Removing the conditioning on the length of the sequence is straightforward. By similar arguments, we can show that

$$H(X) \le \frac{1}{n} \sum_{x^n} f(x^n) K(x^n) \le H(X) + \frac{(|\mathcal{X}|+1)\log n}{n} + \frac{c}{n} \tag{14.36}$$

for all n. The lower bound follows from the fact that $K(x^n)$ is also a prefix-free code for the source, and the upper bound can be derived from the fact that $K(x^n) \le K(x^n|n) + 2\log n + c$. Thus,

$$E\frac{1}{n}K(X^n) \to H(X), \tag{14.37}$$

and the compressibility achieved by the computer goes to the entropy limit.

14.4 KOLMOGOROV COMPLEXITY OF INTEGERS

In Section 14.3 we defined the Kolmogorov complexity of a binary string as the length of the shortest program for a universal computer that prints out that string. We can extend that definition to define the Kolmogorov complexity of an integer to be the Kolmogorov complexity of the corresponding binary string.

Definition The *Kolmogorov complexity of an integer n* is defined as

$$K(n) = \min_{p:\mathcal{U}(p)=n} l(p). \tag{14.38}$$

The properties of the Kolmogorov complexity of integers are very similar to those of the Kolmogorov complexity of bit strings. The following properties are immediate consequences of the corresponding properties for strings.

Theorem 14.4.1 *For universal computers $\mathcal{A}$ and $\mathcal{U}$,*

$$K_{\mathcal{U}}(n) \le K_{\mathcal{A}}(n) + c_{\mathcal{A}}. \tag{14.39}$$

Also, since any number can be specified by its binary expansion, we have the following theorem.

Theorem 14.4.2

$$K(n) \le \log^* n + c. \tag{14.40}$$

Theorem 14.4.3 *There are an infinite number of integers n such that $K(n) > \log n$.*

Proof: We know from Lemma 14.3.1 that

$$\sum_n 2^{-K(n)} \leq 1 \tag{14.41}$$

and

$$\sum_n 2^{-\log n} = \sum_n \frac{1}{n} = \infty. \tag{14.42}$$

But if $K(n) < \log n$ for all $n > n_0$, then

$$\sum_{n=n_0}^{\infty} 2^{-K(n)} > \sum_{n=n_0}^{\infty} 2^{-\log n} = \infty, \tag{14.43}$$

which is a contradiction. □

14.5 ALGORITHMICALLY RANDOM AND INCOMPRESSIBLE SEQUENCES

From the examples in Section 14.2, it is clear that there are some long sequences that are simple to describe, like the first million bits of π. By the same token, there are also large integers that are simple to describe, such as

$$2^{2^{2^{2^{2^{2^{2}}}}}}$$

or (100!)!.

We now show that although there are some simple sequences, most sequences do not have simple descriptions. Similarly, most integers are not simple. Hence, if we draw a sequence at random, we are likely to draw a complex sequence. The next theorem shows that the probability that a sequence can be compressed by more than k bits is no greater than 2^{-k}.

Theorem 14.5.1 *Let $X_1, X_2, \ldots, X_n$ be drawn according to a Bernoulli $(\frac{1}{2})$ process. Then*

$$P\left(K(X_1X_2\ldots X_n|n) < n-k\right) < 2^{-k}. \tag{14.44}$$

Proof:

$$\begin{aligned} &P(K(X_1X_2\ldots X_n|n) < n-k) \\ &\quad = \sum_{x_1x_2\ldots x_n:\, K(x_1x_2\ldots x_n|n)<n-k} p(x_1, x_2, \ldots, x_n) && (14.45) \\ &\quad = \sum_{x_1x_2\ldots x_n:\, K(x_1x_2\ldots x_n|n)<n-k} 2^{-n} && (14.46) \\ &\quad = |\{x_1x_2\ldots x_n :\ K(x_1x_2\ldots x_n|n) < n-k\}|\, 2^{-n} \\ &\quad < 2^{n-k}2^{-n} \quad \text{(by Theorem 14.2.4)} && (14.47) \\ &\quad = 2^{-k}. && (14.48) \end{aligned}$$

□

Thus, most sequences have a complexity close to their length. For example, the fraction of sequences of length n that have complexity less than $n - 5$ is less than 1/32. This motivates the following definition:

Definition A sequence $x_1, x_2, \ldots, x_n$ is said to be *algorithmically random* if

$$K(x_1x_2\ldots x_n|n) \ge n. \tag{14.49}$$

Note that by the counting argument, there exists, for each n, at least one sequence x^n such that

$$K(x^n|n) \ge n. \tag{14.50}$$

Definition We call an infinite string x *incompressible* if

$$\lim_{n\to\infty} \frac{K(x_1x_2x_3\cdots x_n|n)}{n} = 1. \tag{14.51}$$

Theorem 14.5.2 (*Strong law of large numbers for incompressible sequences*) *If a string $x_1x_2\ldots$ is incompressible, it satisfies the law of large numbers in the sense that*

$$\frac{1}{n}\sum_{i=1}^{n} x_i \to \frac{1}{2}. \tag{14.52}$$

Hence the proportions of 0's and 1's in any incompressible string are almost equal.

Proof: Let $\theta_n = \frac{1}{n}\sum_{i=1}^n x_i$ denote the proportion of 1's in $x_1x_2\ldots x_n$. Then using the method of Example 14.2, one can write a program of length $nH_0(\theta_n) + 2\log(n\theta_n) + c$ to print x^n. Thus,

$$\frac{K(x^n|n)}{n} < H_0(\theta_n) + 2\frac{\log n}{n} + \frac{c'}{n}. \tag{14.53}$$

By the incompressibility assumption, we also have the lower bound for large enough n,

$$1 - \epsilon \le \frac{K(x^n|n)}{n} \le H_0(\theta_n) + 2\frac{\log n}{n} + \frac{c'}{n}. \tag{14.54}$$

Thus,

$$H_0(\theta_n) > 1 - \frac{2\log n + c'}{n} - \epsilon. \tag{14.55}$$

Inspection of the graph of $H_0(p)$ (Figure 14.2) shows that θ_n is close to $\frac{1}{2}$ for large n. Specifically, the inequality above implies that

$$\theta_n \in \left(\frac{1}{2} - \delta_n, \frac{1}{2} + \delta_n\right), \tag{14.56}$$

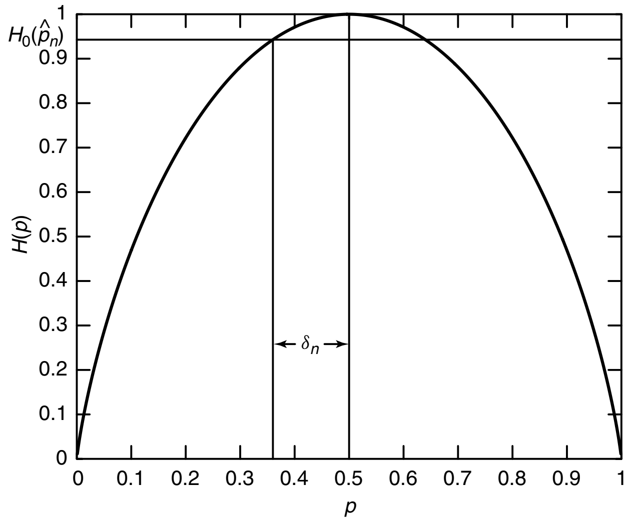


FIGURE 14.2. $H_0(p)$ vs. p.

where δ_n is chosen so that

$$H_0\left(\frac{1}{2}-\delta_n\right) = 1 - \frac{2\log n + c_n + c'}{n}, \tag{14.57}$$

which implies that $\delta_n \to 0$ as $n \to \infty$. Thus, $\frac{1}{n}\sum x_i \to \frac{1}{2}$ as $n \to \infty$. □

We have now proved that incompressible sequences look random in the sense that the proportion of 0's and 1's are almost equal. In general, we can show that if a sequence is incompressible, it will satisfy all computable statistical tests for randomness. (Otherwise, identification of the test that x fails will reduce the descriptive complexity of x, yielding a contradiction.) In this sense, the algorithmic test for randomness is the ultimate test, including within it all other computable tests for randomness.

We now prove a related law of large numbers for the Kolmogorov complexity of Bernoulli(θ) sequences. The Kolmogorov complexity of a sequence of binary random variables drawn i.i.d. according to a Bernoulli(θ) process is close to the entropy $H_0(\theta)$. In Theorem 14.3.1 we proved that the expected value of the Kolmogorov complexity of a random Bernoulli sequence converges to the entropy [i.e., $E\frac{1}{n}K(X_1X_2\ldots X_n|n) \to H_0(\theta)$]. Now we remove the expectation.

Theorem 14.5.3 *Let $X_1, X_2, \ldots, X_n$ be drawn i.i.d. $\sim$ Bernoulli(θ). Then*

$$\frac{1}{n}K(X_1X_2\ldots X_n|n) \to H_0(\theta) \qquad \textit{in probability}. \tag{14.58}$$

Proof: Let $\overline{X}_n = \frac{1}{n}\sum X_i$ be the proportion of 1's in $X_1, X_2, \ldots, X_n$. Then using the method described in (14.23), we have

$$K(X_1X_2\ldots X_n|n) \le nH_0\left(\overline{X}_n\right) + 2\log n + c, \tag{14.59}$$

and since by the weak law of large numbers, $\overline{X}_n \to \theta$ in probability, we have

$$\Pr\left\{\frac{1}{n}K(X_1X_2\ldots X_n|n) - H_0(\theta) \ge \epsilon\right\} \to 0. \tag{14.60}$$

Conversely, we can bound the number of sequences with complexity significantly lower than the entropy. From the AEP, we can divide the set of sequences into the typical set and the nontypical set. There are at least $(1-\epsilon)2^{n(H_0(\theta)-\epsilon)}$ sequences in the typical set. At most $2^{n(H_0(\theta)-c)}$ of these typical sequences can have a complexity less than $n(H_0(\theta)-c)$. The probability that the complexity of the random sequence is less than $n(H_0(\theta)-c)$ is

$$\begin{aligned}
&\Pr(K(X^n|n) < n(H_0(\theta)-c)) \\
&\quad \le \Pr(X^n \notin A_\epsilon^{(n)}) + \Pr(X^n \in A_\epsilon^{(n)}, K(X^n|n) < n(H_0(\theta)-c)) \\
&\quad \le \epsilon + \sum_{x^n \in A_\epsilon^{(n)}, K(x^n|n) < n(H_0(\theta)-c)} p(x^n) && (14.61) \\
&\quad \le \epsilon + \sum_{x^n \in A_\epsilon^{(n)}, K(x^n|n) < n(H_0(\theta)-c)} 2^{-n(H_0(\theta)-\epsilon)} && (14.62) \\
&\quad \le \epsilon + 2^{n(H_0(\theta)-c)} 2^{-n(H_0(\theta)-\epsilon)} && (14.63) \\
&\quad = \epsilon + 2^{-n(c-\epsilon)}, && (14.64)
\end{aligned}$$

which is arbitrarily small for appropriate choice of ϵ, n, and c. Hence with high probability, the Kolmogorov complexity of the random sequence is close to the entropy, and we have

$$\frac{K(X_1, X_2, \ldots, X_n|n)}{n} \to H_0(\theta) \qquad \text{in probability.} \tag{14.65}$$

□

14.6 UNIVERSAL PROBABILITY

Suppose that a computer is fed a random program. Imagine a monkey sitting at a keyboard and typing the keys at random. Equivalently, feed a series of fair coin flips into a universal Turing machine. In either case, most strings will not make sense to the computer. If a person sits at a terminal and types keys at random, he will probably get an error message (i.e., the computer will print the null string and halts). But with a certain probability she will hit on something that makes sense. The computer will then print out something meaningful. Will this output sequence look random?

From our earlier discussions, it is clear that most sequences of length n have complexity close to n. Since the probability of an input program p is $2^{-l(p)}$, shorter programs are much more probable than longer ones; and when they produce long strings, shorter programs do not produce random strings; they produce strings with simply described structure.

The probability distribution on the output strings is far from uniform. Under the computer-induced distribution, simple strings are more likely than complicated strings of the same length. This motivates us to define a universal probability distribution on strings as follows:

Definition The *universal probability* of a string x is

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)} = \Pr(\mathcal{U}(p) = x), \tag{14.66}$$

which is the probability that a program randomly drawn as a sequence of fair coin flips $p_1, p_2, \ldots$ will print out the string x.

This probability is universal in many senses. We can consider it as the probability of observing such a string in nature; the implicit belief is that simpler strings are more likely than complicated strings. For example, if we wish to describe the laws of physics, we might consider the simplest string describing the laws as the most likely. This principle, known as Occam's Razor, has been a general principle guiding scientific research for centuries: If there are many explanations consistent with the observed data, choose the simplest. In our framework, Occam's Razor is equivalent to choosing the shortest program that produces a given string.

This probability mass function is called *universal* because of the following theorem.

Theorem 14.6.1 *For every computer $\mathcal{A}$,*

$$P_{\mathcal{U}}(x) \geq c'_{\mathcal{A}} P_{\mathcal{A}}(x) \tag{14.67}$$

for every string $x \in \{0, 1\}^*$, where the constant $c'_{\mathcal{A}}$ depends only on $\mathcal{U}$ and $\mathcal{A}$.

Proof: From the discussion of Section 14.2, we recall that for every program p' for $\mathcal{A}$ that prints x, there exists a program p for $\mathcal{U}$ of length not more than $l(p') + c_{\mathcal{A}}$ produced by prefixing a simulation program for $\mathcal{A}$. Hence,

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)} \geq \sum_{p':\mathcal{A}(p')=x} 2^{-l(p')-c_{\mathcal{A}}} = c'_{\mathcal{A}} P_{\mathcal{A}}(x). \tag{14.68}$$

$\square$

Any sequence drawn according to a computable probability mass function on binary strings can be considered to be produced by some computer $\mathcal{A}$ acting on a random input (via the probability inverse transformation acting on a random input). Hence, the universal probability distribution includes a mixture of all computable probability distributions.

Remark (*Bounded likelihood ratio*). In particular, Theorem 14.6.1 guarantees that a likelihood ratio test of the hypothesis that X is drawn according to $P_{\mathcal{U}}$ versus the hypothesis that it is drawn according to $P_{\mathcal{A}}$ will have bounded likelihood ratio. If $\mathcal{U}$ and A are universal, the $P_{\mathcal{U}}(x)/P_{\mathcal{A}}(x)$ is bounded away from 0 and infinity for all x. This is in contrast to other simple hypothesis-testing problems (like Bernoulli(θ_1) versus Bernoulli(θ_2)), where the likelihood ratio goes to 0 or ∞ as the sample size goes to infinity. Apparently, $P_{\mathcal{U}}$, which is a mixture of all computable distributions, can never be rejected completely as the true distribution of any data drawn according to some computable probability distribution. In that sense we cannot reject the possibility that the universe is the output of monkeys typing at a computer. However, we can reject the hypothesis that the universe is random (monkeys with no computer).

In Section 14.11 we prove that

$$P_{\mathcal{U}}(x) \approx 2^{-K(x)}, \tag{14.69}$$

thus showing that $K(x)$ and $\log \frac{1}{P_{\mathcal{U}}(x)}$ have equal status as universal algorithmic complexity measures. This is especially interesting since $\log \frac{1}{P_{\mathcal{U}}(x)}$ is the ideal codeword length (the Shannon codeword length) with respect to the universal probability distribution $P_{\mathcal{U}}(x)$.

We conclude this section with an example of a monkey at a typewriter vs. a monkey at a computer keyboard. If the monkey types at random on a typewriter, the probability that it types out all the works of Shakespeare (assuming that the text is 1 million bits long) is $2^{-1,000,000}$. If the monkey sits at a computer terminal, however, the probability that it types out Shakespeare is now $2^{-K(\text{Shakespeare})} \approx 2^{-250,000}$, which although extremely small is still exponentially more likely than when the monkey sits at a dumb typewriter.

The example indicates that a random input to a computer is much more likely to produce "interesting" outputs than a random input to a typewriter. We all know that a computer is an intelligence amplifier. Apparently, it creates sense from nonsense as well.

14.7 THE HALTING PROBLEM AND THE NONCOMPUTABILITY OF KOLMOGOROV COMPLEXITY

Consider the following paradoxical statement:

This statement is false.

This paradox is sometimes stated in a two-statement form:

The next statement is false.
The preceding statement is true.

These paradoxes are versions of what is called the *Epimenides liar paradox*, and it illustrates the pitfalls involved in self-reference. In 1931, Gödel used this idea of self-reference to show that any interesting system of mathematics is not complete; there are statements in the system that are true but that cannot be proved within the system. To accomplish this, he translated theorems and proofs into integers and constructed a statement of the above form, which can therefore not be proved true or false.

The *halting problem* in computer science is very closely connected with Gödel's incompleteness theorem. In essence, it states that for any computational model, there is no general algorithm to decide whether a program will halt or not (go on forever). Note that it is not a statement about any specific program. Quite clearly, there are many programs that can easily be shown to halt or go on forever. The halting problem says that we cannot answer this question for all programs. The reason for this is again the idea of self-reference.

