later we will also use

$$p(x|y) \triangleq \Pr\{X = x|Y = y\},$$

and

$$p(y|x) \triangleq \Pr\{Y = y|X = x\}. \tag{1.6}$$

Definition 1.2. *The joint entropy $H(X, Y)$ of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined as*

$$H(X,Y) \triangleq -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y). \tag{1.7}$$

If we consider the pair (X, Y) to be a (vector valued) random variable $Z = (X, Y)$, then this definition is a direct consequence of the entropy formula (1.1).

Definition 1.3. *The conditional entropy of X, given Y, is defined by*

$$H(X|Y) := \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log\frac{p(y)}{p(x,y)}. \tag{1.8}$$

Just like in the entropy definition we assume that $0\log 0 = 0$. Note that $p(x) = 0$ or $p(y) = 0$ implies that $p(x, y) = 0$. Therefore terms for which $p(x, y) = 0$ do not contribute to the conditional entropy.

Example 1.3. Let $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and let $\Pr\{X = 0 \wedge Y = 0\} = \frac{3}{5}$, $\Pr\{X = 0 \wedge Y = 1\} = 0$, $\Pr\{X = 1 \wedge Y = 0\} = \frac{1}{5}$, and $\Pr\{X = 1 \wedge Y = 1\} = \frac{1}{5}$. Then $\Pr\{X = 0\} = \frac{3}{5}$, $\Pr\{X = 1\} = \frac{2}{5}$, $\Pr\{Y = 0\} = \frac{4}{5}$ and $\Pr\{Y = 1\} = \frac{1}{5}$. Now $H(X, Y) = -\frac{3}{5}\log\frac{3}{5} - 0\log 0 - \frac{1}{5}\log\frac{1}{5} - \frac{1}{5}\log\frac{1}{5} = 1.3710$ bits and $H(X|Y) = \frac{3}{5}\log\frac{4/5}{3/5} + \frac{1}{5}\log\frac{4/5}{1/5} + \frac{1}{5}\log\frac{1/5}{1/5} = 0.649$ bits.

1.1.5 Properties of Joint Entropy and Conditional Entropy

Assume that X and Y are discrete random variables that take values in $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Theorem 1.2 (Chain Rule).

$$H(X, Y) = H(X) + H(Y|X). \tag{1.9}$$

Proof. In the following derivation we use the chain rule for distributions $p(x, y) = p(x)p(y|x)$.

$$\begin{aligned}
H(X, Y) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log p(x, y) \\
&= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log p(x)\frac{p(x, y)}{p(x)} \\
&= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x)p(y|x)\log p(x) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x, y)\log\frac{p(x)}{p(x, y)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log p(x) + H(Y|X) \\
&= H(X) + H(Y|X).
\end{aligned}$$

Theorem 1.3. *The conditional entropy $H(X|Y) \geq 0$. $H(X|Y) \leq H(X)$ with equality only if X and Y are independent.*

Proof. The conditional entropy $H(X|Y)$ is non-negative since $p(y) \geq p(x, y)$ always, so the terms in (1.8) are all non-negative.

Using $p(x) = \sum_{y\in\mathcal{Y}} p(x, y)$ we find that

$$\begin{aligned}
H(X|Y) - H(X) &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\log\frac{p(y)}{p(x, y)} - \sum_{x\in\mathcal{X}} p(x)\log\frac{1}{p(x)} \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\log\frac{p(y)}{p(x, y)} - \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\log\frac{1}{p(x)} \\
&= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\log\frac{p(x)p(y)}{p(x, y)}.
\end{aligned} \tag{1.10}$$

Let $\mathcal{S}$ be the set of pairs (x, y) for which $p(x, y) > 0$. Then we continue

$$\begin{aligned}
&= \frac{1}{\ln 2}\sum_{(x,y)\in\mathcal{S}} p(x, y)\ln\frac{p(x)p(y)}{p(x, y)} \\
&\leq \frac{1}{\ln 2}\sum_{(x,y)\in\mathcal{S}} p(x, y)\left[\frac{p(x)p(y)}{p(x, y)} - 1\right] \\
&= \frac{1}{\ln 2}\left[\sum_{(x,y)\in\mathcal{S}} p(x)p(y) - \sum_{(x,y)\in\mathcal{S}} p(x, y)\right] \\
&\leq \frac{1}{\ln 2}\left[\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x)p(y) - \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\right] = 0.
\end{aligned} \tag{1.11}$$

The first inequality holds with equality only if $p(x, y) = p(x)p(y)$ for all (x, y) for which $p(x, y) > 0$. The second inequality holds with equality only if $p(x)p(y) = 0$ for (x, y) for which

$p(x, y) = 0$. Therefore $H(X) = H(X|Y)$ only if $p(x, y) = p(x)p(y)$ for all $x \in \mathcal{X}$ and all $y \in \mathcal{Y}$. ∎

Example 1.4. If we compute $H(X)$ for the random variables X and Y of the previous example in this section, then we get $H(X) = \frac{3}{5}\log\frac{1}{3/5} + \frac{2}{5}\log\frac{1}{2/5} = h(\frac{2}{5}) = 0.971$ bits. Observe that $H(X) - H(X|Y) = 0.971 - 0.649 > 0$ bits.

1.1.6 Interpretation of Conditional Entropy

The conditional entropy $H(X|Y)$ can be interpreted as the (average) amount of information that X contains, after Y has been revealed. To see this, define

$$H(X|Y = y) = \sum_{x\in\mathcal{X}} p(x|y)\log\frac{1}{p(x|y)}, \quad \text{for } y \text{ such that } p(y) > 0; \tag{1.12}$$

then we can rewrite (1.8) as follows:

$$\begin{aligned} H(X|Y) &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y)\log\frac{p(y)}{p(x, y)} \\ &= \sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}} p(x|y)\log\frac{1}{p(x|y)} = \sum_{y\in\mathcal{Y}} p(y)\cdot H(X|Y = y). \end{aligned} \tag{1.13}$$

Now $H(X|Y)$ is just the average of all entropies $H(X|Y = y)$ of X for fixed y, over all $y \in \mathcal{Y}$.

Example 1.5. Again for the random pair X, Y of the previous example note that $\Pr\{X = 0|Y = 0\} = \frac{3}{4}$ and $\Pr\{X = 1|Y = 0\} = \frac{1}{4}$; therefore $H(X|Y = 0) = \frac{3}{4}\log\frac{1}{3/4} + \frac{1}{4}\log\frac{1}{1/4} = h(\frac{1}{4}) = 0.811$ bits. Furthermore $\Pr\{X = 0|Y = 1\} = 0$ and $\Pr\{X = 1|Y = 1\} = 1$; therefore $H(X|Y = 1) = 0\log\frac{1}{0} + 1\log\frac{1}{1} = 0$. For $H(X|Y)$ we get again $H(X|Y) = \frac{4}{5}h(\frac{1}{4}) + \frac{1}{5}0 = 0.649$ bits.

Another conditional entropy that we shall use has the form $H(X, Y|Z)$ for discrete random variables X, Y, and Z with a joint probability distribution $p(x, y, z)$. This entropy describes the amount of information in the pair (X, Y) given that Z is known (revealed). Likewise is $H(X|Y, Z)$, the amount of information in X given the pair (Y, Z).

$$\begin{aligned} H(X, Y|Z) &= -\sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x, y, z)\log\frac{p(x, y, z)}{p(z)} \\ &= -\sum_{z\in\mathcal{Z}} p(z) \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x, y|z)\log p(x, y|z) \\ &= \sum_{z\in\mathcal{Z}} p(z)H(X, Y|Z = z). \end{aligned} \tag{1.14}$$

$$\begin{aligned} H(X|Y, Z) &= -\sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x, y, z)\log\frac{p(x, y, z)}{p(y, z)} \\ &= -\sum_{(y,z)\in\mathcal{Y}\times\mathcal{Z}} p(y, z)\sum_{x\in\mathcal{X}} p(x|y, z)\log p(x|y, z) \\ &= -\sum_{(y,z)\in\mathcal{Y}\times\mathcal{Z}} p(y, z)H(X|Y = y, Z = z). \end{aligned} \tag{1.15}$$

The chain rule (1.2) applies and we find

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z). \tag{1.16}$$

1.2 SEQUENCES AND INFORMATION SOURCES

In this section we introduce some notation concerning sequences. We also describe different kinds of information sources.

1.2.1 Sequences

Let $\mathcal{U}$ be a discrete alphabet. A concatenation $u = u_1u_2 \cdots u_l$ of $l \geq 0$ symbols $u_i \in \mathcal{U}$, where $i = 1, l$ is called a *sequence over* $\mathcal{U}$, and l is the length of this sequence. If $l = 0$, then the sequence is called *empty*; i.e., it contains no symbols. The empty sequence is denoted by ϕ. Let $\mathcal{U}^*$ be the set of all sequences over $\mathcal{U}$ including the empty sequence. The length of a sequence $u \in \mathcal{U}^*$ is denoted by $\lambda(u)$.

If $u' = u'_1u'_2 \cdots u'_{l'}$ and $u'' = u''_1u''_2 \cdots u''_{l''}$ are two sequences over $\mathcal{U}$, then the concatenation $u = u'u''$ of u' and u'' is defined as $u'_1u'_2 \cdots u'_{l'}u''_1u''_2 \cdots u''_{l''}$. It consists of $l' + l''$ symbols. We call u' a *prefix* and u'' a *suffix* of u. Note that the empty sequence ϕ is a prefix and a suffix of all sequences.

1.2.2 Information Sources

Let time, which is denoted by t, assume the integer values $\ldots, -1, 0, 1, 2, \ldots$ and run from $-\infty$ to ∞. At time t a *discrete information source* emits the source symbol x_t (see Fig. 1.3). Each symbol x_t assumes a value in the *source alphabet* $\mathcal{X}$. This alphabet is assumed to be *discrete*. It contains $|\mathcal{X}|$ elements and in general $|\mathcal{X}| \geq 2$. Let i and j be integers satisfying $i \leq j$. Then the concatenation $x_i^j = x_ix_{i+1} \cdots x_j$ of $j - i + 1$ source symbols generated by our source is a sequence over $\mathcal{X}$, called a *source sequence*.

Let T be some integer that will be specified later. It is now our intention to describe the source sequence $x_1^T = x_1x_2 \cdots x_T$ as "efficiently" as possible. We will specify in Section 1.3.1 what we mean by efficiently here. We will often write x^j instead of x_1^j; hence $x^j = x_1x_2 \cdots x_j$ for $j \geq 0$. Furthermore $x^j = \phi$ for $j = 0$.

The set $\mathcal{X}^T$ of all different sequences contains $|\mathcal{X}|^T$ elements, and the source generates the sequence $x^T \in \mathcal{X}^T$ with probability $\Pr\{X^T = x^T\}$. This probability distribution satisfies

$$\sum_{x^T \in \mathcal{X}^T} \Pr\{X^T = x^T\} = 1, \quad \text{and} \quad \Pr\{X^T = x^T\} \geq 0, \quad \text{for all } x^T \in \mathcal{X}^T. \tag{1.17}$$

We will sometimes denote $\Pr\{X^T = x^T\}$ by $P(x^T)$.