To a practical person, the halting problem may not be of any immediate significance, but it has great theoretical importance as the dividing line between things that can be done on a computer (given unbounded memory and time) and things that cannot be done at all (such as proving all true statements in number theory). Gödel's incompleteness theorem is one of the most important mathematical results of the twentieth century, and its consequences are still being explored. The halting problem is an essential example of Gödel's incompleteness theorem.

One of the consequences of the nonexistence of an algorithm for the halting problem is the noncomputability of Kolmogorov complexity. The only way to find the shortest program in general is to try all short programs and see which of them can do the job. However, at any time some of the short programs may not have halted and there is no effective (finite mechanical) way to tell whether or not they will halt and what they will print out. Hence, there is no effective way to find the shortest program to print a given string.

The noncomputability of Kolmogorov complexity is an example of the *Berry paradox*. The Berry paradox asks for the shortest number not nameable in under 10 words. A number like 1,101,121 cannot be a solution since the defining expression itself is less than 10 words long. This illustrates the problems with the terms *nameable* and *describable*; they are too powerful to be used without a strict meaning. If we restrict ourselves to the meaning "can be described for printing out on a computer," we can resolve Berry's paradox by saying that the smallest number not describable in less than 10 words exists but is not computable. This "description" is not a program for computing the number. E. F. Beckenbach pointed out a similar problem in the classification of numbers as dull or interesting; the smallest dull number must be interesting.

As stated at the beginning of the chapter, one does not really anticipate that practitioners will find the shortest computer program for a given string. The shortest program is not computable, although as more and more programs are shown to produce the string, the estimates from above of the Kolmogorov complexity converge to the true Kolmogorov complexity. (The problem, of course, is that one may have found the shortest program and never know that no shorter program exists.) Even though Kolmogorov complexity is not computable, it provides a framework within which to consider questions of randomness and inference.

14.8 Ω

In this section we introduce Chaitin's mystical, magical number, Ω, which has some extremely interesting properties.

Definition

$$\Omega = \sum_{p:\mathcal{U}(p)\text{ halts}} 2^{-l(p)}. \tag{14.70}$$

Note that $\Omega = \Pr(\mathcal{U}(p)\text{ halts})$, the probability that the given universal computer halts when the input to the computer is a binary string drawn according to a Bernoulli($\frac{1}{2}$) process.

Since the programs that halt are prefix-free, their lengths satisfy the Kraft inequality, and hence the sum above is always between 0 and 1. Let $\Omega_n = .\omega_1\omega_2\cdots\omega_n$ denote the first n bits of Ω. The properties of Ω are as follows:

1. Ω *is noncomputable*. There is no effective (finite, mechanical) way to check whether arbitrary programs halt (the halting problem), so there is no effective way to compute Ω.
2. Ω *is a "philosopher's stone"*. Knowing Ω to an accuracy of n bits will enable us to decide the truth of any provable or finitely refutable mathematical theorem that can be written in less than n bits. Actually, all that this means is that given n bits of Ω, there is an effective procedure to decide the truth of n-bit theorems; the procedure may take an arbitrarily long (but finite) time. Of course, without knowing Ω, it is not possible to check the truth or falsity of

every theorem by an effective procedure (Gödel's incompleteness theorem).

The basic idea of the procedure using n bits of Ω is simple: We run all programs until the sum of the masses $2^{-l(p)}$ contributed by programs that halt equals or exceeds $\Omega_n = 0.\omega_1\omega_2\cdots\omega_n$, the truncated version of Ω that we are given. Then, since

$$\Omega - \Omega_n < 2^{-n}, \tag{14.71}$$

we know that the sum of all further contributions of the form $2^{-l(p)}$ to Ω from programs that halt must also be less than 2^{-n}. This implies that no program of length $\leq n$ that has not yet halted will ever halt, which enables us to decide the halting or nonhalting of all programs of length $\leq n$.

To complete the proof, we must show that it is possible for a computer to run all possible programs in "parallel" in such a way that any program that halts will eventually be found to halt. First, list all possible programs, starting with the null program, Λ:

$$\Lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots. \tag{14.72}$$

Then let the computer execute one clock cycle of Λ for the first cycle. In the next cycle, let the computer execute two clock cycles of Λ and two clock cycles of the program 0. In the third cycle, let it execute three clock cycles of each of the first three programs, and so on. In this way, the computer will eventually run all possible programs and run them for longer and longer times, so that if any program halts, it will eventually be discovered to halt. The computer keeps track of which program is being executed and the cycle number so that it can produce a list of all the programs that halt. Thus, we will ultimately know whether or not any program of less than n bits will halt. This enables the computer to find any proof of the theorem or a counterexample to the theorem if the theorem can be stated in less than n bits. Knowledge of Ω turns previously unprovable theorems into provable theorems. Here Ω acts as an oracle.

Although Ω seems magical in this respect, there are other numbers that carry the same information. For example, if we take the list of programs and construct a real number in which the ith bit indicates whether program i halts, this number can also be used to decide any finitely refutable question in mathematics. This number is very dilute (in information content) because one needs approximately 2^n

bits of this indicator function to decide whether or not an n-bit program halts. Given 2^n bits, one can tell immediately without any computation whether or not any program of length less than n halts. However, Ω is the most compact representation of this information since it is algorithmically random and incompressible.

What are some of the questions that we can resolve using Ω? Many of the interesting problems in number theory can be stated as a search for a counterexample. For example, it is straightforward to write a program that searches over the integers x, y, z, and n and halts only if it finds a counterexample to Fermat's last theorem, which states that

$$x^n + y^n = z^n \tag{14.73}$$

has no solution in integers for $n \geq 3$. Another example is *Goldbach's conjecture*, which states that any even number is a sum of two primes. Our program would search through all the even numbers starting with 2, check all prime numbers less than it and find a decomposition as a sum of two primes. It will halt if it comes across an even number that does not have such a decomposition. Knowing whether this program halts is equivalent to knowing the truth of Goldbach's conjecture.

We can also design a program that searches through all proofs and halts only when it finds a proof of the theorem required. This program will eventually halt if the theorem has a finite proof. Hence knowing n bits of Ω, we can find the truth or falsity of all theorems that have a finite proof or are finitely refutable and which can be stated in less than n bits.

3. Ω *is algorithmically random*.

Theorem 14.8.1 Ω *cannot be compressed by more than a constant; that is, there exists a constant c such that*

$$K(\omega_1\omega_2\ldots\omega_n) \geq n - c \quad \textit{for all } n. \tag{14.74}$$

Proof: We know that if we are given n bits of Ω, we can determine whether or not any program of length $\leq n$ halts. Using $K(\omega_1\omega_2\cdots\omega_n)$ bits, we can calculate n bits of Ω, and then we can generate a list of all programs of length $\leq n$ that halt, together with their corresponding outputs. We find the first string x_0 that is not on this list. The string x_0 is then the shortest string with Kolmogorov complexity $K(x_0) > n$. The complexity of this program to print x_0 is $K(\Omega_n) + c$, which must be at least as long as the shortest program for x_0. Consequently,

$$K(\Omega_n) + c \geq K(x_0) > n \tag{14.75}$$

for all n. Thus, $K(\omega_1\omega_2 \cdots \omega_n) > n - c$, and Ω cannot be compressed by more than a constant. □

14.9 UNIVERSAL GAMBLING

Suppose that a gambler is asked to gamble sequentially on sequences $x \in \{0, 1\}^*$. He has no idea of the origin of the sequence. He is given fair odds (2-for-1) on each bit. How should he gamble? If he knew the distribution of the elements of the string, he might use proportional betting because of its optimal growth-rate properties, as shown in Chapter 6. If he believes that the string occurred naturally, it seems intuitive that simpler strings are more likely than complex ones. Hence, if he were to extend the idea of proportional betting, he might bet according to the universal probability of the string. For reference, note that if the gambler knows the string x in advance, he can increase his wealth by a factor of $2^{l(x)}$ simply by betting all his wealth each time on the next symbol of x. Let the wealth $S(x)$ associated with betting scheme $b(x)$, $\sum b(x) = 1$, be given by

$$S(x) = 2^{l(x)} b(x). \tag{14.76}$$

Suppose that the gambler bets $b(x) = 2^{-K(x)}$ on a string x. This betting strategy can be called *universal gambling*. We note that the sum of the bets

$$\sum_x b(x) = \sum_x 2^{-K(x)} \leq \sum_{p:p \text{ halts}} 2^{-l(p)} = \Omega \leq 1, \tag{14.77}$$

and he will not have used all his money. For simplicity, let us assume that he throws the rest away. For example, the amount of wealth resulting from a bet $b(0110)$ on a sequence $x = 0110$ is $2^{l(x)} b(x) = 2^4 b(0110)$ plus the amount won on all bets $b(0110\ldots)$ on sequences that extend x.

Then we have the following theorem:

Theorem 14.9.1 *The logarithm of the wealth a gambler achieves on a sequence using universal gambling plus the complexity of the sequence is no smaller than the length of the sequence, or*

$$\log S(x) + K(x) \geq l(x). \tag{14.78}$$

Remark This is the counterpart of the gambling conservation theorem $W^* + H = \log m$ from Chapter 6.

Proof: The proof follows directly from the universal gambling scheme, $b(x) = 2^{-K(x)}$, since

$$S(x) = \sum_{x' \sqsupseteq x} 2^{l(x)} b(x') \geq 2^{l(x)} 2^{-K(x)}, \tag{14.79}$$

where $x' \sqsupseteq x$ means that x is a prefix of x'. Taking logarithms establishes the theorem. □

The result can be understood in many ways. For infinite sequences x with finite Kolmogorov complexity,

$$S(x_1 x_2 \cdots x_l) \geq 2^{l-K(x)} = 2^{l-c} \tag{14.80}$$

for all l. Since 2^l is the most that can be won in l gambles at fair odds, this scheme does asymptotically as well as the scheme based on knowing the sequence in advance. For example, if $x = \pi_1 \pi_2 \cdots \pi_n \cdots$, the digits in the expansion of π, the wealth at time n will be $S_n = S(x^n) \geq 2^{n-c}$ for all n.

If the string is actually generated by a Bernoulli process with parameter p, then

$$S(X_1 \ldots X_n) \geq 2^{n - nH_0(\overline{X}_n) - 2\log n - c} \approx 2^{n\left(1 - H_0(p) - 2\frac{\log n}{n} - \frac{c}{n}\right)}, \tag{14.81}$$

which is the same to first order as the rate achieved when the gambler knows the distribution in advance, as in Chapter 6.

From the examples we see that the universal gambling scheme on a random sequence does asymptotically as well as a scheme that uses prior knowledge of the true distribution.

14.10 OCCAM'S RAZOR

In many areas of scientific research, it is important to choose among various explanations of data observed. After choosing the explanation, we wish to assign a confidence level to the predictions that ensue from the laws that have been deduced. For example, Laplace considered the probability that the sun will rise again tomorrow given that it has risen every day in recorded history. Laplace's solution was to assume that the rising of the sun was a Bernoulli(θ) process with unknown parameter θ. He assumed that θ was uniformly distributed on the unit interval. Using the observed data, he calculated the posterior probability that the sun will rise again tomorrow and found that it was

$$\begin{aligned} &P(X_{n+1}=1|X_n=1, X_{n-1}=1,\ldots,X_1=1) \\ &\quad= \frac{P(X_{n+1}=1, X_n=1, X_{n-1}=1,\ldots,X_1=1)}{P(X_n=1, X_{n-1}=1,\ldots,X_1=1)} \\ &\quad= \frac{\int_0^1 \theta^{n+1}\,d\theta}{\int_0^1 \theta^n\,d\theta} && (14.82) \\ &\quad= \frac{n+1}{n+2}, && (14.83) \end{aligned}$$

which he put forward as the probability that the sun will rise on day $n+1$ given that it has risen on days 1 through n.

Using the ideas of Kolmogorov complexity and universal probability, we can provide an alternative approach to the problem. Under the universal probability, let us calculate the probability of seeing a 1 next after having observed n 1's in the sequence so far. The conditional probability that the next symbol is a 1 is the ratio of the probability of all sequences with initial segment 1^n and next bit equal to 1 to the probability of all sequences with initial segment 1^n. The simplest programs carry most of the probability; hence we can approximate the probability that the next bit is a 1 with the probability of the program that says "Print 1's forever." Thus,

$$\sum_y p(1^n 1y) \approx p(1^\infty) = c > 0. \tag{14.84}$$

Estimating the probability that the next bit is 0 is more difficult. Since any program that prints $1^n 0\ldots$ yields a description of n, its length should at least be $K(n)$, which for most n is about $\log n + O(\log\log n)$, and hence ignoring second-order terms, we have

$$\sum_y p(1^n 0y) \approx p(1^n 0) \approx 2^{-\log n} \approx \frac{1}{n}. \tag{14.85}$$

Hence, the conditional probability of observing a 0 next is

$$p(0|1^n) = \frac{p(1^n 0)}{p(1^n 0) + p(1^\infty)} \approx \frac{1}{cn+1}, \tag{14.86}$$

which is similar to the result $p(0|1^n) = 1/(n+1)$ derived by Laplace.

This type of argument is a special case of Occam's Razor, a general principle governing scientific research, weighting possible explanations by their complexity. William of Occam said "Nunquam ponenda est pluralitas sine necesitate": Explanations should not be multiplied beyond necessity [516]. In the end, we choose the simplest explanation that is consistent with the data observed. For example, it is easier to accept the general theory of relativity than it is to accept a correction factor of c/r^3 to the gravitational law to explain the precession of the perihelion of Mercury, since the general theory explains more with fewer assumptions than does a "patched" Newtonian theory.

14.11 KOLMOGOROV COMPLEXITY AND UNIVERSAL PROBABILITY

We now prove an equivalence between Kolmogorov complexity and universal probability. We begin by repeating the basic definitions.

$$K(x) = \min_{p:\mathcal{U}(p)=x} l(p) \tag{14.87}$$

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)}. \tag{14.88}$$

Theorem 14.11.1 (*Equivalence of* $K(x)$ *and* $\log \frac{1}{P_{\mathcal{U}}(x))}$.) *There exists a constant c, independent of x, such that*

$$2^{-K(x)} \le P_{\mathcal{U}}(x) \le c2^{-K(x)} \tag{14.89}$$

for all strings x. Thus, the universal probability of a string x is determined essentially by its Kolmogorov complexity.

Remark This implies that $K(x)$ and $\log \frac{1}{P_{\mathcal{U}}(x)}$ have *equal status* as universal complexity measures, since

$$K(x) - c' \le \log \frac{1}{P_{\mathcal{U}}(x)} \le K(x). \tag{14.90}$$

Recall that the complexity defined with respect to two different computers $K_{\mathcal{U}}$ and $K_{\mathcal{U}'}$ are essentially equivalent complexity measures if $|K_{\mathcal{U}}(x) - K_{\mathcal{U}'}(x)|$ is bounded. Theorem 14.11.1 shows that $K_{\mathcal{U}}(x)$ and $\log \frac{1}{P_{\mathcal{U}}(x)}$ are essentially equivalent complexity measures.

Notice the striking similarity between the relationship of $K(x)$ and $\log \frac{1}{P_{\mathcal{U}}(x)}$ in Kolmogorov complexity and the relationship of $H(X)$ and $\log \frac{1}{p(x)}$ in information theory. The ideal Shannon code length assignment

$l(x) = \log \frac{1}{p(x)}$ achieves an *average* description length $H(X)$, while in Kolmogorov complexity theory, the ideal description length $\log \frac{1}{P_{\mathcal{U}}(x)}$ is almost *equal* to $K(X)$. Thus, $\log \frac{1}{p(x)}$ is the natural notion of descriptive complexity of x in algorithmic as well as probabilistic settings.

The upper bound in (14.90) is obvious from the definitions, but the lower bound is more difficult to prove. The result is very surprising, since there are an infinite number of programs that print x. From any program it is possible to produce longer programs by padding the program with irrelevant instructions. The theorem proves that although there are an infinite number of such programs, the universal probability is essentially determined by the largest term, which is $2^{-K(x)}$. If $P_{\mathcal{U}}(x)$ is large, $K(x)$ is small, and vice versa.

However, there is another way to look at the upper bound that makes it less surprising. Consider any computable probability mass function on strings $p(x)$. Using this mass function, we can construct a Shannon–Fano code (Section 5.9) for the source and then describe each string by the corresponding codeword, which will have length $\log \frac{1}{p(x)}$. Hence, for any computable distribution, we can construct a description of a string using not more than $\log \frac{1}{p(x)} + c$ bits, which is an upper bound on the Kolmogorov complexity $K(x)$. Even though $P_{\mathcal{U}}(x)$ is not a computable probability mass function, we are able to finesse the problem using the rather involved tree construction procedure described below.

Proof: (*of Theorem 14.11.1*). The first inequality is simple. Let p^* be the shortest program for x. Then

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)} \geq 2^{-l(p^*)} = 2^{-K(x)}, \tag{14.91}$$

as we wished to show.

We can rewrite the second inequality as

$$K(x) \leq \log \frac{1}{P_{\mathcal{U}}(x)} + c. \tag{14.92}$$

Our objective in the proof is to find a short program to describe the strings that have high $P_{\mathcal{U}}(x)$. An obvious idea is some kind of Huffman coding based on $P_{\mathcal{U}}(x)$, but $P_{\mathcal{U}}(x)$ cannot be calculated effectively, hence a procedure using Huffman coding is not implementable on a computer. Similarly, the process using the Shannon–Fano code also cannot be implemented. However, if we have the Shannon–Fano code tree, we can reconstruct the string by looking for the corresponding node in the tree. This is the basis for the following tree construction procedure.

To overcome the problem of noncomputability of $P_{\mathcal{U}}(x)$, we use a modified approach, trying to construct a code tree directly. Unlike Huffman coding, this approach is not optimal in terms of minimum expected codeword length. However, it is good enough for us to derive a code for which each codeword for x has a length that is within a constant of $\log \frac{1}{P_{\mathcal{U}}(x)}$.

Before we get into the details of the proof, let us outline our approach. We want to construct a code tree in such a way that strings with high probability have low depth. Since we cannot calculate the probability of a string, we do not know a priori the depth of the string on the tree. Instead, we assign x successively to the nodes of the tree, assigning x to nodes closer and closer to the root as our estimate of $P_{\mathcal{U}}(x)$ improves. We want the computer to be able to recreate the tree and use the lowest depth node corresponding to the string x to reconstruct the string.

We now consider the set of programs and their corresponding outputs $\{(p, x)\}$. We try to assign these pairs to the tree. But we immediately come across a problem—there are an infinite number of programs for a given string, and we do not have enough nodes of low depth. However, as we shall show, if we trim the list of program-output pairs, we will be able to define a more manageable list that can be assigned to the tree. Next, we demonstrate the existence of programs for x of length $\log \frac{1}{P_{\mathcal{U}}(x)}$.

Tree construction procedure: For the universal computer $\mathcal{U}$, we simulate all programs using the technique explained in Section 14.8. We list all binary programs:

$$\Lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots. \tag{14.93}$$

Then let the computer execute one clock cycle of Λ for the first stage. In the next stage, let the computer execute two clock cycles of Λ and two clock cycles of the program 0. In the third stage, let the computer execute three clock cycles of each of the first three programs, and so on. In this way, the computer will eventually run all possible programs and run them for longer and longer times, so that if any program halts, it will be discovered to halt eventually. We use this method to produce a list of all programs that halt in the order in which they halt, together with their associated outputs. For each program and its corresponding output, (p_k, x_k), we calculate n_k, which is chosen so that it corresponds to the current estimate of $P_{\mathcal{U}}(x)$. Specifically,

$$n_k = \left\lceil \log \frac{1}{\hat{P}_{\mathcal{U}}(x_k)} \right\rceil, \tag{14.94}$$

where

$$\hat{P}_{\mathcal{U}}(x_k) = \sum_{(p_i,x_i):x_i=x_k,i\leq k} 2^{-l(p_i)}. \tag{14.95}$$

Note that $\hat{P}_{\mathcal{U}}(x_k) \uparrow P_{\mathcal{U}}(x)$ on the subsequence of times k such that $x_k = x$. We are now ready to construct a tree. As we add to the list of triplets, (p_k, x_k, n_k), of programs that halt, we map some of them onto nodes of a binary tree. For purposes of the construction, we must ensure that all the n_i's corresponding to a particular x_k are distinct. To ensure this, we remove from the list all triplets that have the same x and n as a previous triplet. This will ensure that there is at most one node at each level of the tree that corresponds to a given x.

Let $\{(p_i', x_i', n_i') : i = 1, 2, 3, \ldots\}$ denote the new list. On the winnowed list, we assign the triplet (p_k', x_k', n_k') to the first available node at level $n_k' + 1$. As soon as a node is assigned, all of its descendants become unavailable for assignment. (This keeps the assignment prefix-free.)

We illustrate this by means of an example:

$$\begin{aligned}
&(p_1, x_1, n_1) = (10111, 1110, 5), && n_1 = 5 \text{ because } P_{\mathcal{U}}(x_1) \geq 2^{-l(p_1)} = 2^{-5}\\
&(p_2, x_2, n_2) = (11, 10, 2), && n_2 = 2 \text{ because } P_{\mathcal{U}}(x_2) \geq 2^{-l(p_2)} = 2^{-2}\\
&(p_3, x_3, n_3) = (0, 1110, 1), && n_3 = 1 \text{ because } P_{\mathcal{U}}(x_3) \geq 2^{-l(p_3)} + 2^{-l(p_1)}\\
& && \qquad = 2^{-5} + 2^{-1}\\
& && \qquad \geq 2^{-1}\\
&(p_4, x_4, n_4) = (1010, 1111, 4), && n_4 = 4 \text{ because } P_{\mathcal{U}}(x_4) \geq 2^{-l(p_4)} = 2^{-4}\\
&(p_5, x_5, n_5) = (101101, 1110, 1), && n_5 = 1 \text{ because } P_{\mathcal{U}}(x_5) \geq 2^{-1} + 2^{-5} + 2^{-5}\\
& && \qquad \geq 2^{-1}\\
&(p_6, x_6, n_6) = (100, 1, 3), && n_6 = 3 \text{ because } P_{\mathcal{U}}(x_6) \geq 2^{-l(p_6)} = 2^{-3}.\\
&\quad \vdots
\end{aligned} \tag{14.96}$$

We note that the string $x = (1110)$ appears in positions 1, 3 and 5 in the list, but $n_3 = n_5$. The estimate of the probability $\hat{P}_{\mathcal{U}}(1110)$ has not jumped sufficiently for (p_5, x_5, n_5) to survive the cut. Thus the winnowed list becomes

$$\begin{aligned}
(p_1', x_1', n_1') &= (10111, 1110, 5),\\
(p_2', x_2', n_2') &= (11, 10, 2),\\
(p_3', x_3', n_3') &= (0, 1110, 1),\\
(p_4', x_4', n_4') &= (1010, 1111, 4),\\
(p_5', x_5', n_5') &= (100, 1, 3),\\
&\ \,\vdots
\end{aligned} \tag{14.97}$$

The assignment of the winnowed list to nodes of the tree is illustrated in Figure 14.3.

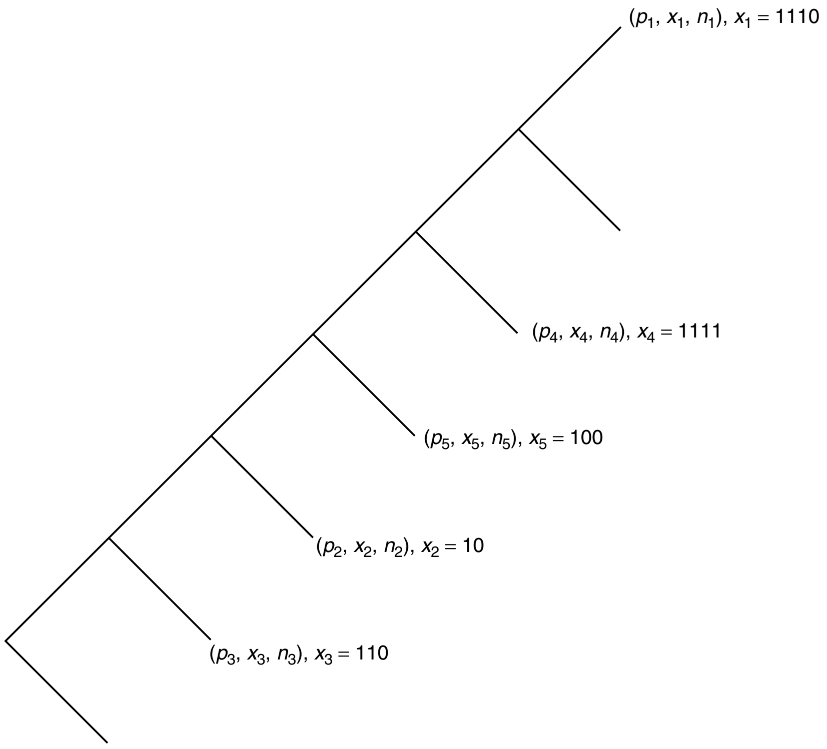

(p_1, x_1, n_1), $x_1 = 1110$
(p_4, x_4, n_4), $x_4 = 1111$
(p_5, x_5, n_5), $x_5 = 100$
(p_2, x_2, n_2), $x_2 = 10$
(p_3, x_3, n_3), $x_3 = 110$

FIGURE 14.3. Assignment of nodes.

In the example, we are able to find nodes at level $n_k + 1$ to which we can assign the triplets. Now we shall prove that there are always enough nodes so that the assignment can be completed. We can perform the assignment of triplets to nodes if and only if the Kraft inequality is satisfied.

We now drop the primes and deal only with the winnowed list illustrated in (14.97). We start with the infinite sum in the Kraft inequality and split it according to the output strings:

$$\sum_{k=1}^{\infty} 2^{-(n_k+1)} = \sum_{x \in \{0,1\}^*} \sum_{k:x_k=x} 2^{-(n_k+1)}. \tag{14.98}$$

We then write the inner sum as

$$\sum_{k:x_k=x} 2^{-(n_k+1)} = 2^{-1} \sum_{k:x_k=x} 2^{-n_k} \tag{14.99}$$

$$\le 2^{-1}\left(2^{\lfloor \log P_{\mathcal{U}}(x)\rfloor} + 2^{\lfloor \log P_{\mathcal{U}}(x)\rfloor - 1} + 2^{\lfloor \log P_{\mathcal{U}}(x)\rfloor - 2} + \cdots\right) \tag{14.100}$$

$$= 2^{-1}2^{\lfloor \log P_{\mathcal{U}}(x)\rfloor}\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right) \tag{14.101}$$

$$= 2^{-1}2^{\lfloor \log P_{\mathcal{U}}(x)\rfloor}2 \tag{14.102}$$

$$\le P_{\mathcal{U}}(x), \tag{14.103}$$

where (14.100) is true because there is at most one node at each level that prints out a particular x. More precisely, the n_k's on the winnowed list for a particular output string x are all different integers. Hence,

$$\sum_k 2^{-(n_k+1)} \le \sum_x \sum_{k:x_k=x} 2^{-(n_k+1)} \le \sum_x P_{\mathcal{U}}(x) \le 1, \tag{14.104}$$

and we can construct a tree with the nodes labeled by the triplets.

If we are given the tree constructed above, it is easy to identify a given x by the path to the lowest depth node that prints x. Call this node $\tilde{p}$. (By construction, $l(\tilde{p}) \le \log \frac{1}{P_{\mathcal{U}}(x)} + 2$.) To use this tree in a program to print x, we specify $\tilde{p}$ and ask the computer to execute the foregoing simulation of all programs. Then the computer will construct the tree as described above and wait for the particular node $\tilde{p}$ to be assigned. Since the computer executes the same construction as the sender, eventually the node $\tilde{p}$ will be assigned. At this point, the computer will halt and print out the x assigned to that node.

This is an effective (finite, mechanical) procedure for the computer to reconstruct x. However, there is no effective procedure to find the lowest depth node corresponding to x. All that we have proved is that there is an (infinite) tree with a node corresponding to x at level $\lceil \log \frac{1}{P_{\mathcal{U}}(x)} \rceil + 1$. But this accomplishes our purpose.

With reference to the example, the description of $x = 1110$ is the path to the node (p_3, x_3, n_3) (i.e., 01), and the description of $x = 1111$ is the path 00001. If we wish to describe the string 1110, we ask the computer to perform the (simulation) tree construction until node 01 is assigned. Then we ask the computer to execute the program corresponding to node 01 (i.e., p_3). The output of this program is the desired string, $x = 1110$.

The length of the program to reconstruct x is essentially the length of the description of the position of the lowest depth node $\tilde{p}$ corresponding to x in the tree. The length of this program for x is $l(\tilde{p}) + c$, where

$$l(\tilde{p}) \leq \left\lceil \log \frac{1}{P_{\mathcal{U}}(x)} \right\rceil + 1, \tag{14.105}$$

and hence the complexity of x satisfies

$$K(x) \leq \left\lceil \log \frac{1}{P_{\mathcal{U}}(x)} \right\rceil + c. \tag{14.106}$$

14.12 KOLMOGOROV SUFFICIENT STATISTIC

Suppose that we are given a sample sequence from a Bernoulli(θ) process. What are the regularities or deviations from randomness in this sequence? One way to address the question is to find the Kolmogorov complexity $K(x^n|n)$, which we discover to be roughly $nH_0(\theta) + \log n + c$. Since, for $\theta \neq \frac{1}{2}$, this is much less than n, we conclude that x^n has structure and is not randomly drawn Bernoulli($\frac{1}{2}$). But what is the structure? The first attempt to find the structure is to investigate the shortest program p^* for x^n. But the shortest description of p^* is about as long as p^* itself; otherwise, we could further compress the description of x^n, contradicting the minimality of p^*. So this attempt is fruitless.

A hint at a good approach comes from an examination of the way in which p^* describes x^n. The program "The sequence has k 1's; of such sequences, it is the ith" is optimal to first order for Bernoulli(θ) sequences. We note that it is a two-stage description, and all of the structure of the sequence is captured in the first stage. Moreover, x^n is maximally complex given the first stage of the description. The first stage, the description of k, requires $\log(n+1)$ bits and defines a set $S = \{x \in \{0,1\}^n : \sum x_i = k\}$. The second stage requires $\log|S| = \log\binom{n}{k} \approx nH_0(\overline{x}_n) \approx nH_0(\theta)$ bits and reveals nothing extraordinary about x^n.

We mimic this process for general sequences by looking for a simple set S that contains x^n. We then follow it with a brute-force description of x^n in S using $\log|S|$ bits. We begin with a definition of the smallest set containing x^n that is describable in no more than k bits.

Definition The *Kolmogorov structure function* $K_k(x^n|n)$ of a binary string $x \in \{0,1\}^n$ is defined as

$$K_k(x^n|n) = \min_{\substack{p\,:\,l(p)\leq k \\ \mathcal{U}(p,n)=S \\ x^n \in S \subseteq \{0,1\}^n}} \log|S|. \tag{14.107}$$

The set S is the smallest set that can be described with no more than k bits and which includes x^n. By $\mathcal{U}(p, n) = S$, we mean that running the program p with data n on the universal computer $\mathcal{U}$ will print out the indicator function of the set S.

Definition For a given small constant c, let k^* be the least k such that

$$K_k(x^n|n) + k \le K(x^n|n) + c. \tag{14.108}$$

Let S^{**} be the corresponding set and let p^{**} be the program that prints out the indicator function of S^{**}. Then we shall say that p^{**} is a *Kolmogorov minimal sufficient statistic* for x^n.

Consider the programs p^* describing sets S^* such that

$$K_k(x^n|n) + k = K(x^n|n). \tag{14.109}$$

All the programs p^* are "sufficient statistics" in that the complexity of x^n given S^* is maximal. But the minimal sufficient statistic is the shortest "sufficient statistic."

The equality in the definition above is up to a large constant depending on the computer U. Then k^* corresponds to the least k for which the two-stage description of x^n is as good as the best single-stage description of x^n. The second stage of the description merely provides the index of x^n within the set S^{**}; this takes $K_k(x^n|n)$ bits if x^n is conditionally maximally complex given the set S^{**}. Hence the set S^{**} captures all the structure within x^n. The remaining description of x^n within S^{**} is essentially the description of the randomness within the string. Hence S^{**} or p^{**} is called the *Kolmogorov sufficient statistic* for x^n.

This is parallel to the definition of a sufficient statistic in mathematical statistics. A statistic T is said to be *sufficient* for a parameter θ if the distribution of the sample given the sufficient statistic is independent of the parameter; that is,

$$\theta \to T(X) \to X \tag{14.110}$$

forms a Markov chain in that order. For the Kolmogorov sufficient statistic, the program p^{**} is sufficient for the "structure" of the string x^n; the remainder of the description of x^n is essentially independent of the "structure" of x^n. In particular, x^n is maximally complex given S^{**}.