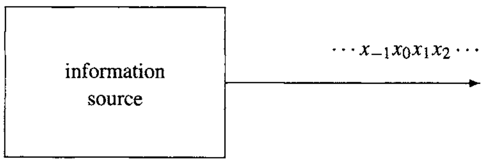

FIGURE 1.3
An information source.

1.2.3 Memoryless Sources

The simplest kind of information source that we can think of is the *memoryless* or *independent and identically distributed* (i.i.d.) source. Independence of the subsequent symbols in the sequence x^T implies that

$$\Pr\{X^T = x^T\} = \prod_{t=1,T} \Pr\{X_t = x_t\}, \quad \text{for } x^T \in \mathcal{X}^T, \tag{1.18}$$

whereas

$$\Pr\{X_t = x\} = p(x), \quad \text{for all } x \in \mathcal{X} \text{ and } t = 1, T \tag{1.19}$$

indicates that all the symbols in the source sequence are identically distributed. The symbol distribution $\{p(x) : x \in \mathcal{X}\}$ satisfies $\sum_{x \in \mathcal{X}} p(x) = 1$ and $p(x) \geq 0$ for all $x \in \mathcal{X}$. In Sections 1.3 and 1.5 we will focus on memoryless sources.

1.2.4 Binary Sources

A *binary* source has source alphabet $\mathcal{X} = \{0, 1\}$ and $|\mathcal{X}| = 2$. The source sequence x^T is now a sequence of binary digits of length T. For a memoryless binary source we have that

$$1 - p(0) = p(1) \triangleq \theta, \tag{1.20}$$

and the statistical behavior of a binary i.i.d. source is therefore completely described by a single parameter θ that specifies the probability that the source generates a "1". So if we consider a sequence x^T with a zeros and $b = T - a$ ones, the probability $\Pr\{X^T = x^T\} = (1 - \theta)^a \theta^b$.

1.2.5 Discrete Stationary Sources

There exists a more general class of sources where the successive source outputs are not independent. In that case we know that due to Theorem 1.3 the entropy of the sequence X^T must be smaller than T times the single-letter entropy. In order to understand these sources it is useful to study the behavior of the entropy for sequences of increasing length.

First we shall discuss briefly the class of sources known as the *stationary sources*. Stationary sources produce infinite-length output strings $\ldots, X_{-2}, X_{-1}, X_0, X_1, X_2, \ldots$ and assign probabilities to arbitrary source output sequences, with the following two restrictions.

1. The probabilities must be *consistent*. This means that if $J = \{i_1, i_2, \ldots, i_n\}$ is a set of n indices, for arbitrary n, and j is an index that is not in J, then for all possible symbol values x_k, for $k = j$ or $k \in J$, the following holds:

$$\begin{aligned}&\Pr\left\{X_{i_1} = x_{i_1} \wedge X_{i_2} = x_{i_2} \wedge \cdots \wedge X_{i_n} = x_{i_n}\right\} \\ &\quad = \sum_{x_j \in \mathcal{X}} \Pr\left\{X_{i_1} = x_{i_1} \wedge X_{i_2} = x_{i_2} \wedge \cdots \wedge X_{i_n} = x_{i_n} \wedge X_j = x_j\right\}.\end{aligned}$$

Example 1.6. An example will clarify this condition. Suppose $\mathcal{X} = \{0, 1, 2\}$ and the source assigns $\Pr\{X_1 = 0 \wedge X_2 = 0\} = \frac{1}{2}$, $\Pr\{X_1 = 0 \wedge X_2 = 1\} = \frac{1}{8}$, and $\Pr\{X_1 = 0 \wedge X_2 = 2\} = \frac{1}{8}$. Then by summing over $X_2 \in \mathcal{X}$ we find $\Pr\{X_1 = 0\} = \frac{3}{4}$. If the source does not assign this probability to $\Pr\{X_1 = 0\}$, it is not consistent.

2. The probabilities depend only on their relative positions, not on the absolute positions. So they are *shift invariant*. If again J is a set of n indices, then

$$\begin{aligned}\Pr\left\{X_{i_1} = x_{i_1} \wedge X_{i_2} = x_{i_2} \wedge \cdots \wedge X_{i_n} = x_{i_n}\right\} \\ = \Pr\left\{X_{i_1+1} = x_{i_1} \wedge X_{i_2+1} = x_{i_2} \wedge \cdots \wedge X_{i_n+1} = x_{i_n}\right\}.\end{aligned}$$

The entropy $H(X^T)$ is usually an unbounded non-decreasing function of T because it is reasonable to assume that for many sources most source outputs contribute to the entropy; i.e., that for an infinite number of source outputs X_i the conditional entropy $H(X_i|X^{i-1})$ is strictly positive. For this reason it is more useful to study the *averaged* per letter entropy defined by

$$H_T(X) \triangleq \frac{1}{T}H(X^T). \tag{1.21}$$

1.2.6 The Entropy Rate

We shall now consider the behavior of $H_T(X)$ as T grows. First note that for a memoryless source with entropy $H(X)$ we have

$$H_T(X) = \frac{1}{T}H(X^T) = \frac{1}{T}TH(X) = H(X). \tag{1.22}$$

So, in this case the *averaged* per letter entropy is indeed the actual per letter entropy.

As it will turn out, the conditional entropy $H(X_T|X^{T-1})$, also known as the *innovation entropy*, plays the same role as $H_T(X)$. The following theorem will describe the behavior of these two entropies.

Theorem 1.4. *For a stationary source $\{X_i\}_{i=-\infty}^{\infty}$ with $H(X_1) < \infty$ holds*

1. $H(X_T|X^{T-1})$ *is non-increasing in T.*
2. $H_T(X) \geq H(X_T|X^{T-1})$ *for all $T = 1, 2, 3, \ldots$.*
3. $H_T(X)$ *is non-increasing in T.*
4. $\lim_{T\to\infty} H_T(X) = \lim_{T\to\infty} H(X_L|X^{T-1})$.

For a proof of this theorem we refer to [2] or [3] where the interested reader can also find a more extended discussion of sources with memory.

Definition 1.4. *The entropy rate $H_\infty(X)$ is defined by*

$$H_\infty(X) \triangleq \lim_{T\to\infty} H_T(X). \tag{1.23}$$

It is possible to prove a source coding theorem that shows that the entropy rate plays the same role for stationary sources as $H(X)$ does for memoryless sources; i.e., it describes the smallest possible code rate for data compression.

1.3 VARIABLE-LENGTH CODES FOR MEMORYLESS SOURCES

In this section we describe the elements in a source coding system and discuss what we want to achieve with such a system. First we investigate codes for single source symbols. We argue that, in order to achieve unique decodability, we can apply prefix codes. We derive the Kraft inequality for prefix codes and use this inequality to show that the expected codeword length of a prefix code cannot be smaller than the entropy of the source. Moreover we show that there exist prefix

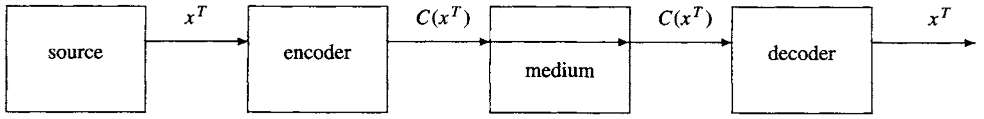

FIGURE 1.4
A source coding system.

codes with an expected codeword length smaller than the source entropy plus 1. Finally we show that codes for large blocks of source symbols achieve an expected codeword length per source symbol arbitrary close to source entropy.

1.3.1 A Source Coding System, Variable-Length Codes for Source Symbols

Suppose that we have a memoryless source with alphabet $\mathcal{X}$ and symbol distribution $\{p(x) : x \in \mathcal{X}\}$. We now want to transmit the sequence $x^T = x_1x_2 \cdots x_T$ generated by the source via a binary medium (or store it into a binary medium). Therefore we must transform the source sequence x^T into a sequence $C(x^T)$ of binary[1] digits called a code sequence from which the original source sequence x^T can be reconstructed. The length of the code sequence is denoted $L(x^T)$. Our notation suggests that $C(\cdot)$ is a mapping from $\mathcal{X}^T$ into the set $\{0, 1\}^*$ of binary sequences of variable lengths. The transformation from a source sequence into a code sequence is performed by an *encoder*; the reverse operation is carried out by the *decoder* (see Fig. 1.4).

We assume that both the encoder and decoder are operating *sequentially*; i.e., the encoder first accepts the first source symbol x_1 of the source sequence x^T as an input, then the second symbol x_2, etc. Using these source symbols it first produces the first binary code digit c_1 of the code sequence $C(x^T)$, then the second binary digit c_2, etc. The first binary digit c_1 in the code sequence $C(x^T)$ is the first input that is applied to the decoder, and the second binary digit c_2 of this code sequence is its second input, etc. The decoder first reconstructs from these code digits the first source symbol x_1, after that the second symbol x_2, etc.

To use the medium efficiently it is important that the transformation is such that the length of the code sequence $C(x^T)$ is *as short as possible.*

In this chapter we only consider transformations of the following kind. We assume that there corresponds a variable-length codeword $c(x) \in \{0, 1\}^*$ with length $l(x)$ to each source symbol $x \in \mathcal{X}$. The mapping $c(\cdot)$ from source symbols in $\mathcal{X}$ into codewords from $\{0, 1\}^*$ is called a *source code*. The code sequence $C(x^T)$ is now the concatenation of the codewords $c(x_t)$ corresponding to all the source symbols x_t for $t = 1, T$; i.e.,

$$C(x^T) = c(x_1)c(x_2) \cdots c(x_T). \tag{1.24}$$

If $L(x^T)$ denotes the length of the code sequence $C(x^T)$, then, by the ergodic theorem (see, e.g., [4] or [11]), we know that with probability 1

$$\lim_{T\to\infty} \frac{L(x^T)}{T} = \lim_{T\to\infty} \frac{l(x_1) + l(x_2) + \cdots + l(x_T)}{T} = \sum_{x\in\mathcal{X}} p_x l(x) = \mathbf{E}[l(X)]. \tag{1.25}$$

Therefore, to achieve short code sequences, we must choose the source code $\{c(x) : x \in \mathcal{X}\}$, such that the expected codeword length $\mathbf{E}[l(X)]$ is as small as possible.

[1] In this chapter we will consider only the case where all code symbols are binary.

Table 1.1

x	$c_1(x)$	$c_2(x)$	$c_3(x)$	$c_4(x)$
a	0	0	00	0
b	0	1	10	10
c	1	00	11	110
d	10	11	110	111

However, to make it possible for the decoder to reconstruct the original source sequence from the code sequence the source code $\{c(x) : x \in \mathcal{X}\}$ also must be *uniquely decodable*. This will be the subject of the next section.

1.3.2 Unique Decodability, Prefix Codes

We start this section with an example.

Example 1.7. Suppose that the source alphabet $\mathcal{X} = \{a, b, c, d\}$. Table 1.1 gives four possible source codes, $c_1(\cdot)$, $c_2(\cdot)$, $c_3(\cdot)$, and $c_4(\cdot)$, for this alphabet. We assume that all symbols $x \in \mathcal{X}$ have positive probability so that there are no codewords for symbols that never occur. Note that we use these codes to encode a sequence of T source symbols.

Note that code $c_1(\cdot)$ has the property that there are two source symbols (a and b) that are represented by the same codeword (0) and the decoder will be confused when it receives this codeword.

The codewords in the second code are all distinct. However, this code is still not good for encoding a *sequence* of source symbols; e.g., the source sequences ac and ca both give the same codeword (000) and again the decoder is confused when receiving this code sequence.

To see how the third code acts, assume that a sequence of codewords is received. The decoder starts from the beginning. If the first two bits in the sequence of codewords are 00 or 10, the first source symbol is a or b, respectively. However, if the first two bits are 11, then the following bits must also be checked. If the third bit is a 1, then the first source symbol is c. If the third bit is 0, the length of the string of 0's starting with this third bit determines the source symbol. An even length can result only from symbol c; an odd length implies that the first codeword is 110 and thus the source symbol is d. Note that eventually after having seen many bits in the sequence of codewords we will be able to determine the first source symbol. Therefore this third code is *uniquely decodable.*

The fourth code is uniquely decodable too but is also *instantaneous*. Observe that no codeword in this code is the prefix of any other codeword in the code. Therefore we call this code a *prefix code*. Now, if the decoder has received a sequence so far which is a codeword, it can immediately output the corresponding source symbol since no other codeword has the received sequence as a prefix.

In this chapter, instead of considering all uniquely decodable codes, we will consider only prefix codes. The reason behind this choice will be given at the end of this section. First we give a definition.

Definition 1.5. *A code is called a prefix code or an instantaneous code if no codeword is the prefix of any other codeword.*

Codewords in a prefix code can be regarded as leaves in a code tree; see Fig. 1.5. Using this tree it is easy to parse a code sequence uniquely into the codewords; e.g., $00101111011\cdots$ is parsed into $0, 0, 10, 111, 10, 11, \ldots$.

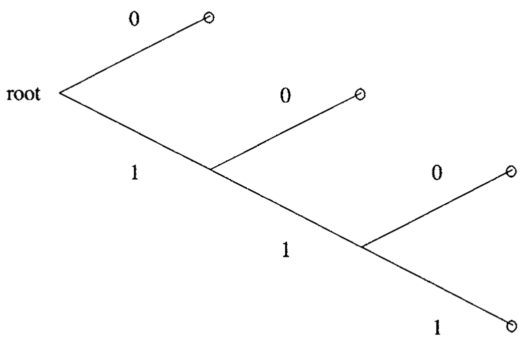

FIGURE 1.5
The code tree corresponding to a code with codewords 0, 10, 110, and 111. The path from the root to a leaf determines the codeword.

Note that prefix codes are uniquely decodable (but not all uniquely decodable codes are prefix codes). If we now return to our original problem, describing source sequences efficiently, we know from (1.25) that we must minimize the expected codeword length $\mathbf{E}[l(X)]$. If we restrict ourselves to prefix codes, we must find out what the smallest possible $\mathbf{E}[l(X)]$ is that can be achieved with prefix codes.

The third code in the example did not satisfy the prefix condition. Nevertheless it was possible to decode this code as we have seen. The third code is *uniquely decodable.* These codes are treated extensively by Cover and Thomas [2, Chapter 5]. Here we mention only that every uniquely decodable code can be changed into a prefix code without changing the codeword lengths $l(x)$, $x \in \mathcal{X}$. This implies that, in general, there is no good reason to use codes other than prefix codes.

1.3.3 Kraft's Inequality for Prefix Codes and Its Counterpart

In order to find the smallest possible expected codeword length $\mathbf{E}[l(X)]$ that can be achieved with prefix codes, the Kraft inequality is of crucial importance.

Theorem 1.5. *The codeword lengths $l(x)$ of any binary prefix code for the source symbols $x \in \mathcal{X}$ must satisfy Kraft's inequality, which says that*

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1. \tag{1.26}$$

There is also a Kraft inequality for non-binary code alphabets. For a D-ary prefix code the inequality $\sum_{x \in \mathcal{X}} D^{-l(x)} \leq 1$ holds. We consider only binary codes here.

Proof. Consider Fig. 1.6. Let $l_{\max} = \max_{x \in \mathcal{X}} l(x)$. In the tree representing the code, a codeword $c(x)$ with length $l(x)$ has $2^{l_{\max}-l(x)}$ descendants at level $l_{\max}$. Because of the prefix condition no codeword is an ancestor of any other codeword in the tree. Therefore the descendants at level $l_{\max}$ of all codewords must be distinct and there are $2^{l_{\max}}$ nodes at level $l_{\max}$; hence

$$\sum_{x \in \mathcal{X}} 2^{l_{\max}-l(x)} \leq 2^{l_{\max}}. \tag{1.27}$$

Dividing both sides of (1.27) by $2^{l_{\max}}$ completes the proof. ∎

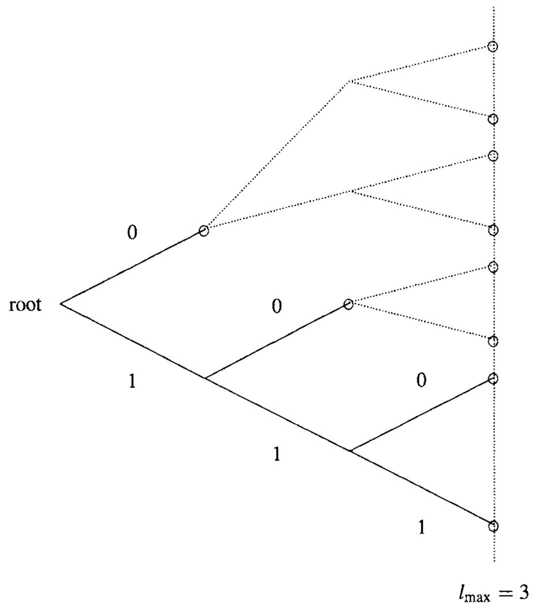

FIGURE 1.6
The prefix code with codewords 0, 10, 110, and 111 and their descendants at level $l_{\max} = 3$.

Example 1.8. For the prefix code with codewords 0, 10, 110, and 111 we get the Kraft sum

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1. \tag{1.28}$$

This Kraft sum satisfies the Kraft inequality with equality.

In the Kraft theorem (1.5) we have shown that codeword lengths in any prefix code satisfy Kraft's inequality. There is also a counterpart to the Kraft theorem. This is our next result.

Theorem 1.6. *If the codeword lengths $l(x) \in \mathcal{X}$ for symbols $x \in \mathcal{X}$ satisfy Kraft's inequality (1.26), there exists a binary prefix code with codewords $c(x)$ that have length $l(x)$.*

Proof. Assume without loss of generality that $\mathcal{X} = \{1, 2, \ldots, M\}$ with $M \geq 2$ and that

$$l(1) \leq l(2) \leq \cdots \leq l(M). \tag{1.29}$$

We start with a full tree with depth $l_{\max} = \max_{x \in \mathcal{X}} l(x)$; this tree has $2^{l_{\max}}$ leaves. The first codeword $c(1)$ always can be placed in the tree. Next suppose that $m < M$ codewords have already been placed in the tree. Then

$$\sum_{x=1}^{m} 2^{-l(x)} + \sum_{x=m+1}^{M} 2^{-l(x)} \leq 1. \tag{1.30}$$

The second term in (1.30) is positive; hence the first term is strictly less than 1. This first term is a multiple of $2^{-l(m)}$ and is therefore less than or equal to $1 - 2^{-l(m)}$, so there still exists a node at level $l(m)$. Therefore a codeword $c(m + 1)$ with length $l(m + 1) \geq l(m)$ definitively fits in the tree. Continuing like this, all codewords get their place in the tree. Note that the path from the root of the tree to a codeword node determines the codeword. ∎

Kraft's inequality and its counterpart play an important role in the analysis and design of codes for (memoryless) sources as we will see in the following sections.

1.3.4 Redundancy, Entropy, and Bounds

Remember that we are still discussing variable-length codes for symbols generated by memoryless sources. The source distribution is $\{p(x) : x \in \mathcal{X}\}$ and a code is described by $\{c(x) : x \in \mathcal{X}\}$.

Definition 1.6. *We define the individual redundancy $\rho(x)$ of the symbol $x \in \mathcal{X}$, whose codeword $c(x)$ has length $l(x)$ as*

$$\rho(x) \triangleq l(x) - \log_2 \frac{1}{p(x)}, \tag{1.31}$$

and the expected redundancy of a code as

$$\mathbf{E}[\rho(X)] \triangleq \sum_{x \in \mathcal{X}} p(x)\rho(x). \tag{1.32}$$

Redundancies are measured in bits.[2]

The term $\log_2(1/p(x))$ in the definition of individual redundancy is called *the ideal codeword length.* The average ideal codeword length is equal to the source entropy $H(X)$.

For the expected redundancy we can show that

$$\begin{aligned}\mathbf{E}[\rho(X)] &= \sum_{x \in \mathcal{X}} p(x)\rho(x)\\ &= \sum_{x \in \mathcal{X}} p(x)l(x) - \sum_{x \in \mathcal{X}} p(x)\log_2 \frac{1}{p(x)} = \mathbf{E}[l(X)] - H(X).\end{aligned} \tag{1.33}$$

If we take

$$l(x) = \left\lceil \log_2 \frac{1}{p(x)} \right\rceil \quad \text{for all possible } x \text{ with } p(x) > 0, \tag{1.34}$$

where $\lceil a \rceil$ is the smallest integer larger than a, then

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} = \sum_{x \in \mathcal{X}} 2^{-\left\lceil \log_2 \frac{1}{p(x)} \right\rceil} \leq \sum_{x \in \mathcal{X}} 2^{-\log_2 \frac{1}{p(x)}} = \sum_{x \in \mathcal{X}} p(x) = 1; \tag{1.35}$$

hence there exists a prefix code with codeword lengths $l(x) = \lceil \log_2(1/p(x)) \rceil$ for all possible x with $p(x) > 0$ by Theorem 1.6. For this codeword length assignment we obtain the following upper bound for the individual redundancy:

$$\rho(x) = l(x) - \log_2 \frac{1}{p(x)} = \left\lceil \log_2 \frac{1}{p(x)} \right\rceil - \log_2 \frac{1}{p(x)} < 1. \tag{1.36}$$

Note that this bounds holds for all symbols $x \in \mathcal{X}$ with $p(x) > 0$. Averaging over $x \in \mathcal{X}$ immediately leads to the following statement:

[2] Logarithms have base 2, here and in the next chapters.

Theorem 1.7. *For symbols generated by a source according to the probability distribution $\{p(x) : x \in \mathcal{X}\}$, there exists a prefix code that achieves*

$$\mathbf{E}[\rho(X)] < 1, \quad \textit{or equivalently} \quad \mathbf{E}[l(X)] < H(X) + 1. \tag{1.37}$$

This theorem tells us that codes with expected codeword length $\mathbf{E}[l(X)]$ less than $H(X) + 1$ exist. An obvious question is now: "Can we find better codes than those described by 1.34?" The next theorem gives at least a partial answer to this question. It says that there can exist only slightly better codes.