A typical graph of the structure function is illustrated in Figure 14.4. When $k = 0$, the only set that can be described is the entire set $\{0, 1\}^n$,

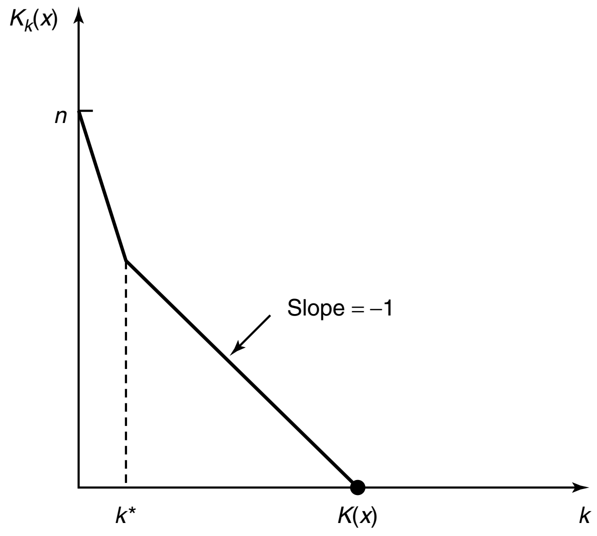


FIGURE 14.4. Kolmogorov sufficient statistic.

so that the corresponding log set size is n. As we increase k, the size of the set drops rapidly until

$$k + K_k(x^n|n) \approx K(x^n|n). \tag{14.111}$$

After this, each additional bit of k reduces the set by half, and we proceed along the line of slope -1 until $k = K(x^n|n)$. For $k \geq K(x^n|n)$, the smallest set that can be described that includes x^n is the singleton $\{x^n\}$, and hence $K_k(x^n|n) = 0$.

We will now illustrate the concept with a few examples.

1. *Bernoulli(θ) sequence*. Consider a sample of length n from a Bernoulli sequence with an unknown parameter θ. As discussed in Example 14.2, we can describible this sequence with $nH\left(\frac{k}{n}\right) + \frac{1}{2}\log n$ bits using a two stage description where we describe k in the first stage (using $\log n$ bits) and then describe the sequence within all sequences with k ones (using $\log \binom{n}{k}$ bits). However, we can use an even shorter first stage description. Instead of describing k exactly, we divide the range of k into bins and describe k only to an accuracy of $\sqrt{\frac{k}{n}\frac{n-k}{n}}\sqrt{n}$ using $\frac{1}{2}\log n$ bits. Then we describe the actual

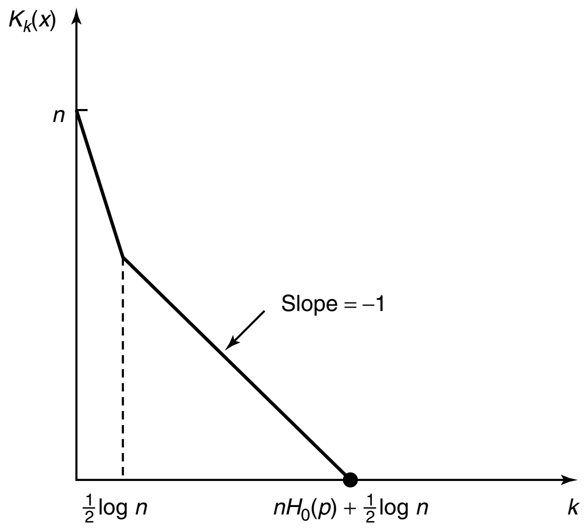


FIGURE 14.5. Kolmogorov sufficient statistic for a Bernoulli sequence.

sequence among all sequences whose type is in the same bin as k. The size of the set of all sequences with l ones, $l \in k \pm \sqrt{\frac{k}{n}\frac{n-k}{n}}\sqrt{n}$ is $nH\binom{k}{n} + o(n)$ by Stirling's formula, so the total description length is still $nH\binom{k}{n} + \frac{1}{2}\log n + o(n)$, but the description length of the Kolmogorov sufficient statistics is $k^* \approx \frac{1}{n}\log n$.

2. *Sample from a Markov chain*. In the same vein as the preceding example, consider a sample from a first-order binary Markov chain. In this case again, p^{**} will correspond to describing the Markov type of the sequence (the number of occurrences of 00's, 01's, 10's, and 11's in the sequence); this conveys all the structure in the sequence. The remainder of the description will be the index of the sequence in the set of all sequences of this Markov type. Hence, in this case, $k^* \approx 2(\frac{1}{2}\log n) = \log n$, corresponding to describing two elements of the conditional joint type to appropriate accuracy. (The other elements of the conditional joint type can be determined from these two.)
3. *Mona Lisa*. Consider an image that consists of a gray circle on a white background. The circle is not uniformly gray but Bernoulli with parameter θ. This is illustrated in Figure 14.6. In this case, the best two-stage description is first to describe the size and position of

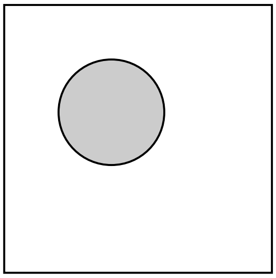


FIGURE 14.6. Mona Lisa.

the circle and its average gray level and then to describe the index of the circle among all the circles with the same gray level. Assuming an n-pixel image (of size $\sqrt{n}$ by $\sqrt{n}$), there are about $n + 1$ possible gray levels, and there are about $(\sqrt{n})^3$ distinguishable circles. Hence, $k^* \approx \frac{5}{2} \log n$ in this case.

14.13 MINIMUM DESCRIPTION LENGTH PRINCIPLE

A natural extension of Occam's razor occurs when we need to describe data drawn from an unknown distribution. Let $X_1, X_2, \ldots, X_n$ be drawn i.i.d. according to probability mass function $p(x)$. We assume that we do not know $p(x)$, but know that $p(x) \in \mathcal{P}$, a class of probability mass functions. Given the data, we can estimate the probability mass function in $\mathcal{P}$ that best fits the data. For simple classes $\mathcal{P}$ (e.g., if $\mathcal{P}$ has only finitely many distributions), the problem is straightforward, and the maximum likelihood procedure [i.e., find $\hat{p} \in \mathcal{P}$ that maximizes $\hat{p}(X_1, X_2, \ldots, X_n)$] works well. However, if the class $\mathcal{P}$ is rich enough, there is a problem of overfitting the data. For example, if $X_1, X_2, \ldots, X_n$ are continuous random variables, and if $\mathcal{P}$ is the set of all probability distributions, the maximum likelihood estimator given $X_1, X_2, \ldots, X_n$ is a distribution that places a single mass point of weight $\frac{1}{n}$ at each observed value. Clearly, this estimate is too closely tied to actual observed data and does not capture any of the structure of the underlying distribution.

To get around this problem, various methods have been applied. In the simplest case, the data are assumed to come from some parametric distribution (e.g., the normal distribution), and the parameters of the distribution are estimated from the data. To validate this method, the data should be tested to check whether the distribution "looks" normal, and if the data pass the test, we could use this description of the data. A more general procedure is to take the maximum likelihood estimate and smooth it out to obtain a smooth density. With enough data, and appropriate smoothness conditions, it is possible to make good estimates of the original density. This process is called *kernel density estimation*.

However, the theory of Kolmogorov complexity (or the Kolmogorov sufficient statistic) suggests a different procedure: Find the $p \in \mathcal{P}$ that minimizes

$$L_p(X_1, X_2, \ldots, X_n) = K(p) + \log \frac{1}{p(X_1, X_2, \ldots, X_n)}. \tag{14.112}$$

This is the length of a two-stage description of the data, where we first describe the distribution p and then, given the distribution, construct the Shannon code and describe the data using $\log \frac{1}{p(X_1, X_2, \ldots, X_n)}$ bits. This procedure is a special case of what is termed the *minimum description length* (MDL) *principle*: Given data and a choice of models, choose the model such that the description of the model plus the conditional description of the data is as short as possible.

SUMMARY

Definition. The *Kolmogorov complexity* $K(x)$ of a string x is

$$K(x) = \min_{p:\mathcal{U}(p)=x} l(p) \tag{14.113}$$

$$K(x|l(x)) = \min_{p:\mathcal{U}(p,l(x))=x} l(p). \tag{14.114}$$

Universality of Kolmogorov complexity. There exists a universal computer $\mathcal{U}$ such that for any other computer $\mathcal{A}$,

$$K_{\mathcal{U}}(x) \le K_{\mathcal{A}}(x) + c_{\mathcal{A}} \tag{14.115}$$

for any string x, where the constant $c_{\mathcal{A}}$ does not depend on x. If $\mathcal{U}$ and $\mathcal{A}$ are universal, $|K_{\mathcal{U}}(x) - K_{\mathcal{A}}(x)| \le c$ for all x.

Upper bound on Kolmogorov complexity

$$K(x|l(x)) \le l(x) + c \tag{14.116}$$

$$K(x) \le K(x|l(x)) + 2\log l(x) + c. \tag{14.117}$$

Kolmogorov complexity and entropy. If $X_1, X_2, \ldots$ are i.i.d. integer-valued random variables with entropy H, there exists a constant c such that for all n,

$$H \le \frac{1}{n} E K(X^n|n) \le H + |\mathcal{X}| \frac{\log n}{n} + \frac{c}{n}. \tag{14.118}$$

Lower bound on Kolmogorov complexity. There are no more than 2^k strings x with complexity $K(x) < k$. If $X_1, X_2, \ldots, X_n$ are drawn according to a Bernoulli($\frac{1}{2}$) process,

$$\Pr\left(K(X_1X_2\ldots X_n|n) \le n - k\right) \le 2^{-k}. \tag{14.119}$$

Definition A sequence x is said to be *incompressible* if $K(x_1x_2\ldots x_n|n)/n \to 1$.

Strong law of large numbers for incompressible sequences

$$\frac{K(x_1, x_2, \ldots, x_n)}{n} \to 1 \Rightarrow \frac{1}{n}\sum_{i=1}^{n} x_i \to \frac{1}{2}. \tag{14.120}$$

Definition The *universal probability* of a string x is

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)} = \Pr(\mathcal{U}(p) = x). \tag{14.121}$$

Universality of $P_{\mathcal{U}}(x)$. For every computer $\mathcal{A}$,

$$P_{\mathcal{U}}(x) \ge c_{\mathcal{A}} P_{\mathcal{A}}(x) \tag{14.122}$$

for every string $x \in \{0, 1\}^*$, where the constant $c_{\mathcal{A}}$ depends only on $\mathcal{U}$ and $\mathcal{A}$.

Definition $\Omega = \sum_{p:\mathcal{U}(p) \text{ halts}} 2^{-l(p)} = \Pr(\mathcal{U}(p) \text{ halts})$ is the probability that the computer halts when the input p to the computer is a binary string drawn according to a Bernoulli($\frac{1}{2}$) process.

Properties of Ω

1. Ω *is not computable*.
2. Ω *is a "philosopher's stone"*.
3. Ω *is algorithmically random (incompressible)*.

Equivalence of K(x) and log $\left(\frac{1}{P_{\mathcal{U}}(x)}\right)$. There exists a constant c independent of x such that

$$\left|\log \frac{1}{P_{\mathcal{U}}(x)} - K(x)\right| \le c \tag{14.123}$$

for all strings x. Thus, the universal probability of a string x is essentially determined by its Kolmogorov complexity.

Definition The *Kolmogorov structure function* $K_k(x^n|n)$ of a binary string $x^n \in \{0,1\}^n$ is defined as

$$K_k(x^n|n) = \min_{\substack{p\,:\,l(p)\le k\\ \mathcal{U}(p,n)=S\\ x\in S}} \log|S|. \tag{14.124}$$

Definition Let k^* be the least k such that

$$K_{k^*}(x^n|n) + k^* = K(x^n|n). \tag{14.125}$$

Let S^{**} be the corresponding set and let p^{**} be the program that prints out the indicator function of S^{**}. Then p^{**} is the *Kolmogorov minimal sufficient statistic* for x.

PROBLEMS

14.1 *Kolmogorov complexity of two sequences.* Let $x, y \in \{0,1\}^*$. Argue that $K(x,y) \le K(x) + K(y) + c$.

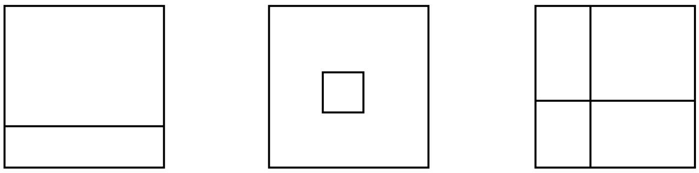
14.2 *Complexity of the sum*

(a) Argue that $K(n) \le \log n + 2\log\log n + c$.

(b) Argue that $K(n_1 + n_2) \le K(n_1) + K(n_2) + c$.

(c) Give an example in which n_1 and n_2 are complex but the sum is relatively simple.

14.3 *Images.* Consider an $n \times n$ array x of 0's and 1's . Thus, x has n^2 bits.



Find the Kolmogorov complexity $K(x \mid n)$ (to first order) if:

(a) x is a horizontal line.

(b) x is a square.

(c) x is the union of two lines, each line being vertical or horizontal.

14.4 *Do computers reduce entropy?* Feed a random program P into an universal computer. What is the entropy of the corresponding output? Specifically, let $X = \mathcal{U}(P)$, where P is a Bernoulli($\frac{1}{2}$) sequence. Here the binary sequence X is either undefined or is in $\{0, 1\}^*$. Let $H(X)$ be the Shannon entropy of X. Argue that $H(X) = \infty$. Thus, although the computer turns nonsense into sense, the output entropy is still infinite.

14.5 *Monkeys on a computer*. Suppose that a random program is typed into a computer. Give a rough estimate of the probability that the computer prints the following sequence:

(a) 0^n followed by any arbitrary sequence.

(b) $\pi_1\pi_2 \ldots \pi_n$ followed by any arbitrary sequence, where π_i is the ith bit in the expansion of π.

(c) 0^n1 followed by any arbitrary sequence.

(d) $\omega_1\omega_2 \ldots \omega_n$ followed by any arbitrary sequence.

(e) A proof of the four-color theorem.

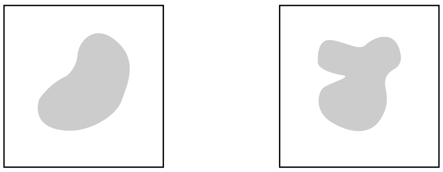
14.6 *Kolmogorov complexity and ternary programs*. Suppose that the input programs for a universal computer $\mathcal{U}$ are sequences in $\{0, 1, 2\}^*$ (ternary inputs). Also, suppose that $\mathcal{U}$ prints ternary outputs. Let $K(x|l(x)) = \min_{U(p,l(x))=x} l(p)$. Show that:

(a) $K(x^n|n) \leq n + c$.

(b) $|x^n \in \{0, 1\}^* : K(x^n|n) < k| < 3^k$.

14.7 *Law of large numbers*. Using ternary inputs and outputs as in Problem 14.14.6, outline an argument demonstrating that if a sequence x is algorithmically random [i.e., if $K(x|l(x)) \approx l(x)$], the proportion of 0's, 1's, and 2's in x must each be near $\frac{1}{3}$. It may be helpful to use Stirling's approximation $n! \approx (n/e)^n$.

14.8 *Image complexity*. Consider two binary subsets A and B (of an $n \times n$ grid): for example,



Find general upper and lower bounds, in terms of $K(A|n)$ and $K(B|n)$, for:

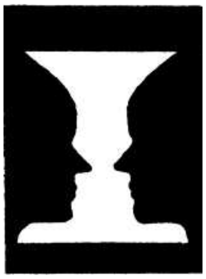
(a) $K(A^c|n)$.

(b) $K(A \cup B|n)$.

(c) $K(A \cap B|n)$.

14.9 *Random program*. Suppose that a random program (symbols i.i.d. uniform over the symbol set) is fed into the nearest available computer. To our surprise the first n bits of the binary expansion of $1/\sqrt{2}$ are printed out. Roughly what would you say the probability is that the next output bit will agree with the corresponding bit in the expansion of $1/\sqrt{2}$?

14.10 *Face–vase illusion*

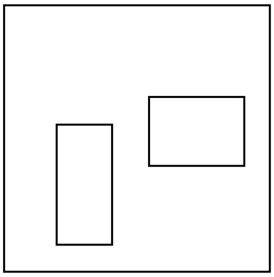


(a) What is an upper bound on the complexity of a pattern on an $m \times m$ grid that has mirror-image symmetry about a vertical axis through the center of the grid *and* consists of horizontal line segments?

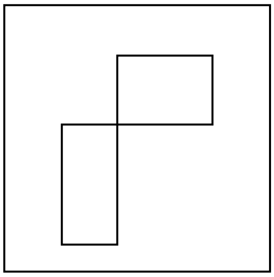
(b) What is the complexity K if the image differs in one cell from the pattern described above?

14.11 *Kolmogorov complexity*. Assume that n is very large and known. Let all rectangles be parallel to the frame.

(a) What is the (maximal) Kolmogorov complexity of the union of two rectangles on an $n \times n$ grid?



(b) What if the rectangles intersect at a corner?



(c) What if they have the same (unknown) shape?

(d) What if they have the same (unknown) area?

(e) What is the minimum Kolmogorov complexity of the union of two rectangles? That is, what is the simplest union?

(f) What is the (maximal) Kolmogorov complexity over all images (not necessarily rectangles) on an $n \times n$ grid?

14.12 *Encrypted text*. Suppose that English text x^n is encrypted into y^n by a substitution cypher: a 1-to-1 reassignment of each of the 27 letters of the alphabet (A–Z, including the space character) to itself. Suppose that the Kolmogorov complexity of the text x^n is $K(x^n) = \frac{n}{4}$. (This is about right for English text. We're now assuming a 27-symbol programming language, instead of a binary symbol-set for the programming language. So, the length of the shortest program, using a 27-ary programming language, that prints out a particular string of English text of length n, is approximately n/4.)