Theorem 1.8. *For any prefix code for the source symbols generated by a source with probability distribution $\{p(x) : x \in \mathcal{X}\}$,*

$$\mathbf{E}[\rho(X)] \geq 0, \quad \textit{or equivalently,} \quad \mathbf{E}[l(X)] \geq H(X), \tag{1.38}$$

where equality occurs if and only if $l(x) = \log_2(1/p(x))$, i.e., if the codeword lengths are equal to the ideal codeword lengths, for all $x \in \mathcal{X}$.

Proof. Consider the following chain of (in)equalities. The summations are only over symbols $x \in \mathcal{X}$ with $p(x) > 0$. We now obtain

$$\begin{aligned}\mathbf{E}[\rho(X)] &= \sum_{x\in\mathcal{X}} p(x)l(x) - \sum_{x\in\mathcal{X}} p(x)\log_2\frac{1}{p(x)} \\ &= -\frac{1}{\ln 2}\sum_{x\in\mathcal{X}} p(x)\ln\frac{2^{-l(x)}}{p(x)} \\ &\geq -\frac{1}{\ln 2}\sum_{x\in\mathcal{X}} p(x)\left(\frac{2^{-l(x)}}{p(x)} - 1\right) = \frac{1}{\ln 2}\left(\sum_{x\in\mathcal{X}} p(x) - \sum_{x\in\mathcal{X}} 2^{-l(x)}\right) \geq 0. \end{aligned} \tag{1.39}$$

In this proof the first inequality holds since $\ln a \leq a - 1$ for $a > 0$. The second inequality is a consequence of Kraft's inequality, which holds for prefix codes (see Theorem 1.5). Note (see Fig. 1.2) that equality can occur only if $a = 1$ or equivalently $l(x) = \log_2(1/p(x))$ for all $x \in \mathcal{X}$. ∎

From Theorem 1.7 and Theorem 1.8 we can conclude that for a source with entropy $H(X)$ there exists a binary prefix code such that

$$H(X) \leq \mathbf{E}[l(X)] < H(X) + 1. \tag{1.40}$$

1.3.5 Variable-Length Codes for Blocks of Symbols

In Section 1.3.1 we described a system in which each source symbol $x_t, t = 1, 2, \ldots$ was encoded separately. We have investigated the properties of such a system and we could show (see (1.40)) that there exists a prefix code with expected codeword length $\mathbf{E}[l(X)]$ not more than one binary digit per source symbol larger than the symbol entropy $H(X)$. For small symbol entropies this is not a very exciting result. A better result can be obtained by *blocking*. In a blocking system, blocks of T source symbols are encoded together into a single codeword from a prefix code.

Consider a prefix code that maps source sequence $x^T = x_1x_2\cdots x_T$ onto codeword $C(x^T)$ having length $L(x^T)$ for all $x^T \in \mathcal{X}^T$. Then it is not a big surprise that

$$\mathbf{E}[L(X^T)] \geq H(X^T), \tag{1.41}$$

for any prefix code for source blocks of length T. However, there exists a prefix code that achieves

$$\mathbf{E}[L(X^T)] < H(X^T) + 1. \tag{1.42}$$

For memoryless sources, see Theorems 1.2 and 1.3.

$$H(X^T) = H(X_1X_2\cdots X_T) = H(X_1) + H(X_2) + \cdots + H(X_T) = TH(X). \tag{1.43}$$

Therefore the expected code sequence length per source symbol for any prefix code for a block of T source symbols satisfies

$$\frac{\mathbf{E}[L(X^T)]}{T} \geq H(X), \tag{1.44}$$

but there exists a prefix code that achieves

$$\frac{\mathbf{E}[L(X^T)]}{T} < H(X) + \frac{1}{T}. \tag{1.45}$$

Consequently

$$\lim_{T\to\infty} \frac{\mathbf{E}[L(X^T)]}{T} = H(X); \tag{1.46}$$

in other words, $H(X)$ is the smallest possible number of codebits needed to describe a source symbol of a source with symbol entropy. This demonstrates the importance of the notion of entropy.

1.4 VARIABLE-LENGTH CODES FOR SOURCES WITH MEMORY

In this section we consider codes for source with memory. First we consider codes for blocks of source symbols. The practical value of these codes is not very large. Therefore, we also discuss the Elias algorithm, which is the basis of all arithmetic codes. An arithmetic encoder computes the code sequence from the source sequence and the corresponding decoder computes the source sequence from the code sequence. We demonstrate that the performance need not be much worse than that of any prefix code for a block of source symbols.

1.4.1 Block Codes Again

For sources with memory we can also use blocking to increase the efficiency of the coded representation. In a blocking system, blocks of T source symbols are encoded together into a single codeword from a prefix code. Consider a prefix code that maps source sequence $x^T = x_1x_2\cdots x_T$ onto codeword $C(x^T)$ having length $L(x^T)$ for all $x^T \in \mathcal{X}^T$. Then it is not a big surprise that

$$\mathbf{E}[L(X^T)] \geq H(X^T), \tag{1.47}$$

for any prefix code for source blocks of length T. However, there exists a prefix code that achieves

$$\mathbf{E}[L(X^T)] < H(X^T) + 1. \tag{1.48}$$

Again, as in (1.44)–(1.46) we consider the expected code sequence length per source symbol and obtain

$$\lim_{T\to\infty} \frac{\mathbf{E}[L(X^T)]}{T} = H_\infty(X). \tag{1.49}$$

From this we conclude that in general we should prefer large values of T.

Note, however, that it is a huge job to design and implement a prefix code for larger values of T. For example, for binary sources and $T = 30$ we get more than a billion codewords! A better approach is described in the next section.

1.4.2 The Elias Algorithm

1.4.2.1 Introduction to the Elias Algorithm

An *arithmetic* encoder *computes* the code sequence $C(x^T)$ that corresponds to the actual source sequence x^T. The corresponding decoder reconstructs the actual source sequence from this code sequence again by performing simple computations. Such an approach differs from encoding and decoding using tables. These tables need to be constructed first and then stored in memory. Arithmetic coding does not require construction and storage of the complete code before the encoding and decoding process is begun. On the other hand, we see that arithmetic coding requires some computational effort during encoding and decoding.

Using arithmetic codes it is possible to process long source sequences. Large values of the length T reduce the redundancy per source symbol. All arithmetic codes are based on the Elias algorithm, which was unpublished but described by Abramson [1] and Jelinek [6].

The first idea behind the Elias algorithm is that to each source sequence x^T there corresponds a *subinterval* of the unit interval $[0, 1)$. This principle can be traced back to Shannon [10].

The second idea, due to Elias, is to order the sequences x^t of length t *lexicographically*, for $t = 1, \dots, T$. For two sequences x^t and $\tilde{x}^t$ we have that $x^t < \tilde{x}^t$ if and only if there exists a $\tau \in \{1, 2, \dots, t\}$ such that $x_i = \tilde{x}_i$ for $i = 1, 2, \dots, \tau - 1$ and $x_\tau < \tilde{x}_\tau$. This lexicographical ordering makes it possible to compute the subinterval corresponding to a source sequence sequentially (recursively).

To make things simple we assume in this section that the source alphabet is binary; hence $\mathcal{X} = \{0, 1\}$.

1.4.3 Representation of Sequences by Intervals

Let distribution $P_c(x^t)$, $x^t \in \{0, 1\}^t$, $t = 0, 1, \dots, T$ be what we shall call the *coding distribution*. It is known to both encoder and decoder. We require that

$$\begin{aligned}
&P_c(\phi) = 1, \\
&P_c(x^{t-1}) = P_c\left(x^{t-1}, X_t = 0\right) + P_c\left(x^{t-1}, X_t = 1\right), \quad \text{for all } x^{t-1} \in \{0, 1\}^{t-1}, t = 1, \dots, T, \\
&P_c(x^T) > 0 \quad \text{for all } x^T \in \{0, 1\}^T \text{ with } \Pr\{X^T = x^T\} > 0. \qquad (1.50)
\end{aligned}$$

To each source sequence there now corresponds a subinterval of $[0, 1)$.

Definition 1.7. *The interval $I(x^t)$ corresponding to $x^t \in \{0, 1\}^t$, $t = 0, 1, \dots, T$ is defined as*

$$I(x^t) \triangleq [B(x^t), B(x^t) + P_c(x^t)), \qquad (1.51)$$

where $B(x^t) \triangleq \sum_{\tilde{x}^t < x^t} P_c(\tilde{x}^t)$.

Note that for $t = 0$ we have that $P_c(\phi) = 1$ and $B(\phi) = 0$ (the only sequence of length 0 is ϕ itself), and consequently $I(\phi) = [0, 1)$. Observe that for any fixed value of t, $t = 0, 1, \dots, T$, all intervals $I(x^t)$ are disjoint, and their union is $[0, 1)$. Each interval has a length equal to the corresponding coding probability.

Just like all source sequences, each code sequence $C = c_1 \cdots c_L$ can be associated with a subinterval of $[0, 1)$.

Definition 1.8. *The interval $J(c_1 \cdots c_L)$ corresponding to the code sequence $c_1 \cdots c_L$ is defined as*

$$J(c_1 \cdots c_L) \triangleq [F(c_1 \cdots c_L), F(c_1 \cdots c_L) + 2^{-L}), \tag{1.52}$$

with $F(c_1 \cdots c_L) \triangleq \sum_{l=1,L} c_l 2^{-l}$.

To understand this, note that the code sequence $c_1 \cdots c_L$ can be considered as a binary fraction $F(c_1 \cdots c_L)$. Since $c_1 \cdots c_L$ is followed by other code sequences, the decoder receives a stream of code digits from which only the first L digits correspond to $c_1 \cdots c_L$. The decoder can determine the value that is represented by the binary fraction formed by the total stream $c_1 c_2 \cdots c_L c_{L+1} \cdots$; i.e.,

$$F_\infty \triangleq \sum_{l=1,\infty} c_l 2^{-l}, \tag{1.53}$$

where it should be noted that the length of the total stream is not necessarily infinite. Since $F(c_1 \cdots c_L) \leq F_\infty < F(c_1 \cdots c_L) + 2^{-L}$ we may say that the interval $J(c_1 \cdots c_L)$ corresponds to the code sequence $c_1 \cdots c_L$.

1.4.3.1 Compression

To compress a sequence x^T, we search for a short code sequence $c_1 \cdots c_L$ whose code interval $J(c_1 \cdots c_L)$ is contained in the sequence interval $I(x^T)$. This is the main principle in arithmetic coding.

Definition 1.9. *The codeword $c_1 \cdots c_L$ for source sequence x^T consists of*

$$L \triangleq \left\lceil \log_2 \frac{1}{P_c(x^T)} \right\rceil + 1 \tag{1.54}$$

binary digits such that

$$F(c_1 \cdots c_L) \triangleq \lceil B(x^T) \cdot 2^L \rceil \cdot 2^{-L}. \tag{1.55}$$

Note that we need to consider only sequences x^T with $\Pr\{X^T = x^T\} > 0$. *They have $P_c(x^T) > 0$.*

Since

$$F(c_1 \cdots c_L) \geq B(x^T) \tag{1.56}$$

and

$$F(c_1 \cdots c_L) + 2^{-L} < B(x^T) + 2^{-L} + 2^{-L} \leq B(x^T) + P_c(x^T), \tag{1.57}$$

we may conclude that $J(c_1 \cdots c_L) \subseteq I(x^T)$, and therefore $F_\infty \in I(x^T)$. Since the intervals $I(x^T)$ for $x^T \in \{0, 1\}^T$ are all disjoint, the decoder can reconstruct the source sequence x^T from F_∞. Note that after this reconstruction, the decoder can compute the code sequence $c_1 \cdots c_L$, just like the encoder, and find the location of the first digit of the *next* code sequence. Note also that, since all code intervals are disjoint, no code sequence is the prefix of any other code sequence. Thus the Elias code satisfies the prefix condition. From (1.55) we obtain the following result:

Theorem 1.9. *The arithmetic code that we have just described achieves code sequence lengths $L(x^T)$ that satisfy*

$$L(x^T) < \log_2 \frac{1}{P_c(x^T)} + 2, \tag{1.58}$$

for all $x^T \in \{0, 1\}^T$ with $\Pr\{X^T = x^T\} > 0$.

Note that the difference between the code sequence length $L(x^T)$ and $\log(1/P_c(x^T))$ is always less than 2 bits. We say that the *individual coding redundancy* is less than 2 bits. We immediately see that this results in an upper bound of 2 bits for the expected redundancy.

1.4.3.2 Sequential Computation

The lexicographical ordering over the source sequences makes it possible to compute the interval $I(x^T)$ sequentially. To do so, we transform the starting interval $I(\phi) = [0, 1)$ into $I(x_1)$, $I(x_1x_2), \dots,$ and $I(x_1x_2 \cdots x_T)$, respectively. The consequence of the lexicographical ordering over the source sequences is that

$$\begin{aligned} B(x^t) &= \sum_{\tilde{x}^t < x^t} P_c(x^t) \\ &= \sum_{\tilde{x}^{t-1} < x^{t-1}} P_c(\tilde{x}^{t-1}) + \sum_{\tilde{x}_t < x_t} P_c(x^{t-1}, \tilde{x}_t) \\ &= B(x^{t-1}) + \sum_{\tilde{x}_t < x_t} P_c(x^{t-1}, \tilde{x}_t). \end{aligned} \tag{1.59}$$

In other words $B(x^t)$ can be computed from $B(x^{t-1})$ and $P_c(x^{t-1}, X_t = 0)$. Therefore the encoder and the decoder can easily find $I(x^t)$ after having determined $I(x^{t-1})$, if it is "easy" to determine probabilities $P_c(x^{t-1}, X_t = 0)$ and $P_c(x^{t-1}, X_t = 1)$ after having processed $x_1x_2 \cdots x_{t-1}$. Observe that when the symbol x_t is being processed, the source interval $I(x^{t-1}) = [B(x^{t-1}), B(x^{t-1}) + P_c(x^{t-1}))$ is subdivided into two subintervals

$$I(x^{t-1}, X_t = 0) = [B(x^{t-1}), B(x^{t-1}) + P_c(x^{t-1}, X_t = 0))$$

and

$$I(x^{t-1}, X_t = 1) = [B(x^{t-1}) + P_c(x^{t-1}, X_t = 0), B(x^{t-1}) + P_c(x^{t-1})). \tag{1.60}$$

The encoder proceeds with one of these subintervals depending on the actual symbol x_t; therefore $I(x^t) \subseteq I(x^{t-1})$. This implies that

$$I(x^T) \subseteq I(x^{T-1}) \subseteq \cdots \subseteq I(\phi). \tag{1.61}$$

The decoder determines from F_∞ the source symbols $x_1, x_2, \dots, x_T$, respectively, by comparing F_∞ with thresholds $D(x^{t-1})$.

Definition 1.10. *The thresholds $D(x^{t-1})$ are defined as*

$$D(x^{t-1}) \triangleq B(x^{t-1}) + P_c(x^{t-1}, X_t = 0), \tag{1.62}$$

for $t = 1, 2, \dots, T$.

Observe that threshold $D(x^{t-1})$ splits up the interval $I(x^{t-1})$ in two parts (see (1.60)). It is the upper boundary point of $I(x^{t-1}, X_t = 0)$ but also the lower boundary point of $I(x^{t-1}, X_t = 1)$.

Since $F_\infty \in I(x^T) \subseteq I(x^t)$, we have for $D(x^{t-1})$ that

$$\begin{aligned} F_\infty &< B(x^t) + P_c(x^t) \\ &= B(x^{t-1}) + P_c(x^{t-1}, X_t = 0) \\ &= D(x^{t-1}) \quad \text{if } x_t = 0, \end{aligned} \tag{1.63}$$

and

$$\begin{aligned} F_\infty &\geq B(x^t) \\ &= B(x^{t-1}) + P_c(x^{t-1}, X_t = 0\} \\ &= D(x^{t-1}) \quad \text{if } x_t = 1. \end{aligned} \tag{1.64}$$

Therefore the decoder can easily find x_t by comparing F_∞ with the threshold $D(x^{t-1})$. Consequently the decoder can also operate sequentially.

We conclude this section with the observation that the Elias algorithm combines an acceptable coding redundancy with a desirable *sequential* implementation. The number of operations is *linear* in the source sequence length T. It is *crucial*, however, that the encoder and decoder have access to or can easily determine the probabilities $P_c(x^{t-1}, X_t = 0)$ and $P_c(x^{t-1}, X_t = 1)$ after having processed $x_1 x_2 \cdots x_{t-1}$. If we accept a loss of at most 2 bits of coding redundancy, we are left with the problem of finding good, sequentially available, coding distributions $P_c(x^T)$.

Example 1.9. Assume that $T = 3$ and the source emits the sequence $x^T = 011$. The coding distribution is as in Fig. 1.7 in which (some of) the conditional coding probabilities are shown.

An arithmetic encoder now computes the codeword for $x^T = 011$ as follows. From $B(\phi) = 0$ the encoder first computes $B(0) = B(\phi) = 0$ and $P_c(0) = 0.7$. Then $B(01) = B(0) + P_c(00) = 0 + P_c(0) \cdot 0.5 = 0.7 \cdot 0.5 = 0.35$ and $P_c(01) = P_c(0) \cdot 0.5 = 0.7 \cdot 0.5 = 0.35$. Finally $B(011) = B(01) + P_c(010) = 0.35 + P_c(01) \cdot 0.9 = 0.35 + 0.35 \cdot 0.9 = 0.665$ and $P_c(011) = P_c(01) \cdot 0.1 = 0.35 \cdot 0.1 = 0.035$. Now the codeword can be determined. The length $L = \lceil \log(1/0.035) \rceil + 1 = 6$ and therefore c^L is the binary expansion of $\lceil 0.665 \cdot 64 \rceil / 64 = 43/64$, which is 101011. See Fig. 1.8.

To see how the decoder operates note that $\frac{43}{64} \leq F_\infty < \frac{44}{64}$. First the decoder compares F_∞ with $D(\phi) = B(\phi) + P_c(0) = 0 + 0.7 = 0.7$. Clearly $F_\infty < \frac{44}{64} < 0.7$ and therefore the decoder finds $x_1 = 0$. Just like the encoder, the decoder computes $B(0)$ and $P_c(0)$. The next threshold is $D(0) = B(0) + P_c(00) = 0 + P_c(0) \cdot 0.5 = 0.7 \cdot 0.5 = 0.35$. Now $F_\infty \geq \frac{43}{64} > 0.35$ and $x_2 = 1$. The decoder computes $B(01)$ and $P_c(01)$ and the next threshold $D(01) = B(01) + P_c(010) = 0.35 + P_c(01) \cdot 0.9 = 0.35 + 0.35 \cdot 0.9 = 0.665$. Again $F_\infty \geq \frac{43}{64} > 0.665$ and the decoder produces $x_3 = 1$. After computation of $B(011)$ and $P_c(011)$ the decoder can reconstruct the

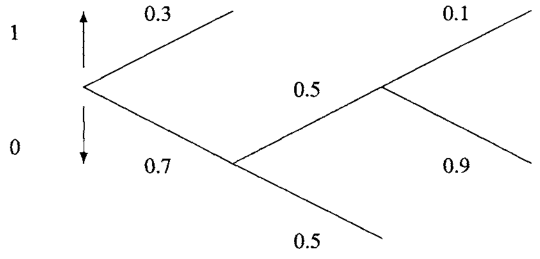

FIGURE 1.7
A graph with some conditional coding probabilities, e.g., $P_c(X_1 = 0) = 0.7$, $P_c(X_2 = 1|X_1 = 0) = 0.5$, and $P_c(X_3 = 1|X_1 = 0, X_2 = 1) = 0.1$.

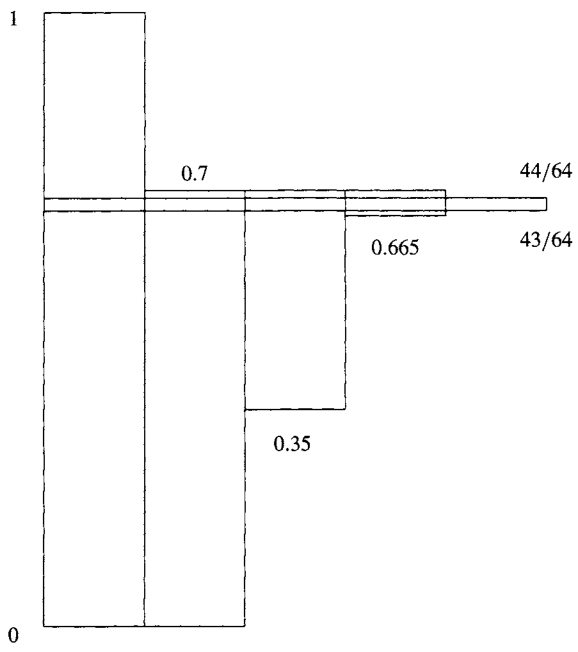

FIGURE 1.8
Subintervals corresponding to the coding of 011.

codeword 101011. Note that for this example the coding redundancy is $6 - \log(1/0.035) = 1.163$ bits.

1.4.3.3 Achieving Entropy

In the previous subsections we have seen that code sequence lengths approximately equal to $-\log_2 P_c(x^T)$ are achievable if we use the Elias algorithm. The question now is how the coding distribution should be chosen. We consider only the case here when the probability distribution over the source sequences is known. In that case we can take

$$P_c(x^T) = \Pr\{X^T = x^T\} = P(x^T), \quad \text{for all } x^T \in \mathcal{X}^T. \tag{1.65}$$

What we achieve for each source sequence x^T with $P(x^T) > 0$ by doing so is

$$L(x^T) < \log_2 \frac{1}{P_c(x^T)} + 2 = \log_2 \frac{1}{P(x^T)} + 2; \tag{1.66}$$

hence the code sequence length is not more than 2 digits larger than the *ideal code sequence length*. Therefore the following bound for the expected codeword length is obtained:

$$\begin{aligned} \mathbf{E}[L(X^T)] &= \sum_{x^T \in \mathcal{X}^T} P(x^T) L(x^T) \\ &< \sum_{x^T \in \mathcal{X}^T} P(x^T) \left(\log_2 \frac{1}{P(x^T)} + 2 \right) = H(X^T) + 2. \end{aligned} \tag{1.67}$$

If we compare upper bound (1.67) with upper bound (1.48) we see that the difference is only one binary digit. Note that the lower bound (1.47) also holds for expected code sequence length corresponding to the Elias method.