(a) What is the Kolmogorov complexity of the encryption map?

(b) Estimate the Kolmogorov complexity of the encrypted text y^n.

(c) How high must n be before you would expect to be able to decode y^n?

14.13 *Kolmogorov complexity*. Consider the Kolmogorov complexity $K(n)$ over the integers n. If a specific integer n_1 has a low Kolmogorov complexity $K(n_1)$, by how much can the Kolmogorov complexity $K(n_1 + k)$ for the integer $n_1 + k$ vary from $K(n_1)$?

14.14 *Complexity of large numbers*. Let $A(n)$ be the set of positive integers x for which a terminating program p of length less than or equal to n bits exists that outputs x. Let $B(n)$ be the complement of $A(n)$ [i.e., $B(n)$ is the set of integers x for which no program of length less than or equal to n outputs x]. Let $M(n)$ be the maximum integer in $A(n)$, and let $S(n)$ be the minimum integer in $B(n)$. What is the Kolmogorov complexity $K(M(n))$ (approximately)? What is $K(S(n))$ (approximately)? Which is larger ($M(n)$ or $S(n)$)? Give a reasonable lower bound on $M(n)$ and a reasonable upper bound on $S(n)$.

HISTORICAL NOTES

The original ideas of Kolmogorov complexity were put forth independently and almost simultaneously by Kolmogorov [321, 322], Solomonoff [504], and Chaitin [89]. These ideas were developed further by students of Kolmogorov such as Martin-Löf [374], who defined the notion of algorithmically random sequences and algorithmic tests for randomness, and by Levin and Zvonkin [353], who explored the ideas of universal probability and its relationship to complexity. A series of papers by Chaitin [90]–[92] developed the relationship between algorithmic complexity and mathematical proofs. C. P. Schnorr studied the universal notion of randomness and related it to gambling in [466]–[468].

The concept of the Kolmogorov structure function was defined by Kolmogorov at a talk at the Tallin conference in 1973, but these results were not published. V'yugin pursues this in [549], where he shows that there are some very strange sequences x^n that reveal their structure arbitrarily slowly in the sense that $K_k(x^n|n) = n - k$, $k < K(x^n|n)$. Zurek [606]–[608] addresses the fundamental questions of Maxwell's demon and the second law of thermodynamics by establishing the physical consequences of Kolmogorov complexity.

Rissanen's minimum description length (MDL) principle is very close in spirit to the Kolmogorov sufficient statistic. Rissanen [445, 446] finds a low-complexity model that yields a high likelihood of the data. Barron and Cover [32] argue that the density minimizing $K(f) + \log \frac{1}{\prod f(X_i)}$ yields consistent density estimation.

A nontechnical introduction to the various measures of complexity can be found in a thought-provoking book by Pagels [412]. Additional references to work in the area can be found in a paper by Cover et al. [114] on Kolmogorov's contributions to information theory and algorithmic complexity. A comprehensive introduction to the field, including applications of the theory to analysis of algorithms and automata, may be found in the book by Li and Vitanyi [354]. Additional coverage may be found in the books by Chaitin [86, 93].

CHAPTER 15

NETWORK INFORMATION THEORY

A system with many senders and receivers contains many new elements in the communication problem: interference, cooperation, and feedback. These are the issues that are the domain of network information theory. The general problem is easy to state. Given many senders and receivers and a channel transition matrix that describes the effects of the interference and the noise in the network, decide whether or not the sources can be transmitted over the channel. This problem involves distributed source coding (data compression) as well as distributed communication (finding the capacity region of the network). This general problem has not yet been solved, so we consider various special cases in this chapter.

Examples of large communication networks include computer networks, satellite networks, and the phone system. Even within a single computer, there are various components that talk to each other. A complete theory of network information would have wide implications for the design of communication and computer networks.

Suppose that m stations wish to communicate with a common satellite over a common channel, as shown in Figure 15.1. This is known as a *multiple-access channel*. How do the various senders cooperate with each other to send information to the receiver? What rates of communication are achievable simultaneously? What limitations does interference among the senders put on the total rate of communication? This is the best understood multiuser channel, and the above questions have satisfying answers.

In contrast, we can reverse the network and consider one TV station sending information to m TV receivers, as in Figure 15.2. How does the sender encode information meant for different receivers in a common signal? What are the rates at which information can be sent to the different receivers? For this channel, the answers are known only in special cases.

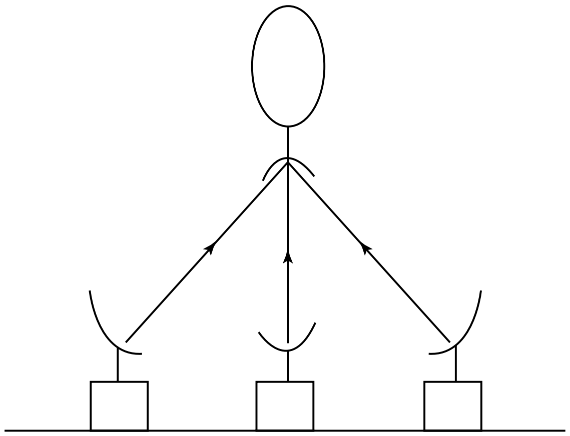


FIGURE 15.1. Multiple-access channel.

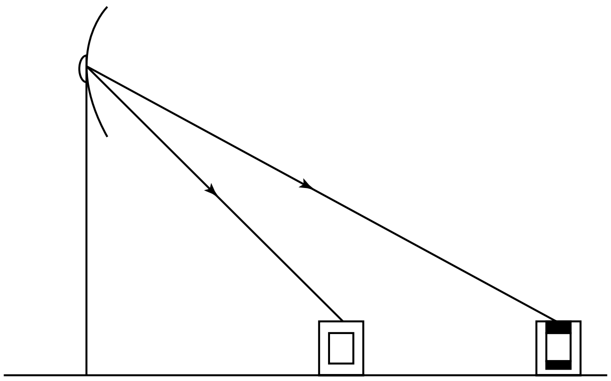


FIGURE 15.2. Broadcast channel.

There are other channels, such as the relay channel (where there is one source and one destination, but one or more intermediate sender–receiver pairs that act as relays to facilitate the communication between the source and the destination), the interference channel (two senders and two receivers with crosstalk), and the two-way channel (two sender–receiver pairs sending information to each other). For all these channels, we have only some of the answers to questions about achievable communication rates and the appropriate coding strategies.

All these channels can be considered special cases of a general communication network that consists of m nodes trying to communicate with each other, as shown in Figure 15.3. At each instant of time, the ith node sends a symbol x_i that depends on the messages that it wants to send

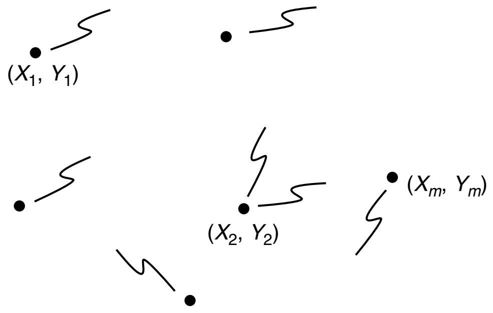


FIGURE 15.3. Communication network.

and on past received symbols at the node. The simultaneous transmission of the symbols $(x_1, x_2, \ldots, x_m)$ results in random received symbols $(Y_1, Y_2, \ldots, Y_m)$ drawn according to the conditional probability distribution $p(y^{(1)}, y^{(2)}, \ldots, y^{(m)}|x^{(1)}, x^{(2)}, \ldots, x^{(m)})$. Here $p(\cdot|\cdot)$ expresses the effects of the noise and interference present in the network. If $p(\cdot|\cdot)$ takes on only the values 0 and 1, the network is deterministic.

Associated with some of the nodes in the network are stochastic data sources, which are to be communicated to some of the other nodes in the network. If the sources are independent, the messages sent by the nodes are also independent. However, for full generality, we must allow the sources to be dependent. How does one take advantage of the dependence to reduce the amount of information transmitted? Given the probability distribution of the sources and the channel transition function, can one transmit these sources over the channel and recover the sources at the destinations with the appropriate distortion?

We consider various special cases of network communication. We consider the problem of source coding when the channels are noiseless and without interference. In such cases, the problem reduces to finding the set of rates associated with each source such that the required sources can be decoded at the destination with a low probability of error (or appropriate distortion). The simplest case for distributed source coding is the Slepian–Wolf source coding problem, where we have two sources that must be encoded separately, but decoded together at a common node. We consider extensions to this theory when only one of the two sources needs to be recovered at the destination.

The theory of flow in networks has satisfying answers in such domains as circuit theory and the flow of water in pipes. For example, for the single-source single-sink network of pipes shown in Figure 15.4, the maximum flow from A to B can be computed easily from the Ford–Fulkerson theorem . Assume that the edges have capacities C_i as shown. Clearly,

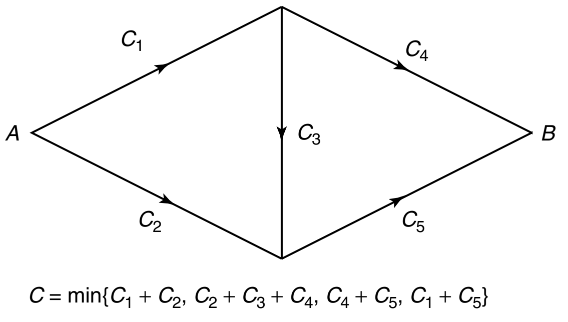

C_1
C_2
C_3
C_4
C_5
A
B
$C = \min\{C_1 + C_2, C_2 + C_3 + C_4, C_4 + C_5, C_1 + C_5\}$

FIGURE 15.4. Network of water pipes.

the maximum flow across any cut set cannot be greater than the sum of the capacities of the cut edges. Thus, minimizing the maximum flow across cut sets yields an upper bound on the capacity of the network. The Ford–Fulkerson theorem [214] shows that this capacity can be achieved.

The theory of information flow in networks does not have the same simple answers as the theory of flow of water in pipes. Although we prove an upper bound on the rate of information flow across any cut set, these bounds are not achievable in general. However, it is gratifying that some problems, such as the relay channel and the cascade channel, admit a simple max-flow min-cut interpretation. Another subtle problem in the search for a general theory is the absence of a source–channel separation theorem, which we touch on briefly in Section 15.10. A complete theory combining distributed source coding and network channel coding is still a distant goal.

In the next section we consider Gaussian examples of some of the basic channels of network information theory. The physically motivated Gaussian channel lends itself to concrete and easily interpreted answers. Later we prove some of the basic results about joint typicality that we use to prove the theorems of multiuser information theory. We then consider various problems in detail: the multiple-access channel, the coding of correlated sources (Slepian–Wolf data compression), the broadcast channel, the relay channel, the coding of a random variable with side information, and the rate distortion problem with side information. We end with an introduction to the general theory of information flow in networks. There are a number of open problems in the area, and there does not yet exist a comprehensive theory of information networks. Even if such a theory is found, it may be too complex for easy implementation. But the theory will be able to tell communication designers how close they are to optimality and perhaps suggest some means of improving the communication rates.

15.1 GAUSSIAN MULTIPLE-USER CHANNELS

Gaussian multiple-user channels illustrate some of the important features of network information theory. The intuition gained in Chapter 9 on the Gaussian channel should make this section a useful introduction. Here the key ideas for establishing the capacity regions of the Gaussian multiple-access, broadcast, relay, and two-way channels will be given without proof. The proofs of the coding theorems for the discrete memoryless counterparts to these theorems are given in later sections of the chapter.

The basic discrete-time additive white Gaussian noise channel with input power P and noise variance N is modeled by

$$Y_i = X_i + Z_i, \qquad i = 1, 2, \ldots, \tag{15.1}$$

where the Z_i are i.i.d. Gaussian random variables with mean 0 and variance N. The signal $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ has a power constraint

$$\frac{1}{n}\sum_{i=1}^{n} X_i^2 \le P. \tag{15.2}$$

The Shannon capacity C is obtained by maximizing $I(X;Y)$ over all random variables X such that $EX^2 \le P$ and is given (Chapter 9) by

$$C = \frac{1}{2}\log\left(1 + \frac{P}{N}\right) \qquad \text{bits per transmission.} \tag{15.3}$$

In this chapter we restrict our attention to discrete-time memoryless channels; the results can be extended to continuous-time Gaussian channels.

15.1.1 Single-User Gaussian Channel

We first review the single-user Gaussian channel studied in Chapter 9. Here $Y = X + Z$. Choose a rate $R < \frac{1}{2}\log(1 + \frac{P}{N})$. Fix a good $(2^{nR}, n)$ codebook of power P. Choose an index w in the set 2^{nR}. Send the wth codeword $\mathbf{X}(w)$ from the codebook generated above. The receiver observes $\mathbf{Y} = \mathbf{X}(w) + \mathbf{Z}$ and then finds the index $\hat{w}$ of the codeword closest to $\mathbf{Y}$. If n is sufficiently large, the probability of error $\Pr(w \ne \hat{w})$ will be arbitrarily small. As can be seen from the definition of joint typicality, this minimum-distance decoding scheme is essentially equivalent to finding the codeword in the codebook that is jointly typical with the received vector $\mathbf{Y}$.

15.1.2 Gaussian Multiple-Access Channel with *m* Users

We consider m transmitters, each with a power P. Let

$$Y = \sum_{i=1}^{m} X_i + Z. \tag{15.4}$$

Let

$$C\left(\frac{P}{N}\right) = \frac{1}{2}\log\left(1 + \frac{P}{N}\right) \tag{15.5}$$

denote the capacity of a single-user Gaussian channel with signal-to-noise ratio P/N. The achievable rate region for the Gaussian channel takes on the simple form given in the following equations:

$$R_i < C\left(\frac{P}{N}\right) \tag{15.6}$$

$$R_i + R_j < C\left(\frac{2P}{N}\right) \tag{15.7}$$

$$R_i + R_j + R_k < C\left(\frac{3P}{N}\right) \tag{15.8}$$

$$\vdots \tag{15.9}$$

$$\sum_{i=1}^{m} R_i < C\left(\frac{mP}{N}\right). \tag{15.10}$$

Note that when all the rates are the same, the last inequality dominates the others.

Here we need m codebooks, the ith codebook having 2^{nR_i} codewords of power P. Transmission is simple. Each of the independent transmitters chooses an arbitrary codeword from its own codebook. The users send these vectors simultaneously. The receiver sees these codewords added together with the Gaussian noise $\mathbf{Z}$.

Optimal decoding consists of looking for the m codewords, one from each codebook, such that the vector sum is closest to $\mathbf{Y}$ in Euclidean distance. If $(R_1, R_2, \ldots, R_m)$ is in the capacity region given above, the probability of error goes to 0 as n tends to infinity.

Remarks It is exciting to see in this problem that the sum of the rates of the users $C(mP/N)$ goes to infinity with m. Thus, in a cocktail party with m celebrants of power P in the presence of ambient noise N, the intended listener receives an unbounded amount of information as the number of people grows to infinity. A similar conclusion holds, of course, for ground communications to a satellite. Apparently, the increasing interference as the number of senders $m \to \infty$ does not limit the total received information.

It is also interesting to note that the optimal transmission scheme here does not involve time-division multiplexing. In fact, each of the transmitters uses all of the bandwidth all of the time.

15.1.3 Gaussian Broadcast Channel

Here we assume that we have a sender of power P and two distant receivers, one with Gaussian noise power N_1 and the other with Gaussian noise power N_2. Without loss of generality, assume that $N_1 < N_2$. Thus, receiver Y_1 is less noisy than receiver Y_2. The model for the channel is $Y_1 = X + Z_1$ and $Y_2 = X + Z_2$, where Z_1 and Z_2 are arbitrarily correlated Gaussian random variables with variances N_1 and N_2, respectively. The sender wishes to send independent messages at rates R_1 and R_2 to receivers Y_1 and Y_2, respectively.

Fortunately, all scalar Gaussian broadcast channels belong to the class of degraded broadcast channels discussed in Section 15.6.2. Specializing that work, we find that the capacity region of the Gaussian broadcast channel is

$$R_1 < C\left(\frac{\alpha P}{N_1}\right) \tag{15.11}$$

$$R_2 < C\left(\frac{(1-\alpha)P}{\alpha P + N_2}\right), \tag{15.12}$$

where α may be arbitrarily chosen ($0 \le \alpha \le 1$) to trade off rate R_1 for rate R_2 as the transmitter wishes.

To encode the messages, the transmitter generates two codebooks, one with power αP at rate R_1, and another codebook with power $\overline{\alpha} P$ at rate R_2, where R_1 and R_2 lie in the capacity region above. Then to send an index $w_1 \in \{1, 2, \ldots, 2^{nR_1}\}$ and $w_2 \in \{1, 2, \ldots, 2^{nR_2}\}$ to Y_1 and Y_2, respectively, the transmitter takes the codeword $\mathbf{X}(w_1)$ from the first codebook and codeword $\mathbf{X}(w_2)$ from the second codebook and computes the sum. He sends the sum over the channel.