Theorem 1.10. *The Elias method yields a prefix code with*

$$H(X^T) \leq \mathbf{E}[L(X^T)] < H(X^T) + 2. \tag{1.68}$$

This is achieved by taking the actual probability distribution of the source sequences as coding distribution.

1.4.3.4 A Special Case

We assume that the source probabilities $P(x^T)$ satisfy a special condition, namely, that for all sequences x^T holds

$$P(x^T) = 2^{-i(x^T)}, \quad \text{for positive integers } i(x^T). \tag{1.69}$$

Such a probability distribution is called a *dyadic distribution.*

The first consequence of a dyadic distribution is that $-\log_2 P(x^T)$ is an integer and thus the upward rounding in (1.54) is not necessary.

If we also assume that the probabilities are ordered in non-increasing order, then $B(x^T) \cdot \frac{1}{P(x^T)}$ is a non-negative integer. This is easy to see if we start from the definition of $B(x^T)$ just below (1.51).

Let $\ell = -\log_2 P(x^T)$, so ℓ is a positive integer. We write

$$\begin{aligned} B(x^T) \cdot 2^\ell &= \sum_{\tilde{x}^T < x^T} P(\tilde{x}^T) 2^\ell, \\ &= \sum_{\tilde{x}^T < x^T} 2^{-i(x^T)} 2^\ell, \\ &= \sum_{\tilde{x}^T < x^T} 2^{\ell - i(x^T)}. \end{aligned} \tag{1.70}$$

Because the probabilities are ordered in non-increasing order we know that for all $\tilde{x}^T$ with $\tilde{x}^T < x^T$, $\ell - i(x^T) \geq 0$ and so $B(x^T) \cdot 2^\ell$ is a sum of positive integers. This implies that the upward rounding in (1.55) is not needed and we find

$$F(c_1 \cdots c_L) = B(x^T), \tag{1.71}$$

$$F(c_1 \cdots c_L) + 2^{-\ell} = B(x^T) + P(x^T). \tag{1.72}$$

So we can conclude that the code sequences with lengths $-\log_2 P(x^T)$ form a prefix code. Clearly the expected code sequence length of this code equals the entropy $H(X^T)$ and thus this code realizes the Shannon lower bound.

Note that if the source is memoryless, the lexicographical order that is required by the Elias algorithm does not satisfy the above non-increasing order property.

In universal source coding problems the actual probability distribution of the source sequences is unknown. Then clever choices of the coding distribution will lead to expected code sequence lengths that are not much larger than what we have found here.

1.4.3.5 Accuracy Issues

At the end of this section we mention that we have avoided discussing accuracy issues. These issues were first investigated by Rissanen [9], who has contributed a lot more to arithmetic coding, and (independently) by Pasco [8]. More details and practical implementations of the arithmetic coding algorithm can be found in Chapter 5.

1.4.4 Competitive Optimality

The Elias algorithm realizes an expected code sequence length close to the minimal expected code sequence length. This of course does not imply that the Elias algorithm gives the shortest code sequence lengths for all source sequences x^T. A better question to ask is how likely is it that another code produces code sequences that are (much) smaller than those of the Elias code.

With the code sequence lengths from (1.54) we can find the following result.

Theorem 1.11. *Let $L(x^T)$ be the code sequence length of the Elias code and let $L'(x^T)$ be the code sequence length of any other (decodable) code. Also, let c be an arbitrary positive constant; then*

$$\Pr\{L(X^T) \geq L'(X^T) + c\} \leq 2^{-c+2}. \tag{1.73}$$

Proof.

$$\begin{aligned}
\Pr\{L(X^T) \geq L'(X^T) + c\} &= \Pr\{\lceil -\log_2 P(X^T)\rceil \geq L'(X^T) + c - 1\} &(1.74)\\
&\leq \Pr\{-\log_2 P(X^T) \geq L'(X^T) + c - 2\} &(1.75)\\
&\leq \Pr\{P(X^T) \leq 2^{-L'(X^T)-c+2}\} &(1.76)\\
&= \sum_{x^T : P(x^T) \leq 2^{-L'(x^T)-c+2}} P(x^T) &(1.77)\\
&\leq \sum_{x^T : P(x^T) \leq 2^{-L'(x^T)-c+2}} 2^{-L'(x^T)-c+2} &(1.78)\\
&\leq 2^{-c+2} \sum_{x^T} 2^{-L'(x^T)} &(1.79)\\
&\leq 2^{-c+2}, &(1.80)
\end{aligned}$$

where in the last step we used Kraft's inequality. ∎

Because the code sequence length $L(x^T)$ can become much larger than c by choosing T large enough, we may conclude that the Elias code cannot be much worse than any other code most of the time.

An even stronger result can be obtained in the special case where the probability distribution is ordered in non-increasing order and is dyadic; see the definition in (1.69). First we shall define a property that we are interested in.

Definition 1.11. *A code with code sequence lengths $L(x^T)$ is said to be competitively optimal with respect to a given source (probability distribution) if for all other uniquely decodable codes with lengths $L'(x^T)$ the following holds:*

$$\Pr\{L(X^T) < L'(X^T)\} \geq \Pr\{L(X^T) > L'(X^T)\}. \tag{1.81}$$

Note that the optimality with respect to *expected code sequence length* does not automatically imply competitive optimality.

In the case of the Elias code for dyadic and non-increasingly ordered probability distributions we know that $L(x^T) = -\log_2 P(x^T)$. The result we now obtain will be stated in the following theorem.

Theorem 1.12 (Cover and Thomas). *For a dyadic and non-increasingly ordered probability distribution $P(x^T)$ the Elias code has code sequence lengths $L(x^T) = -\log_2 P(x^T)$. Let $L'(x^T)$*

be the lengths of any other uniquely decodable code for this source. Then

$$\Pr\{L(X^T) < L'(X^T)\} \geq \Pr\{L(X^T) > L'(X^T)\}, \tag{1.82}$$

with equality if and only if $L'(x^T) = L(x^T)$ for all x^T.

Thus the code whose lengths satisfy the condition $L(x^T) = -\log_2 P(x^T)$ is the *unique* competitively optimal code.

Proof. Consider the function sgn(x) defined as

$$\operatorname{sgn}(x) = \begin{cases} 1; & \text{if } x > 0 \\ 0; & \text{if } x = 0 \\ -1; & \text{if } x < 0\cdot \end{cases} \tag{1.83}$$

It is easy to check that

$$\operatorname{sgn}(x) \leq 2^x - 1, \quad \text{for all } x \text{ with } x \leq 0 \text{ or } x \geq 1. \tag{1.84}$$

Equality holds only when $x = 0$ or $x = 1$.

Consider the difference

$$\begin{aligned} \Pr\{L(X^T) > L'(X^T)\} - \Pr\{L(X^T) < L'(X^T)\} &= \sum_{x^T : L'(x^T) < L(x^T)} P(x^T) - \sum_{x^T : L'(x^T) > L(x^T)} P(x^T) \\ &= \sum_{x^T} P(x^T) \operatorname{sgn}(L(x^T) - L'(x^T)) \\ &\leq \sum_{x^T} 2^{-L(x^T)} \left(2^{L(x^T) - L'(x^T)} - 1\right) \\ &= \sum_{x^T} 2^{-L'(x^T)} - \sum_{x^T} 2^{-L(x^T)} \\ &\leq 0. \end{aligned} \tag{1.85}$$

In this derivation we used (1.84) and, in the last step, we used the Kraft inequality twice, which holds with equality for the lengths $L(x^T)$.

Equality in the theorem holds if the two inequalities in the derivation above hold with equality. For the first one this requires that for all x^T we have either $L(x^T) - L'(x^T) = 0$ or $L(x^T) - L'(x^T) = 1$. The second requirement implies that the Kraft inequality must hold with equality for $L'(x^T)$ also and together this implies that $L'(x^T) = L(x^T)$ for all x^T. ∎

The results in this section are a slightly different variation of the results presented by Cover and Thomas [2, Section 5.11].

1.5 FIXED-LENGTH CODES FOR MEMORYLESS SOURCES, THE AEP

In this section we will pursue a different approach to source coding. It is based on the AEP. We want the code sequences to have a fixed length that is as short as possible. If the length of the code sequences is smaller than the source sequence length, there do not exist code sequences for all source sequences and errors can occur. We show here that the error probability can be made

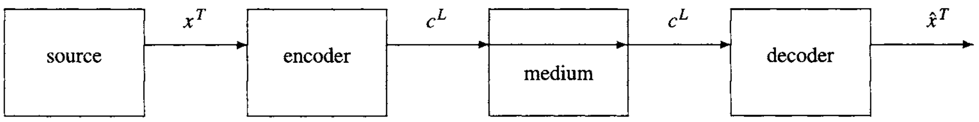

FIGURE 1.9
A source coding system.

arbitrarily small only if the number of code symbols per source symbol is larger than the source entropy.

1.5.1 The Fixed-Length Source Coding Problem

Consider the source coding situation in Fig. 1.9. There a source generates a sequence $x^T = x_1x_2\cdots x_T$ consisting of T source symbols. For reasons of simplicity we assume in this chapter that the source is binary; hence $\mathcal{X} = \{0, 1\}$. The encoder observes the source sequence x^T and produces the fixed-length code sequence $c^L = c_1\cdots c_L$ that corresponds to sequence x^T, so $c^L = f_e(x^T)$. The sequence c^L consists of L binary digits and is conveyed to the decoder. The decoder produces the sequence $\hat{x}^T$ that is represented by the codeword c^L; hence $\hat{x}^T = f_d(c^L)$.

In general the decoder will not always produce the correct estimate $\hat{x}^T$ of the actual source sequence x^T. Hence $\hat{x}^T = x^T$ will not always hold. Therefore the *error probability* P_ϵ, which is defined by

$$P_\epsilon \triangleq \Pr\{\hat{X}^T \neq X^T\}, \tag{1.86}$$

will not be zero. It is our problem in this chapter is to find an encoder and a decoder that achieve a desirable balance between minimizing the code sequence length L and minimizing the error probability P_ϵ. The solution to this problem depends strongly on the *asymptotic equipartition property.* We will develop this concept in the next sections.

1.5.2 Some Probabilities

Consider a binary memoryless source with parameter $\theta = \Pr\{X_t = 1\}$ for $t = 1, \cdots T$. The probability that this source generates a *specific* sequence $x_1x_2\cdots x_T$ that contains e ones (and hence $T - e$ zeros) is $\theta^e(1-\theta)^{T-e}$. There are $\binom{T}{e} = T!/((T-e)!e!)$ sequences containing e ones and $T - e$ zeros. Therefore the probability that the source produces an arbitrary sequence with e ones (and hence $T - e$ zeros) is equal to

$$\Pr\{E = e\} = \binom{T}{e}\theta^e(1-\theta)^{T-e}. \tag{1.87}$$

Here E is the random variable whose value is the number of ones in the sequence $x_1x_2\cdots x_T$.