The receivers must now decode the messages. First consider the bad receiver Y_2. He merely looks through the second codebook to find the closest codeword to the received vector $\mathbf{Y}_2$. His effective signal-to-noise ratio is $\overline{\alpha}P/(\alpha P + N_2)$, since Y_1's message acts as noise to Y_2. (This can be proved.)

The good receiver Y_1 first decodes Y_2's codeword, which he can accomplish because of his lower noise N_1. He subtracts this codeword $\hat{\mathbf{X}}_2$ from $\mathbf{Y}_1$. He then looks for the codeword in the first codebook closest to $\mathbf{Y}_1 - \hat{\mathbf{X}}_2$. The resulting probability of error can be made as low as desired.

A nice dividend of optimal encoding for degraded broadcast channels is that the better receiver Y_1 always knows the message intended for receiver Y_2 in addition to the message intended for himself.

15.1.4 Gaussian Relay Channel

For the relay channel, we have a sender X and an ultimate intended receiver Y. Also present is the relay channel, intended solely to help the receiver. The Gaussian relay channel (Figure 15.31 in Section 15.7) is given by

$$Y_1 = X + Z_1, \tag{15.13}$$

$$Y = X + Z_1 + X_1 + Z_2, \tag{15.14}$$

where Z_1 and Z_2 are independent zero-mean Gaussian random variables with variance N_1 and N_2, respectively. The encoding allowed by the relay is the causal sequence

$$X_{1i} = f_i(Y_{11}, Y_{12}, \ldots, Y_{1i-1}). \tag{15.15}$$

Sender X has power P and sender X_1 has power P_1. The capacity is

$$C = \max_{0\le\alpha\le1} \min\left\{C\left(\frac{P + P_1 + 2\sqrt{\overline{\alpha}PP_1}}{N_1 + N_2}\right), C\left(\frac{\alpha P}{N_1}\right)\right\}, \tag{15.16}$$

where $\overline{\alpha} = 1 - \alpha$. Note that if

$$\frac{P_1}{N_2} \ge \frac{P}{N_1}, \tag{15.17}$$

it can be seen that $C = C(P/N_1)$,which is achieved by $\alpha = 1$. The channel appears to be noise-free after the relay, and the capacity $C(P/N_1)$ from X to the relay can be achieved. Thus, the rate $C(P/(N_1 + N_2))$ without the relay is increased by the presence of the relay to $C(P/N_1)$. For large

N_2 and for $P_1/N_2 \geq P/N_1$, we see that the increment in rate is from $C(P/(N_1 + N_2)) \approx 0$ to $C(P/N_1)$.

Let $R_1 < C(\alpha P/N_1)$. Two codebooks are needed. The first codebook has 2^{nR_1} words of power αP. The second has 2^{nR_0} codewords of power $\overline{\alpha}P$. We shall use codewords from these codebooks successively to create the opportunity for cooperation by the relay. We start by sending a codeword from the first codebook. The relay now knows the index of this codeword since $R_1 < C(\alpha P/N_1)$, but the intended receiver has a list of possible codewords of size $2^{n(R_1 - C(\alpha P/(N_1+N_2)))}$. This list calculation involves a result on list codes.

In the next block, the transmitter and the relay wish to cooperate to resolve the receiver's uncertainty about the codeword sent previously that is on the receiver's list. Unfortunately, they cannot be sure what this list is because they do not know the received signal Y. Thus, they randomly partition the first codebook into 2^{nR_0} cells with an equal number of codewords in each cell. The relay, the receiver, and the transmitter agree on this partition. The relay and the transmitter find the cell of the partition in which the codeword from the first codebook lies and cooperatively send the codeword from the second codebook with that index. That is, X and X_1 send the same designated codeword. The relay, of course, must scale this codeword so that it meets his power constraint P_1. They now transmit their codewords simultaneously. An important point to note here is that the cooperative information sent by the relay and the transmitter is sent coherently. So the power of the sum as seen by the receiver Y is $(\sqrt{\overline{\alpha}P} + \sqrt{P_1})^2$.

However, this does not exhaust what the transmitter does in the second block. He also chooses a fresh codeword from the first codebook, adds it "on paper" to the cooperative codeword from the second codebook, and sends the sum over the channel.

The reception by the ultimate receiver Y in the second block involves first finding the cooperative index from the second codebook by looking for the closest codeword in the second codebook. He subtracts the codeword from the received sequence and then calculates a list of indices of size 2^{nR_0} corresponding to all codewords of the first codebook that might have been sent in the second block.

Now it is time for the intended receiver to complete computing the codeword from the first codebook sent in the first block. He takes his list of possible codewords that might have been sent in the first block and intersects it with the cell of the partition that he has learned from the cooperative relay transmission in the second block. The rates and powers have been chosen so that it is highly probable that there is only one codeword in the intersection. This is Y's guess about the information sent in the first block.

We are now in steady state. In each new block, the transmitter and the relay cooperate to resolve the list uncertainty from the previous block. In addition, the transmitter superimposes some fresh information from his first codebook to this transmission from the second codebook and transmits the sum. The receiver is always one block behind, but for sufficiently many blocks, this does not affect his overall rate of reception.

15.1.5 Gaussian Interference Channel

The interference channel has two senders and two receivers. Sender 1 wishes to send information to receiver 1. He does not care what receiver 2 receives or understands; similarly with sender 2 and receiver 2. Each channel interferes with the other. This channel is illustrated in Figure 15.5. It is not quite a broadcast channel since there is only one intended receiver for each sender, nor is it a multiple access channel because each receiver is only interested in what is being sent by the corresponding transmitter. For symmetric interference, we have

$$Y_1 = X_1 + aX_2 + Z_1 \tag{15.18}$$

$$Y_2 = X_2 + aX_1 + Z_2, \tag{15.19}$$

where Z_1, Z_2 are independent $\mathcal{N}(0, N)$ random variables. This channel has not been solved in general even in the Gaussian case. But remarkably, in the case of high interference, it can be shown that the capacity region of this channel is the same as if there were no interference whatsoever.

To achieve this, generate two codebooks, each with power P and rate $C(P/N)$. Each sender independently chooses a word from his book and

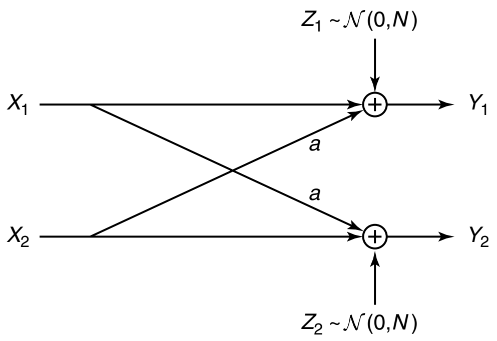


FIGURE 15.5. Gaussian interference channel.

sends it. Now, if the interference a satisfies $C(a^2P/(P+N)) > C(P/N)$, the first transmitter understands perfectly the index of the second transmitter. He finds it by the usual technique of looking for the closest codeword to his received signal. Once he finds this signal, he subtracts it from his waveform received. Now there is a clean channel between him and his sender. He then searches the sender's codebook to find the closest codeword and declares that codeword to be the one sent.

15.1.6 Gaussian Two-Way Channel

The two-way channel is very similar to the interference channel, with the additional provision that sender 1 is attached to receiver 2 and sender 2 is attached to receiver 1, as shown in Figure 15.6. Hence, sender 1 can use information from previous received symbols of receiver 2 to decide what to send next. This channel introduces another fundamental aspect of network information theory: namely, feedback. Feedback enables the senders to use the partial information that each has about the other's message to cooperate with each other.

The capacity region of the two-way channel is not known in general. This channel was first considered by Shannon [486], who derived upper and lower bounds on the region (see Problem 15.15). For Gaussian channels, these two bounds coincide and the capacity region is known; in fact, the Gaussian two-way channel decomposes into two independent channels.

Let P_1 and P_2 be the powers of transmitters 1 and 2, respectively, and let N_1 and N_2 be the noise variances of the two channels. Then the rates $R_1 < C(P_1/N_1)$ and $R_2 < C(P_2/N_2)$ can be achieved by the techniques described for the interference channel. In this case, we generate two codebooks of rates R_1 and R_2. Sender 1 sends a codeword from the first codebook. Receiver 2 receives the sum of the codewords sent by the

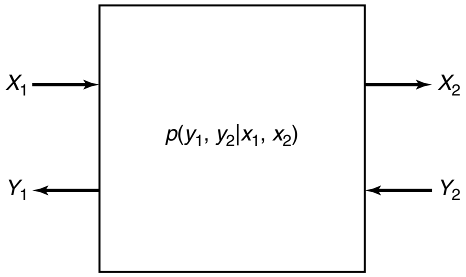


FIGURE 15.6. Two-way channel.

two senders plus some noise. He simply subtracts out the codeword of sender 2 and he has a clean channel from sender 1 (with only the noise of variance N_1). Hence, the two-way Gaussian channel decomposes into two independent Gaussian channels. But this is not the case for the general two-way channel; in general, there is a trade-off between the two senders so that both of them cannot send at the optimal rate at the same time.

15.2 JOINTLY TYPICAL SEQUENCES

We have previewed the capacity results for networks by considering multiuser Gaussian channels. We begin a more detailed analysis in this section, where we extend the joint AEP proved in Chapter 7 to a form that we will use to prove the theorems of network information theory. The joint AEP will enable us to calculate the probability of error for jointly typical decoding for the various coding schemes considered in this chapter.

Let $(X_1, X_2, \ldots, X_k)$ denote a finite collection of discrete random variables with some fixed joint distribution, $p(x^{(1)}, x^{(2)}, \ldots, x^{(k)})$, $(x^{(1)}, x^{(2)}, \ldots, x^{(k)}) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k$. Let S denote an ordered subset of these random variables and consider n independent copies of S. Thus,

$$\Pr\{S = s\} = \prod_{i=1}^{n} \Pr\{S_i = s_i\}, \qquad s \in \mathcal{S}^n. \tag{15.20}$$

For example, if $S = (X_j, X_l)$, then

$$\Pr\{S = s\} = \Pr\left\{(\mathbf{X}_j, \mathbf{X}_l) = (\mathbf{x}_j, \mathbf{x}_l)\right\} \tag{15.21}$$

$$= \prod_{i=1}^{n} p(x_{ij}, x_{il}). \tag{15.22}$$

To be explicit, we will sometimes use $X(S)$ for S. By the law of large numbers, for any subset S of random variables,

$$-\frac{1}{n} \log p(S_1, S_2, \ldots, S_n) = -\frac{1}{n} \sum_{i=1}^{n} \log p(S_i) \to H(S), \tag{15.23}$$

where the convergence takes place with probability 1 for all 2^k subsets, $S \subseteq \{X^{(1)}, X^{(2)}, \ldots, X^{(k)}\}$.

Definition The set $A_\epsilon^{(n)}$ of ϵ-typical n-sequences $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ is defined by

$$\begin{aligned}
&A_\epsilon^{(n)}(X^{(1)}, X^{(2)}, \ldots, X^{(k)}) \\
&= A_\epsilon^{(n)} \\
&= \Bigg\{(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) : \left|-\frac{1}{n}\log p(\mathbf{s}) - H(S)\right| < \epsilon,\ \forall S \subseteq \{X^{(1)}, X^{(2)}, \ldots, \\
&\qquad X^{(k)}\}\Bigg\}. \qquad (15.24)
\end{aligned}$$

Let $A_\epsilon^{(n)}(S)$ denote the restriction of $A_\epsilon^{(n)}$ to the coordinates of S. Thus, if $S = (X_1, X_2)$, we have

$$\begin{aligned}
A_\epsilon^{(n)}(X_1, X_2) = \{(\mathbf{x}_1, \mathbf{x}_2) : \\
&\left|-\frac{1}{n}\log p(\mathbf{x}_1, \mathbf{x}_2) - H(X_1, X_2)\right| < \epsilon, \\
&\left|-\frac{1}{n}\log p(\mathbf{x}_1) - H(X_1)\right| < \epsilon, \\
&\left|-\frac{1}{n}\log p(\mathbf{x}_2) - H(X_2)\right| < \epsilon\}. \qquad (15.25)
\end{aligned}$$

Definition We will use the notation $a_n \doteq 2^{n(b\pm\epsilon)}$ to mean that

$$\left|\frac{1}{n}\log a_n - b\right| < \epsilon \qquad (15.26)$$

for n sufficiently large.

Theorem 15.2.1 *For any $\epsilon > 0$, for sufficiently large n,*

1. $P(A_\epsilon^{(n)}(S)) \ge 1 - \epsilon, \quad \forall S \subseteq \{X^{(1)}, X^{(2)}, \ldots, X^{(k)}\}.$ (15.27)
2. $\mathbf{s} \in A_\epsilon^{(n)}(S) \Longrightarrow p(\mathbf{s}) \doteq 2^{n(H(S)\pm\epsilon)}.$ (15.28)
3. $|A_\epsilon^{(n)}(S)| \doteq 2^{n(H(S)\pm 2\epsilon)}.$ (15.29)

4. *Let* $S_1, S_2 \subseteq \{X^{(1)}, X^{(2)}, \ldots, X^{(k)}\}$. *If* $(\mathbf{s}_1, \mathbf{s}_2) \in A_\epsilon^{(n)}(S_1, S_2)$, *then*

$$p(\mathbf{s}_1|\mathbf{s}_2) \doteq 2^{-n(H(S_1|S_2)\pm 2\epsilon)}. \tag{15.30}$$

Proof

1. This follows from the law of large numbers for the random variables in the definition of $A_\epsilon^{(n)}(S)$.
2. This follows directly from the definition of $A_\epsilon^{(n)}(S)$.
3. This follows from

$$\begin{aligned} 1 &\geq \sum_{\mathbf{s}\in A_\epsilon^{(n)}(S)} p(\mathbf{s}) && (15.31)\\ &\geq \sum_{\mathbf{s}\in A_\epsilon^{(n)}(S)} 2^{-n(H(S)+\epsilon)} && (15.32)\\ &= |A_\epsilon^{(n)}(S)|2^{-n(H(S)+\epsilon)}. && (15.33) \end{aligned}$$

If n is sufficiently large, we can argue that

$$\begin{aligned} 1-\epsilon &\leq \sum_{\mathbf{s}\in A_\epsilon^{(n)}(S)} p(\mathbf{s}) && (15.34)\\ &\leq \sum_{\mathbf{s}\in A_\epsilon^{(n)}(S)} 2^{-n(H(S)-\epsilon)} && (15.35)\\ &= |A_\epsilon^{(n)}(S)|2^{-n(H(S)-\epsilon)}. && (15.36) \end{aligned}$$

Combining (15.33) and (15.36), we have $|A_\epsilon^{(n)}(S)| \doteq 2^{n(H(S)\pm 2\epsilon)}$ for sufficiently large n.

4. For $(\mathbf{s}_1, \mathbf{s}_2) \in A_\epsilon^{(n)}(S_1, S_2)$, we have $p(\mathbf{s}_1) \doteq 2^{-n(H(S_1)\pm\epsilon)}$ and $p(\mathbf{s}_1, \mathbf{s}_2) \doteq 2^{-n(H(S_1,S_2)\pm\epsilon)}$. Hence,

$$p(\mathbf{s}_2|\mathbf{s}_1) = \frac{p(\mathbf{s}_1, \mathbf{s}_2)}{p(\mathbf{s}_1)} \doteq 2^{-n(H(S_2|S_1)\pm 2\epsilon)}. \quad \square \tag{15.37}$$

The next theorem bounds the number of conditionally typical sequences for a given typical sequence.