Example 1.10. For parameter $\theta = 1/5$ and for source sequence length $T = 5$ the probability that a specific sequence with 2 ones, e.g., 10001, occurs is $(1/5)^2(4/5)^3 = 64/3125$. The number of sequences of length 5 with 2 ones is $\binom{5}{2} = \frac{5!}{2!3!} = 10$; thus the probability that an arbitrary sequence of length 5 containing 2 ones occurs is

$$\Pr\{E = 2\} = \binom{5}{2}(1/5)^2(4/5)^3 = 128/625. \tag{1.88}$$

1.5.3 An Example Demonstrating the Asymptotic Equipartition Property

Example 1.11. Again assume that the source parameter $\theta = 0.2$. For $T = 10$, 100, and 1000 we have computed the probabilities $\Pr\{E = e\}$ for $e = 0, 1, \ldots, T$. The results for $T = 10$, $T = 100$, and $T = 1000$ are shown in Fig. 1.10. Observe that the distribution of E concentrates around $0.2T$ when the sequence length T increases. This is a consequence of the law of large numbers.

Next we compute for sequence lengths $T = 10, 100, 1000$, (a) the probability $\Pr\{E \leq 0.25T\}$ that a source with parameter $\theta = 0.2$ produces a sequence with not more than $0.25T$ ones and (b) the number of sequences $\sum_{e \leq 0.25T} \binom{T}{e}$ containing not more than $0.25T$ ones. These probabilities and numbers can be found in Table 1.2. Observe that (a) the probability $\Pr\{E \leq 0.25T\}$ approaches 1 for $T \to \infty$ and (b) the number $\sum_{e \leq 0.25T} \binom{T}{e}$ is always considerably less than 2^T, the total number of sequences of length T. In the next sections we will make all this more precise.

1.5.4 The Idea behind Fixed-Length Source Coding

The idea behind source coding is to reserve binary code sequences of length L only for sequences x^T in a properly chosen subset $\mathcal{A}$ of $\{0, 1\}^T$ (see Fig. 1.11). Hence there are only $|\mathcal{A}|$ code sequences. The set $\mathcal{A}$, if it is well chosen, is often called the *set of typical sequences* or *typical set*. For the length L of the code sequences we can now write

$$L = \lceil \log_2 |\mathcal{A}| \rceil. \tag{1.89}$$

If the source produces a sequence $x^T \notin \mathcal{A}$ one of the code sequences for source sequences in $\mathcal{A}$ is used but this will certainly lead to an error; hence

$$P_\epsilon = \Pr\{X^T \notin \mathcal{A}\}. \tag{1.90}$$

Example 1.12. Let $T = 10$ and assume that the set $\mathcal{A}$ contains all sequences with 0, 1, or 2 ones; hence $|\mathcal{A}| = 1 + \binom{10}{1} + \binom{10}{2} = 56$. We need code sequence length $L = \lceil \log_2 56 \rceil = 6$ to obtain unique code sequences for all source sequences in $\mathcal{A}$; see Fig. 1.12. Note that an error occurs only if $x^T \notin \mathcal{A}$. In that case the code sequence for source sequence 0000000000 is used.

1.5.5 Rate and Error Probability

The performance of a fixed-length source coding system is determined by the *rate* R and the *error probability* P_ϵ. The rate is defined as

$$R \stackrel{\Delta}{=} \frac{L}{T} = \frac{\lceil \log_2 |\mathcal{A}| \rceil}{T}. \tag{1.91}$$

It specifies how many binary code digits per source symbol are necessary. For the error probability we can write

$$P_\epsilon = \Pr\{\hat{X}^T \neq X^T\} = \Pr\{X^T \notin \mathcal{A}\}. \tag{1.92}$$

We must now choose the set $\mathcal{A}$ such that R is small without making P_ϵ large.

Example 1.13. For $T = 10$ and set $\mathcal{A}$ that contains all sequences with 0, 1, or 2 ones, the rate and error probability are $R = \frac{\lceil \log_2 56 \rceil}{10} = \frac{\lceil 5.8074 \rceil}{10} = 0.6$ and $P_\epsilon = \Pr\{X^T \notin \mathcal{A}\} = 0.3222$, respectively.

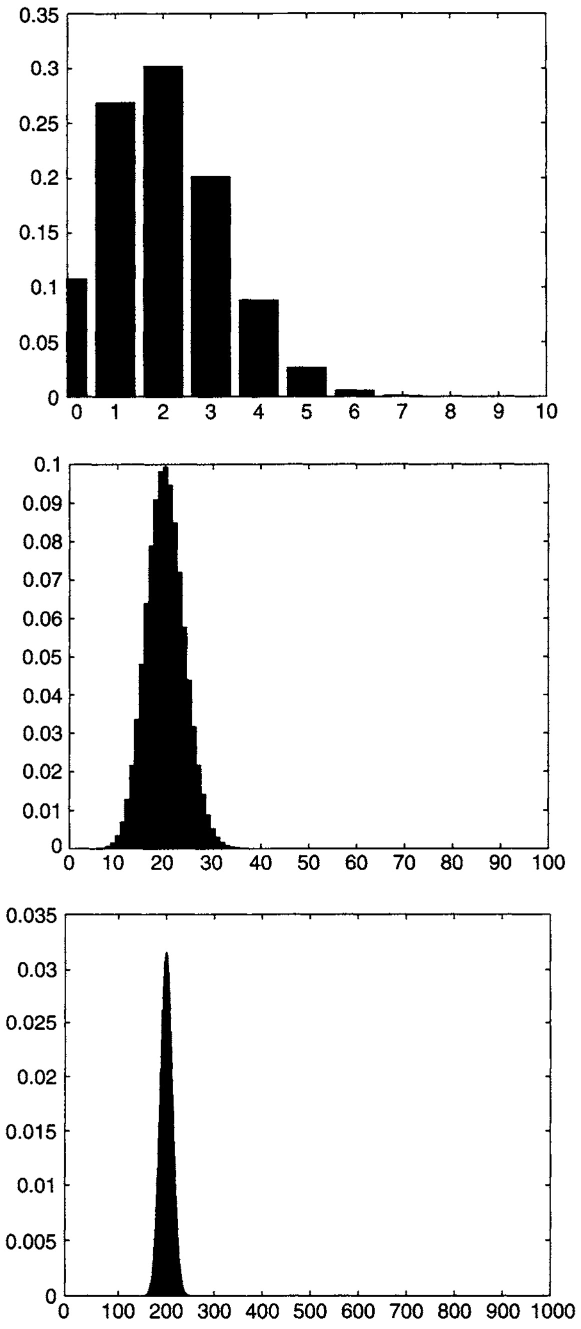

FIGURE 1.10
Probabilities $\Pr\{E = e\}$ for $e = 0, 1, \ldots, T$ and $t = 10$, 100, and 1000.

Table 1.2

T	$\Pr\{E \le 0.25T\}$	$\sum_{e \le 0.25T} \binom{T}{e}$
10	0.6778	$56 = 2^{5.8074}$
100	0.9125	$2^{78.241}$
1000	0.9999	$2^{806.76}$

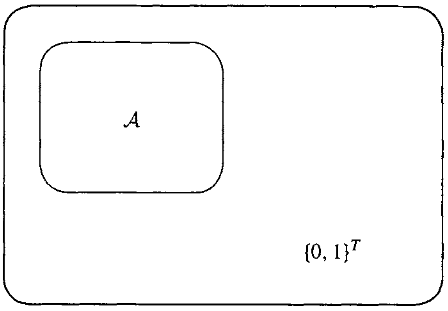

FIGURE 1.11
The typical set $\mathcal{A}$, a subset of $\{0, 1\}^T$.

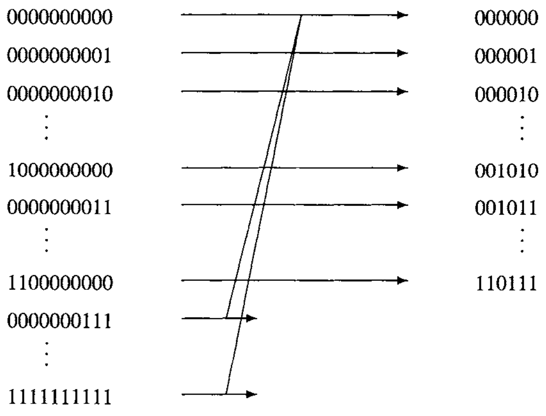

FIGURE 1.12
Source sequences of length $T = 10$ and the corresponding code sequences.

1.5.6 A Hamming Ball

Suppose that we can afford M code sequences. This results in a rate $R = \lceil \log_2 M \rceil / T$. We then achieve the smallest possible error probability P_ϵ when we choose for the set $\mathcal{A}$ the M *most probable* source sequences. Assume that $0 \le \theta < \frac{1}{2}$. If we now rank the source sequences according to their probability, the "first" sequence is $000 \cdots 0$. Then "follow" the sequences with a single one, and then the sequences with 2 ones, etc. The "last" sequence $111 \cdots 1$ is the least probable sequence. The typical sets $\mathcal{A}_\delta$ for $0 \le \delta \le 1$ are therefore defined as

$$\mathcal{A}_\delta \triangleq \{x^T : w_H(x^T) \le \delta T\} \tag{1.93}$$

and are therefore optimal; they achieve the smallest possible P_ϵ over all sets with the same size. The Hamming weight $w_H(x^T)$ of a sequence x^T is defined as the number of non-zero symbols in this sequence. We call a set $\mathcal{A}_\delta$ a *Hamming ball.*

1.5.7 An Optimal Balance between R and P_ϵ

Assume that the set $\mathcal{A}$, i.e., the set of source sequences that have their own code sequence, is the Hamming ball $\mathcal{A}_\delta$. We can now find an upper bound on the rate R and on the error probability P_ϵ as a function of δ, for $\delta > \theta$. Therefore we must define two functions first. Note that the binary entropy function was used in Example 1.2 and given in Fig. 1.1. It is repeated here for convenience.

Definition 1.12. *Suppose that* $0 \le \delta \le 1$. *First we define the binary entropy function as*

$$h(\delta) \triangleq \delta \log_2 \frac{1}{\delta} + (1-\delta)\log_2 \frac{1}{1-\delta}. \tag{1.94}$$

If moreover we assume that $0 < \theta < 1$, *we can define the binary divergence function as*

$$d(\delta||\theta) \triangleq \delta \log_2 \frac{\delta}{\theta} + (1-\delta)\log_2 \frac{1-\delta}{1-\theta}. \tag{1.95}$$

Example 1.14. Plots of $h(\delta)$ and $d(\delta||\theta)$ are shown in Fig. 1.13. Observe that the entropy function $h(\delta)$ is a non-negative increasing convex-∩ function of δ for $0 \le \delta \le \frac{1}{2}$. The divergence function $d(\delta||\theta)$ is related to the tangent to the entropy function $h(\delta)$ in $(\theta, h(\theta))$. This tangent is described by $r_\theta(\delta) = \delta \log_2 \frac{1}{\theta} + (1-\delta)\log_2 \frac{1}{1-\theta}$. Now the divergence function $d(\delta||\theta) = r_\theta(\delta) - h(\delta)$. In Fig. 1.13 the value $\theta = 0.2$.

We now can formulate the following result.