Theorem 15.2.2 *Let S_1, S_2 be two subsets of $X^{(1)}, X^{(2)}, \ldots, X^{(k)}$. For any $\epsilon > 0$, define $A_\epsilon^{(n)}(S_1|\mathbf{s}_2)$ to be the set of $\mathbf{s}_1$ sequences that are jointly ϵ-typical with a particular $\mathbf{s}_2$ sequence. If $\mathbf{s}_2 \in A_\epsilon^{(n)}(S_2)$, then for sufficiently large n, we have*

$$|A_\epsilon^{(n)}(S_1|\mathbf{s}_2)| \le 2^{n(H(S_1|S_2)+2\epsilon)} \tag{15.38}$$

and

$$(1-\epsilon)2^{n(H(S_1|S_2)-2\epsilon)} \le \sum_{\mathbf{s}_2} p(\mathbf{s}_2)|A_\epsilon^{(n)}(S_1|\mathbf{s}_2)|. \tag{15.39}$$

Proof: As in part 3 of Theorem 15.2.1, we have

$$1 \ge \sum_{\mathbf{s}_1 \in A_\epsilon^{(n)}(S_1|\mathbf{s}_2)} p(\mathbf{s}_1|\mathbf{s}_2) \tag{15.40}$$

$$\ge \sum_{\mathbf{s}_1 \in A_\epsilon^{(n)}(S_1|\mathbf{s}_2)} 2^{-n(H(S_1|S_2)+2\epsilon)} \tag{15.41}$$

$$= |A_\epsilon^{(n)}(\mathbf{S}_1|\mathbf{s}_2)|2^{-n(H(S_1|S_2)+2\epsilon)}. \tag{15.42}$$

If n is sufficiently large, we can argue from (15.27) that

$$1-\epsilon \le \sum_{\mathbf{s}_2} p(\mathbf{s}_2) \sum_{\mathbf{s}_1 \in A_\epsilon^{(n)}(S_1|\mathbf{s}_2)} p(\mathbf{s}_1|\mathbf{s}_2) \tag{15.43}$$

$$\le \sum_{\mathbf{s}_2} p(\mathbf{s}_2) \sum_{\mathbf{s}_1 \in A_\epsilon^{(n)}(S_1|\mathbf{s}_2)} 2^{-n(H(S_1|S_2)-2\epsilon)} \tag{15.44}$$

$$= \sum_{\mathbf{s}_2} p(\mathbf{s}_2)|A_\epsilon^{(n)}(S_1|\mathbf{s}_2)|2^{-n(H(S_1|S_2)-2\epsilon)}. \quad \square \tag{15.45}$$

To calculate the probability of decoding error, we need to know the probability that conditionally independent sequences are jointly typical. Let S_1, S_2, and S_3 be three subsets of $\{X^{(1)}, X^{(2)}, \ldots, X^{(k)}\}$. If S_1' and S_2' are conditionally independent given S_3' but otherwise share the same pairwise marginals of (S_1, S_2, S_3), we have the following probability of joint typicality.

Theorem 15.2.3 *Let $A_\epsilon^{(n)}$ denote the typical set for the probability mass function $p(s_1, s_2, s_3)$, and let*

$$P(\mathbf{S}_1' = \mathbf{s}_1, \mathbf{S}_2' = \mathbf{s}_2, \mathbf{S}_3' = \mathbf{s}_3) = \prod_{i=1}^{n} p(s_{1i}|s_{3i})p(s_{2i}|s_{3i})p(s_{3i}). \tag{15.46}$$

Then

$$P\{(\mathbf{S}_1', \mathbf{S}_2', \mathbf{S}_3') \in A_\epsilon^{(n)}\} \doteq 2^{n(I(S_1;S_2|S_3)\pm 6\epsilon)}. \tag{15.47}$$

Proof: We use the $\doteq$ notation from (15.26) to avoid calculating the upper and lower bounds separately. We have

$$P\{(\mathbf{S}_1', \mathbf{S}_2', \mathbf{S}_3') \in A_\epsilon^{(n)}\}$$

$$= \sum_{(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3) \in A_\epsilon^{(n)}} p(\mathbf{s}_3)p(\mathbf{s}_1|\mathbf{s}_3)p(\mathbf{s}_2|\mathbf{s}_3) \tag{15.48}$$

$$\doteq |A_\epsilon^{(n)}(S_1, S_2, S_3)|2^{-n(H(S_3)\pm\epsilon)}2^{-n(H(S_1|S_3)\pm 2\epsilon)}2^{-n(H(S_2|S_3)\pm 2\epsilon)} \tag{15.49}$$

$$\doteq 2^{n(H(S_1,S_2,S_3)\pm\epsilon)}2^{-n(H(S_3)\pm\epsilon)}2^{-n(H(S_1|S_3)\pm 2\epsilon)}2^{-n(H(S_2|S_3)\pm 2\epsilon)} \tag{15.50}$$

$$\doteq 2^{-n(I(S_1;S_2|S_3)\pm 6\epsilon)}. \quad \square \tag{15.51}$$

We will specialize this theorem to particular choices of S_1, S_2, and S_3 for the various achievability proofs in this chapter.

15.3 MULTIPLE-ACCESS CHANNEL

The first channel that we examine in detail is the multiple-access channel, in which two (or more) senders send information to a common receiver. The channel is illustrated in Figure 15.7. A common example of this channel is a satellite receiver with many independent ground stations, or a set of cell phones communicating with a base station. We see that the senders must contend not only with the receiver noise but with interference from each other as well.

Definition A *discrete memoryless multiple-access channel* consists of three alphabets, $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{Y}$, and a probability transition matrix $p(y|x_1, x_2)$.

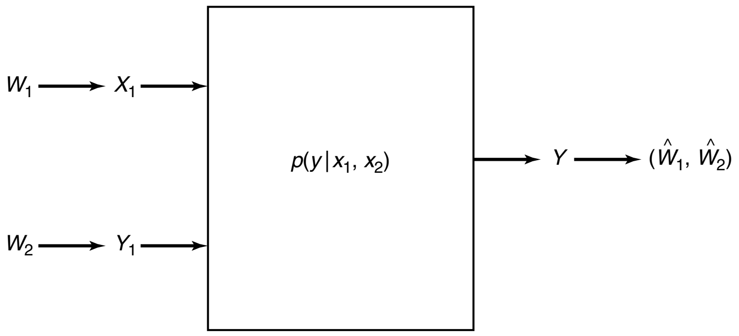


FIGURE 15.7. Multiple-access channel.

Definition A $((2^{nR_1}, 2^{nR_2}), n)$ code for the multiple-access channel consists of two sets of integers $\mathcal{W}_1 = \{1, 2, \ldots, 2^{nR_1}\}$ and $\mathcal{W}_2 = \{1, 2, \ldots, 2^{nR_2}\}$, called the *message sets*, two *encoding functions*,

$$X_1 : \mathcal{W}_1 \to \mathcal{X}_1^n \tag{15.52}$$

and

$$X_2 : \mathcal{W}_2 \to \mathcal{X}_2^n, \tag{15.53}$$

and a *decoding function*,

$$g : \mathcal{Y}^n \to \mathcal{W}_1 \times \mathcal{W}_2. \tag{15.54}$$

There are two senders and one receiver for this channel. Sender 1 chooses an index W_1 uniformly from the set $\{1, 2, \ldots, 2^{nR_1}\}$ and sends the corresponding codeword over the channel. Sender 2 does likewise. Assuming that the distribution of messages over the product set $\mathcal{W}_1 \times \mathcal{W}_2$ is uniform (i.e., the messages are independent and equally likely), we define the *average probability of error* for the $((2^{nR_1}, 2^{nR_2}), n)$ code as follows:

$$P_e^{(n)} = \frac{1}{2^{n(R_1+R_2)}} \sum_{(w_1,w_2)\in\mathcal{W}_1\times\mathcal{W}_2} \Pr\left\{g(Y^n) \neq (w_1, w_2)|(w_1, w_2) \text{ sent}\right\}. \tag{15.55}$$

Definition A rate pair (R_1, R_2) is said to be *achievable* for the multiple-access channel if there exists a sequence of $((2^{nR_1}, 2^{nR_2}), n)$ codes with $P_e^{(n)} \to 0$.

Definition The *capacity region* of the multiple-access channel is the closure of the set of achievable (R_1, R_2) rate pairs.

An example of the capacity region for a multiple-access channel is illustrated in Figure 15.8. We first state the capacity region in the form of a theorem.

Theorem 15.3.1 (*Multiple-access channel capacity*) *The capacity of a multiple-access channel* $(\mathcal{X}_1 \times \mathcal{X}_2, p(y|x_1, x_2), \mathcal{Y})$ *is the closure of the convex hull of all* (R_1, R_2) *satisfying*

$$R_1 < I(X_1; Y|X_2), \tag{15.56}$$

$$R_2 < I(X_2; Y|X_1), \tag{15.57}$$

$$R_1 + R_2 < I(X_1, X_2; Y) \tag{15.58}$$

for some product distribution $p_1(x_1)p_2(x_2)$ *on* $\mathcal{X}_1 \times \mathcal{X}_2$.

Before we prove that this is the capacity region of the multiple-access channel, let us consider a few examples of multiple-access channels:

Example 15.3.1 (*Independent binary symmetric channels*) Assume that we have two independent binary symmetric channels, one from sender 1 and the other from sender 2, as shown in Figure 15.9. In this case, it is obvious from the results of Chapter 7 that we can send at rate $1 - H(p_1)$ over the first channel and at rate $1 - H(p_2)$ over the second channel.

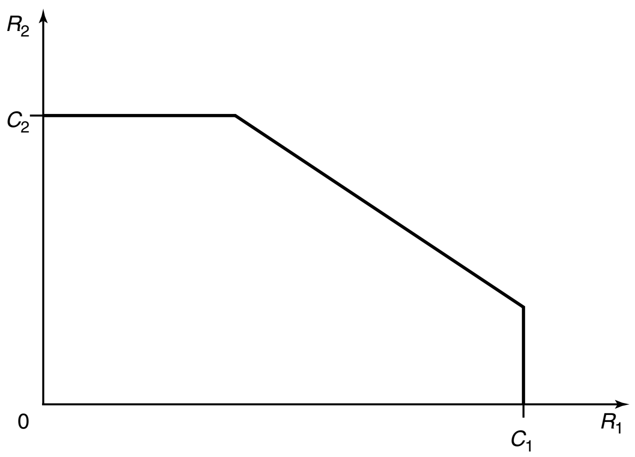


FIGURE 15.8. Capacity region for a multiple-access channel.

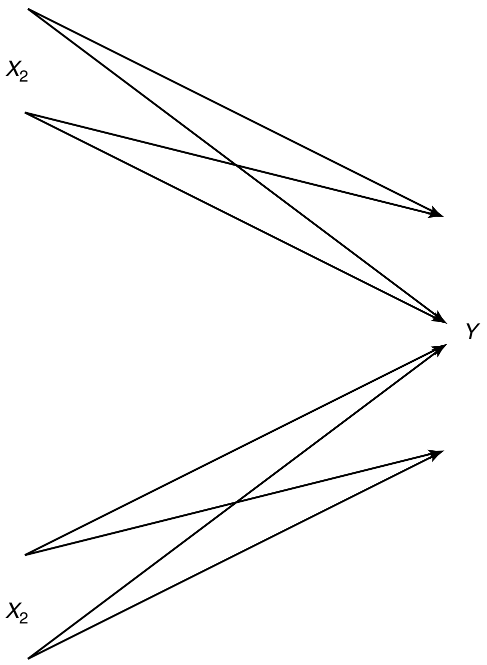


FIGURE 15.9. Independent binary symmetric channels.

Since the channels are independent, there is no interference between the senders. The capacity region in this case is shown in Figure 15.10.

Example 15.3.2 (*Binary multiplier channel*) Consider a multiple-access channel with binary inputs and output

$$Y = X_1 X_2. \tag{15.59}$$

Such a channel is called a *binary multiplier channel*. It is easy to see that by setting $X_2 = 1$, we can send at a rate of 1 bit per transmission from sender 1 to the receiver. Similarly, setting $X_1 = 1$, we can achieve $R_2 = 1$. Clearly, since the output is binary, the combined rates $R_1 + R_2$ of sender 1 and sender 2 cannot be more than 1 bit. By timesharing, we can achieve any combination of rates such that $R_1 + R_2 = 1$. Hence the capacity region is as shown in Figure 15.11.

Example 15.3.3 (*Binary erasure multiple-access channel*) This multiple-access channel has binary inputs, $\mathcal{X}_1 = \mathcal{X}_2 = \{0, 1\}$, and a ternary output, $Y = X_1 + X_2$. There is no ambiguity in (X_1, X_2) if $Y = 0$ or $Y = 2$ is received; but $Y = 1$ can result from either (0,1) or (1,0).

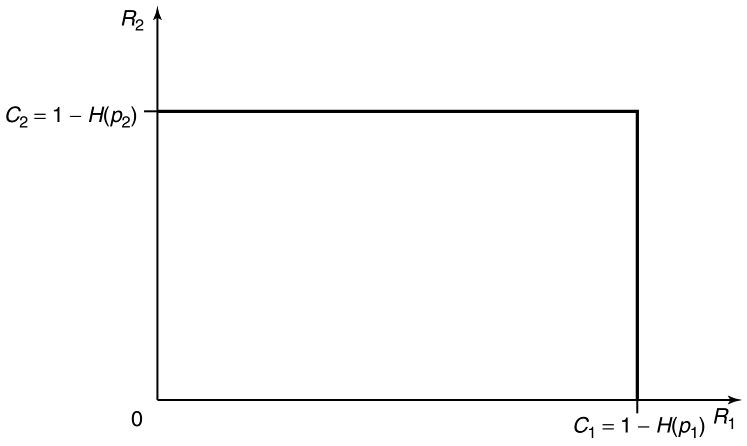


FIGURE 15.10. Capacity region for independent BSCs.

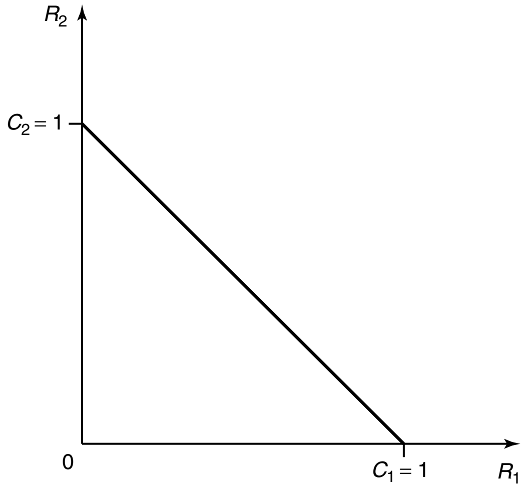


FIGURE 15.11. Capacity region for binary multiplier channel.

We now examine the achievable rates on the axes. Setting $X_2 = 0$, we can send at a rate of 1 bit per transmission from sender 1. Similarly, setting $X_1 = 0$, we can send at a rate $R_2 = 1$. This gives us two extreme points of the capacity region. Can we do better? Let us assume that $R_1 = 1$, so that the codewords of X_1 must include all possible binary sequences; X_1 would look like a Bernoulli($\frac{1}{2}$) process. This acts like noise for the transmission from X_2. For X_2, the channel looks like the channel in Figure 15.12. This is the binary erasure channel of Chapter 7. Recalling the results, the

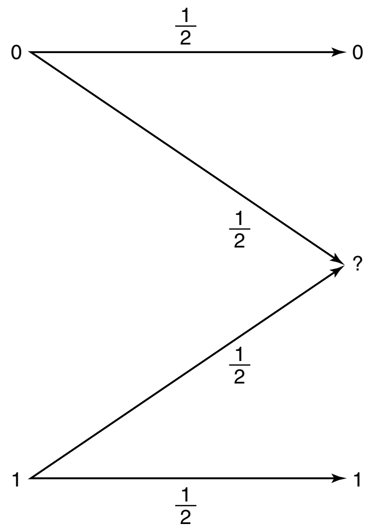


FIGURE 15.12. Equivalent single-user channel for user 2 of a binary erasure multiple-access channel.

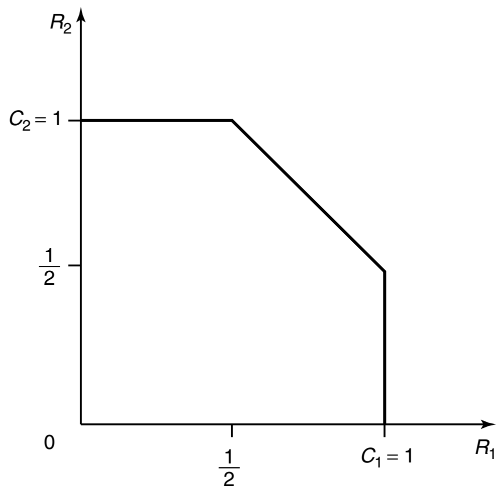


FIGURE 15.13. Capacity region for binary erasure multiple-access channel.

capacity of this channel is $\frac{1}{2}$ bit per transmission. Hence when sending at maximum rate 1 for sender 1, we can send an additional $\frac{1}{2}$ bit from sender 2. Later, after deriving the capacity region, we can verify that these rates are the best that can be achieved. The capacity region for a binary erasure channel is illustrated in Figure 15.13.

15.3.1 Achievability of the Capacity Region for the Multiple-Access Channel

We now prove the achievability of the rate region in Theorem 15.3.1; the proof of the converse will be left until the next section. The proof of achievability is very similar to the proof for the single-user channel. We therefore only emphasize the points at which the proof differs from the single-user case. We begin by proving the achievability of rate pairs that satisfy (15.58) for some fixed product distribution $p(x_1)p(x_2)$. In Section 15.3.3 we extend this to prove that all points in the convex hull of (15.58) are achievable.

Proof: (*Achievability in Theorem 15.3.1*). Fix $p(x_1, x_2) = p_1(x_1)p_2(x_2)$.