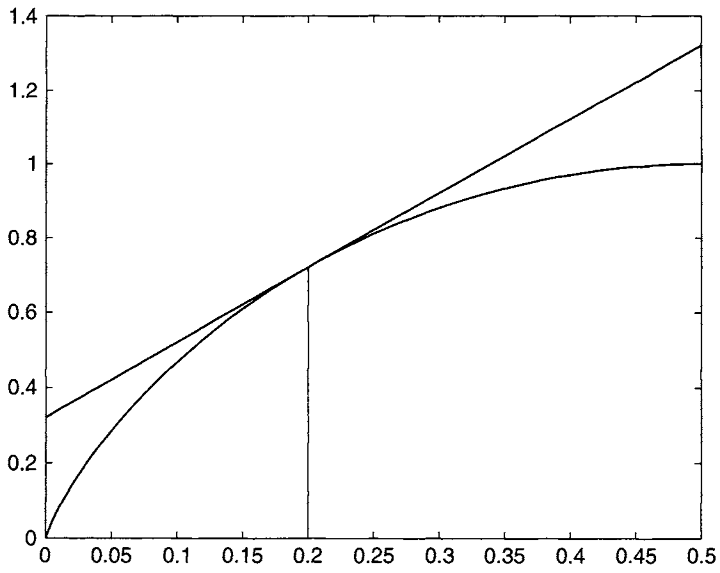

FIGURE 1.13
The binary entropy function $h(\delta)$ and its tangent $r_\theta(\delta)$ in $(\theta, h(\theta))$.

Theorem 1.13. *For a binary memoryless source with parameter θ consider fixed-length source codes that assign code sequences only to the source sequences in a Hamming ball $\mathcal{A}_\delta$. These codes achieve, for $0 < \theta < \delta \le \frac{1}{2}$ and source sequence length T, rates and error probabilities that satisfy*

$$R = \frac{\lceil \log_2 |\mathcal{A}_\delta| \rceil}{T} \le h(\delta) + \frac{1}{T}, \tag{1.96}$$

$$P_\epsilon = \Pr\{X^T \notin \mathcal{A}_\delta\} \le 2^{-Td(\delta||\theta)}. \tag{1.97}$$

Note that for a source with $\theta = 0$ we can easily achieve $R = 0$ and at the same time $P_\epsilon = 0$. Achieving $R = 1$ together with $P_\epsilon = 0$ is also straightforward.

Before we prove this theorem we give an example.

Example 1.15. We can compare the bounds (1.96) and (1.97) to the actual rates R and error probabilities P_ϵ. With $\theta = 0.2$ and $\delta = 0.25$ we first determine

$$h(\delta) = h(0.25) = 0.8113$$

$$d(\delta||\theta) = d(0.25||0.2) = 0.0106.$$

This gives us, for source sequence lengths $T = 10$, 100 and 1000, the following upper bounds on the rates and error probabilities:

T	R	$h(\delta) + \frac{1}{T}$	P_ϵ	$2^{-Td(\delta\|\|\theta)}$
10	0.6	0.9113	0.3222	0.9288
100	0.79	0.8213	0.0875	0.4780
1000	0.807	0.8123	5.07×10^{-5}	6.22×10^{-4}

The error probabilities and rates in this table follow from Table 1.2.

Proof. (A) We will first upper bound the volume of a Hamming ball. Let $0 \le \delta \le \frac{1}{2}$; then [7]

$$|\mathcal{A}_\delta| = \sum_{0 \le e \le \delta T} \binom{T}{e} \le 2^{Th(\delta)}. \tag{1.98}$$

This is a consequence of

$$\begin{aligned}
1 &= (\delta + (1-\delta))^T \\
&\overset{(a)}{\ge} \sum_{0 \le e \le \delta T} \binom{T}{e} \delta^e (1-\delta)^{T-e} = \sum_{0 \le e \le \delta T} \binom{T}{e} (1-\delta)^T \left(\frac{\delta}{1-\delta}\right)^e \\
&\overset{(b)}{\ge} \sum_{0 \le e \le \delta T} \binom{T}{e} (1-\delta)^T \left(\frac{\delta}{1-\delta}\right)^{\delta T} = \delta^{\delta T}(1-\delta)^{(1-\delta)T} \sum_{0 \le e \le \delta T} \binom{T}{e}.
\end{aligned} \tag{1.99}$$

Here (a) follows from Newton's binomium, leaving out some terms, and (b) follows from the fact that $\delta/(1-\delta) \leq 1$ and $e \leq \delta T$. From (1.98) we now easily obtain (1.96)

$$R = \frac{\lceil \log_2 |\mathcal{A}_\delta| \rceil}{T} \leq \frac{\log_2 |\mathcal{A}_\delta| + 1}{T} = h(\delta) + \frac{1}{T}. \tag{1.100}$$

(B) Next we derive an upper bound for the probability of occurrence of a sequence outside a Hamming ball. Let $0 < \theta < \delta \leq 1$; then

$$\sum_{\delta T \leq e \leq T} \binom{T}{e} \theta^e (1-\theta)^{T-e} \leq 2^{-Td(\delta||\theta)}. \tag{1.101}$$

This can be shown by first considering the case where $\delta < 1$. Then

$$\begin{aligned}
1 = (\delta + (1-\delta))^T &\overset{(a)}{\geq} \sum_{\delta T \leq e \leq T} \binom{T}{e} \delta^e (1-\delta)^{T-e} \\
&\overset{(b)}{=} \sum_{\delta T \leq e \leq T} \binom{T}{e} \left(\frac{1-\delta}{1-\theta}\right)^T \left(\frac{\delta(1-\theta)}{\theta(1-\delta)}\right)^e \theta^e (1-\theta)^{T-e} \\
&\overset{(c)}{\geq} \sum_{\delta T \leq e \leq T} \binom{T}{e} \left(\frac{1-\delta}{1-\theta}\right)^T \left(\frac{\delta(1-\theta)}{\theta(1-\delta)}\right)^{\delta T} \theta^e (1-\theta)^{T-e} \\
&= \left(\frac{\delta}{\theta}\right)^{\delta T} \left(\frac{1-\delta}{1-\theta}\right)^{(1-\delta)T} \sum_{\delta T \leq e \leq T} \binom{T}{e} \theta^e (1-\theta)^{T-e}.
\end{aligned} \tag{1.102}$$

Here (a) follows again from Newton's binomium, (b) is just rewriting, and (c) follows from $\delta(1-\theta) > \theta(1-\delta)$ and $e \geq \delta T$. Finally we consider the case where $\delta = 1$. Then $d(\delta||\theta) = -\log_2(\theta)$. From (1.101) we easily obtain (1.97) since

$$P_\epsilon = \Pr\{X^T \notin \mathcal{A}_\delta\} = \sum_{\delta T \leq e \leq T} \binom{T}{e} \theta^e (1-\theta)^{T-e} \leq 2^{-Td(\delta||\theta)}. \tag{1.103}$$

This finishes the proof of Theorem 1.13. ∎

1.5.8 The Fixed-Length Coding Theorem

Theorem 1.14. *For a binary memoryless source with parameter $\frac{1}{2} > \theta > 0$ there exist source codes with rates $1 \geq R > h(\theta)$ that achieve*

$$\lim_{T \to \infty} P_\epsilon = 0. \tag{1.104}$$

Moreover for $\theta = 0$ or $\theta = \frac{1}{2}$ there exist codes with rates $R = h(\theta)$ and error probability $P_\epsilon = 0$. Therefore we say that the entropy $h(\theta)$ is achievable.

Proof. The proof of this theorem is simple. Just choose $\delta \leq \frac{1}{2}$ such that $h(\delta) = R$. By the monotonocity of the binary entropy function $\frac{1}{2} \geq \delta > \theta > 0$. Therefore $d(\delta||\theta) > 0$. The result is trivial for $\theta = 0$ or $\theta = \frac{1}{2}$. ∎

So far we have looked only at binary memoryless sources with $0 \leq \theta \leq \frac{1}{2}$. Note, however, that for $\theta > \frac{1}{2}$ we can interchange zeros and ones in the source sequences. Then $h(1-\theta)$ is seen to be achievable. However, by the symmetry of the binary entropy function around $\theta = \frac{1}{2}$ we know that $h(1-\theta) = h(\theta)$. Therefore for binary memoryless sources with parameter $0 \leq \theta \leq 1$ the entropy $h(\theta)$ is achievable.

1.5.9 Converse and Conclusion

We have shown that entropy can be achieved. However, it is possible for even better codes to exist. Since we have chosen code sequences for only the most probable source sequences our coding methods are optimal. Nevertheless this does not yet imply that we cannot achieve compression rates below entropy with vanishing error probabilities.

So, consider our optimal coding scheme again, but now with $\delta < \theta$. So, with (1.100) we may conclude that for T large enough the code rate will be below the source entropy $h(\theta)$. However, the error probability of this scheme can be bounded in a way similar to (1.103), only now we consider $P_c = 1 - P_\epsilon$. Again we can show that

$$P_c = \Pr\{X^T \in \mathcal{A}_\delta\} = \sum_{0 \leq e \leq \delta T} \binom{T}{e} \theta^e (1-\theta)^{T-e} \leq 2^{-Td(\delta\|\theta)}. \tag{1.105}$$

So, the probability for correct decoding is bounded from below and arbitrarily small error probabilities cannot be achieved.

We have looked here only at the binary case. But the results of the previous sections can be generalized to memoryless sources with non-binary alphabets as well.

In this chapter we have discussed aspects of information theory relevant to lossless compression. There are a number of books available that deal with the subject of information. Two highly popular books are those by Cover and Thomas [2] and Gallager [3].

1.6 REFERENCES

1. Abramson, N., 1963. *Information Theory and Coding,* pp. 61–62. McGraw-Hill, New York.
2. Cover, T. M., and J. A. Thomas, 1991. *Elements of Information Theory.* Wiley, New York.
3. Gallager, R. G., 1968. *Information Theory and Reliable Communication.* Wiley, New York.
4. Gray, R. M., and L. D. Davisson, 1986. *Random Processes: A Mathematical Approach for Engineers.* Prentice-Hall, Englewood Cliffs, NJ.
5. Hartley, R. V. L., 1928. The transmission of information. *Bell Systems Technical Journal,* Vol. 17, pp. 535–564, July 1928.
6. Jelinek, F., 1968. *Probabilistic Information Theory,* pp. 476–489, McGraw-Hill, New York.
7. van Lint, J. H., 1992. *Introduction to Coding Theory.* Springer-Verlag, Berlin/New York.
8. Pasco, R., 1976. *Source Coding Algorithms for Fast Data Compression,* Ph.D. thesis. Stanford University, Palo Alto, CA.
9. Rissanen, J., 1976. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development,* Vol. 20, p. 198.
10. Shannon, C. E., 1948. A mathematical theory of communication. *Bell Systems Technical Journal,* Vol. 27, pp. 379–423, July 1948. Reprinted in *Key Papers in the Development of Information Theory* (D. Slepian, Ed.), pp. 5–18, IEEE Press, New York, 1974.
11. Shields, P. C., 1987. The ergodic and entropy theorems revisited. *IEEE Transactions on Information Theory,* Vol. IT-33, pp. 263–266, March 1987.