Codebook generation: Generate 2^{nR_1} independent codewords $\mathbf{X}_1(i)$, $i \in \{1, 2, \ldots, 2^{nR_1}\}$, of length n, generating each element i.i.d. $\sim \prod_{i=1}^n p_1(x_{1i})$. Similarly, generate 2^{nR_2} independent codewords $\mathbf{X}_2(j)$, $j \in \{1, 2, \ldots, 2^{nR_2}\}$, generating each element i.i.d. $\sim \prod_{i=1}^n p_2(x_{2i})$. These codewords form the codebook, which is revealed to the senders and the receiver.

Encoding: To send index i, sender 1 sends the codeword $\mathbf{X}_1(i)$. Similarly, to send j, sender 2 sends $\mathbf{X}_2(j)$.

Decoding: Let $A_\epsilon^{(n)}$ denote the set of typical $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y})$ sequences. The receiver Y^n chooses the pair (i, j) such that

$$(\mathbf{x}_1(i), \mathbf{x}_2(j), \mathbf{y}) \in A_\epsilon^{(n)} \tag{15.60}$$

if such a pair (i, j) exists and is unique; otherwise, an error is declared.

Analysis of the probability of error: By the symmetry of the random code construction, the conditional probability of error does not depend on which pair of indices is sent. Thus, the conditional probability of error is the same as the unconditional probability of error. So, without loss of generality, we can assume that $(i, j) = (1, 1)$ was sent.

We have an error if either the correct codewords are not typical with the received sequence or there is a pair of incorrect codewords that are typical with the received sequence. Define the events

$$E_{ij} = \{(\mathbf{X}_1(i), \mathbf{X}_2(j), \mathbf{Y}) \in A_\epsilon^{(n)}\}. \tag{15.61}$$

Then by the union of events bound,

$$P_e^{(n)} = P\left(E_{11}^c \bigcup \cup_{(i,j)\neq(1,1)} E_{ij}\right) \tag{15.62}$$

$$\le P(E_{11}^c) + \sum_{i\neq 1,\, j=1} P(E_{i1}) + \sum_{i=1,\, j\neq 1} P(E_{1j}) + \sum_{i\neq 1,\, j\neq 1} P(E_{ij}), \tag{15.63}$$

where P is the conditional probability given that (1, 1) was sent. From the AEP, $P(E_{11}^c) \to 0$. By Theorems 15.2.1 and 15.2.3, for $i \neq 1$, we have

$$P(E_{i1}) = P((\mathbf{X}_1(i), \mathbf{X}_2(1), \mathbf{Y}) \in A_\epsilon^{(n)}) \tag{15.64}$$

$$= \sum_{(\mathbf{x}_1,\mathbf{x}_2,\mathbf{y})\in A_\epsilon^{(n)}} p(\mathbf{x}_1)p(\mathbf{x}_2, \mathbf{y}) \tag{15.65}$$

$$\le |A_\epsilon^{(n)}| 2^{-n(H(X_1)-\epsilon)} 2^{-n(H(X_2,Y)-\epsilon)} \tag{15.66}$$

$$\le 2^{-n(H(X_1)+H(X_2,Y)-H(X_1,X_2,Y)-3\epsilon)} \tag{15.67}$$

$$= 2^{-n(I(X_1;X_2,Y)-3\epsilon)} \tag{15.68}$$

$$= 2^{-n(I(X_1;Y|X_2)-3\epsilon)}, \tag{15.69}$$

where the equivalence of (15.68) and (15.69) follows from the independence of X_1 and X_2, and the consequent $I(X_1; X_2, Y) = I(X_1; X_2) + I(X_1; Y|X_2) = I(X_1; Y|X_2)$. Similarly, for $j \neq 1$,

$$P(E_{1j}) \le 2^{-n(I(X_2;Y|X_1)-3\epsilon)}, \tag{15.70}$$

and for $i \neq 1$, $j \neq 1$,

$$P(E_{ij}) \le 2^{-n(I(X_1,X_2;Y)-4\epsilon)}. \tag{15.71}$$

It follows that

$$P_e^{(n)} \le P(E_{11}^c) + 2^{nR_1} 2^{-n(I(X_1;Y|X_2)-3\epsilon)} + 2^{nR_2} 2^{-n(I(X_2;Y|X_1)-3\epsilon)} + 2^{n(R_1+R_2)} 2^{-n(I(X_1,X_2;Y)-4\epsilon)}. \tag{15.72}$$

Since $\epsilon > 0$ is arbitrary, the conditions of the theorem imply that each term tends to 0 as $n \to \infty$. Thus, the probability of error, conditioned on a particular codeword being sent, goes to zero if the conditions of the theorem are met. The above bound shows that the average probability of error, which by symmetry is equal to the probability for an individual codeword, averaged over all choices of codebooks in the random code construction, is arbitrarily small. Hence, there exists at least one code $\mathcal{C}^*$ with arbitrarily small probability of error.

This completes the proof of achievability of the region in (15.58) for a fixed input distribution. Later, in Section 15.3.3, we show that time-sharing allows any (R_1, R_2) in the convex hull to be achieved, completing the proof of the forward part of the theorem. □

15.3.2 Comments on the Capacity Region for the Multiple-Access Channel

We have now proved the achievability of the capacity region of the multiple-access channel, which is the closure of the convex hull of the set of points (R_1, R_2) satisfying

$$R_1 < I(X_1; Y|X_2), \tag{15.73}$$

$$R_2 < I(X_2; Y|X_1), \tag{15.74}$$

$$R_1 + R_2 < I(X_1, X_2; Y) \tag{15.75}$$

for some distribution $p_1(x_1)p_2(x_2)$ on $\mathcal{X}_1 \times \mathcal{X}_2$. For a particular $p_1(x_1)p_2(x_2)$, the region is illustrated in Figure 15.14.

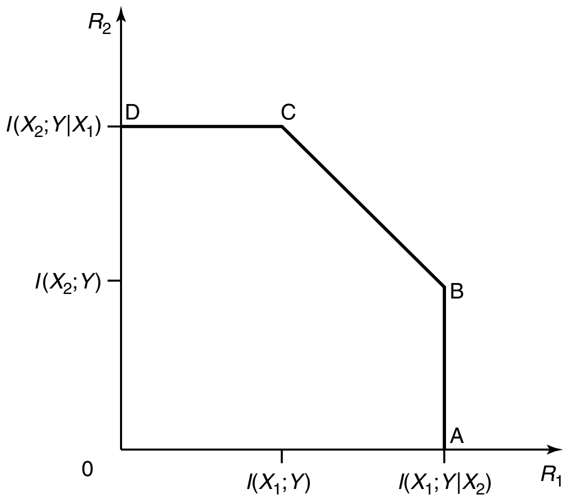


FIGURE 15.14. Achievable region of multiple-access channel for a fixed input distribution.

Let us now interpret the corner points in the region. Point A corresponds to the maximum rate achievable from sender 1 to the receiver when sender 2 is not sending any information. This is

$$\max R_1 = \max_{p_1(x_1)p_2(x_2)} I(X_1; Y|X_2). \tag{15.76}$$

Now for any distribution $p_1(x_1)p_2(x_2)$,

$$I(X_1; Y|X_2) = \sum_{x_2} p_2(x_2) I(X_1; Y|X_2 = x_2) \tag{15.77}$$

$$\leq \max_{x_2} I(X_1; Y|X_2 = x_2), \tag{15.78}$$

since the average is less than the maximum. Therefore, the maximum in (15.76) is attained when we set $X_2 = x_2$, where x_2 is the value that maximizes the conditional mutual information between X_1 and Y. The distribution of X_1 is chosen to maximize this mutual information. Thus, X_2 must facilitate the transmission of X_1 by setting $X_2 = x_2$.

The point B corresponds to the maximum rate at which sender 2 can send as long as sender 1 sends at his maximum rate. This is the rate that is obtained if X_1 is considered as noise for the channel from X_2 to Y. In this case, using the results from single-user channels, X_2 can send at a rate $I(X_2; Y)$. The receiver now knows which X_2 codeword was used and can "subtract" its effect from the channel. We can consider the channel now to be an indexed set of single-user channels, where the index is the X_2 symbol used. The X_1 rate achieved in this case is the average mutual information, where the average is over these channels, and each channel occurs as many times as the corresponding X_2 symbol appears in the codewords. Hence, the rate achieved is

$$\sum_{x_2} p(x_2) I(X_1; Y|X_2 = x_2) = I(X_1; Y|X_2). \tag{15.79}$$

Points C and D correspond to B and A, respectively, with the roles of the senders reversed. The noncorner points can be achieved by time-sharing. Thus, we have given a single-user interpretation and justification for the capacity region of a multiple-access channel.

The idea of considering other signals as part of the noise, decoding one signal, and then "subtracting" it from the received signal is a very useful one. We will come across the same concept again in the capacity calculations for the degraded broadcast channel.

15.3.3 Convexity of the Capacity Region of the Multiple-Access Channel

We now recast the capacity region of the multiple-access channel in order to take into account the operation of taking the convex hull by introducing a new random variable. We begin by proving that the capacity region is convex.

Theorem 15.3.2 *The capacity region $\mathcal{C}$ of a multiple-access channel is convex [i.e., if $(R_1, R_2) \in \mathcal{C}$ and $(R_1', R_2') \in \mathcal{C}$, then $(\lambda R_1 + (1-\lambda)R_1', \lambda R_2 + (1-\lambda)R_2') \in \mathcal{C}$ for $0 \le \lambda \le 1$].*

Proof: The idea is time-sharing. Given two sequences of codes at different rates $\mathbf{R} = (R_1, R_2)$ and $\mathbf{R}' = (R_1', R_2')$, we can construct a third codebook at a rate $\lambda\mathbf{R} + (1-\lambda)\mathbf{R}'$ by using the first codebook for the first λn symbols and using the second codebook for the last $(1-\lambda)n$ symbols. The number of X_1 codewords in the new code is

$$2^{n\lambda R_1} 2^{n(1-\lambda)R_1'} = 2^{n(\lambda R_1 + (1-\lambda)R_1')}, \tag{15.80}$$

and hence the rate of the new code is $\lambda\mathbf{R} + (1-\lambda)\mathbf{R}'$. Since the overall probability of error is less than the sum of the probabilities of error for each of the segments, the probability of error of the new code goes to 0 and the rate is achievable. □

We can now recast the statement of the capacity region for the multiple-access channel using a time-sharing random variable Q. Before we prove this result, we need to prove a property of convex sets defined by linear inequalities like those of the capacity region of the multiple-access channel. In particular, we would like to show that the convex hull of two such regions defined by linear constraints is the region defined by the convex combination of the constraints. Initially, the equality of these two sets seems obvious, but on closer examination, there is a subtle difficulty due to the fact that some of the constraints might not be active. This is best illustrated by an example. Consider the following two sets defined by linear inequalities:

$$C_1 = \{(x, y) : x \ge 0, y \ge 0, x \le 10, y \le 10, x + y \le 100\} \tag{15.81}$$

$$C_2 = \{(x, y) : x \ge 0, y \ge 0, x \le 20, y \le 20, x + y \le 20\}. \tag{15.82}$$

In this case, the $(\frac{1}{2}, \frac{1}{2})$ convex combination of the constraints defines the region

$$C = \{(x, y) : x \ge 0, y \ge 0, x \le 15, y \le 15, x + y \le 60\}. \tag{15.83}$$

It is not difficult to see that any point in C_1 or C_2 has $x + y < 20$, so any point in the convex hull of the union of C_1 and C_2 satisfies this property. Thus, the point (15,15), which is in C, is not in the convex hull of $(C_1 \cup C_2)$. This example also hints at the cause of the problem—in the definition for C_1, the constraint $x + y \le 100$ is not active. If this constraint were replaced by a constraint $x + y \le a$, where $a \le 20$, the above result of the equality of the two regions would be true, as we now prove.

We restrict ourselves to the pentagonal regions that occur as components of the capacity region of a two-user multiple-access channel. In this case, the capacity region for a fixed $p(x_1)p(x_2)$ is defined by three mutual informations, $I(X_1; Y|X_2)$, $I(X_2; Y|X_1)$, and $I(X_1, X_2; Y)$, which we shall call I_1, I_2, and I_3, respectively. For each $p(x_1)p(x_2)$, there is a corresponding vector, $\mathbf{I} = (I_1, I_2, I_3)$, and a rate region defined by

$$C_{\mathbf{I}} = \{(R_1, R_2) : R_1 \ge 0, R_2 \ge 0, R_1 \le I_1, R_2 \le I_2, R_1 + R_2 \le I_3\}. \tag{15.84}$$

Also, since for any distribution $p(x_1)p(x_2)$, we have $I(X_2; Y|X_1) = H(X_2|X_1) - H(X_2|Y, X_1) = H(X_2) - H(X_2|Y, X_1) = I(X_2; Y, X_1) = I(X_2; Y) + I(X_2; X_1|Y) \ge I(X_2; Y)$, and therefore, $I(X_1; Y|X_2) + I(X_2; Y|X_1) \ge I(X_1; Y|X_2) + I(X_2; Y) = I(X_1, X_2; Y)$, we have for all vectors I that $I_1 + I_2 \ge I_3$. This property will turn out to be critical for the theorem.

Lemma 15.3.1 *Let $\mathbf{I}_1, \mathbf{I}_2 \in \mathcal{R}^3$ be two vectors of mutual informations that define rate regions $C_{\mathbf{I}_1}$ and $C_{\mathbf{I}_2}$, respectively, as given in (15.84). For $0 \le \lambda \le 1$, define $\mathbf{I}_\lambda = \lambda \mathbf{I}_1 + (1 - \lambda)\mathbf{I}_2$, and let $C_{\mathbf{I}_\lambda}$ be the rate region defined by $\mathbf{I}_\lambda$. Then*

$$C_{\mathbf{I}_\lambda} = \lambda C_{\mathbf{I}_1} + (1 - \lambda) C_{\mathbf{I}_2}. \tag{15.85}$$

Proof: We shall prove this theorem in two parts. We first show that any point in the $(\lambda, 1 - \lambda)$ mix of the sets $C_{\mathbf{I}_1}$ and $C_{\mathbf{I}_2}$ satisfies the constraints $\mathbf{I}_\lambda$. But this is straightforward, since any point in $C_{\mathbf{I}_1}$ satisfies the inequalities for $\mathbf{I}_1$ and a point in $C_{\mathbf{I}_2}$ satisfies the inequalities for $\mathbf{I}_2$, so the $(\lambda, 1 - \lambda)$ mix of these points will satisfy the $(\lambda, 1 - \lambda)$ mix of the constraints. Thus, it follows that

$$\lambda C_{\mathbf{I}_1} + (1 - \lambda) C_{\mathbf{I}_2} \subseteq C_{\mathbf{I}_\lambda}. \tag{15.86}$$

To prove the reverse inclusion, we consider the extreme points of the pentagonal regions. It is not difficult to see that the rate regions defined in (15.84) are always in the form of a pentagon, or in the extreme case when $I_3 = I_1 + I_2$, in the form of a rectangle. Thus, the capacity region $C_{\mathbf{I}}$ can be also defined as a convex hull of five points:

$$(0, 0), (I_1, 0), (I_1, I_3 - I_1), (I_3 - I_2, I_2), (0, I_2). \tag{15.87}$$

Consider the region defined by $\mathbf{I}_\lambda$; it, too, is defined by five points. Take any one of the points, say $(I_3^{(\lambda)} - I_2^{(\lambda)}, I_2^{(\lambda)})$. This point can be written as the $(\lambda, 1-\lambda)$ mix of the points $(I_3^{(1)} - I_2^{(1)}, I_2^{(1)})$ and $(I_3^{(2)} - I_2^{(2)}, I_2^{(2)})$, and therefore lies in the convex mixture of $C_{\mathbf{I}_1}$ and $C_{\mathbf{I}_2}$. Thus, all extreme points of the pentagon $C_{\mathbf{I}_\lambda}$ lie in the convex hull of $C_{\mathbf{I}_1}$ and $C_{\mathbf{I}_2}$, or

$$C_{\mathbf{I}_\lambda} \subseteq \lambda C_{\mathbf{I}_1} + (1-\lambda) C_{\mathbf{I}_2}. \tag{15.88}$$

Combining the two parts, we have the theorem. □

In the proof of the theorem, we have implicitly used the fact that all the rate regions are defined by five extreme points (at worst, some of the points are equal). All five points defined by the $\mathbf{I}$ vector were within the rate region. If the condition $I_3 \le I_1 + I_2$ is not satisfied, some of the points in (15.87) may be outside the rate region and the proof collapses.

As an immediate consequence of the above lemma, we have the following theorem:

Theorem 15.3.3 *The convex hull of the union of the rate regions defined by individual* **I** *vectors is equal to the rate region defined by the convex hull of the* **I** *vectors.*

These arguments on the equivalence of the convex hull operation on the rate regions with the convex combinations of the mutual informations can be extended to the general m-user multiple-access channel. A proof along these lines using the theory of polymatroids is developed in Han [271].

Theorem 15.3.4 *The set of achievable rates of a discrete memoryless multiple-access channel is given by the closure of the set of all* (R_1, R_2) *pairs satisfying*

$$\begin{aligned} R_1 &< I(X_1; Y | X_2, Q), \\ R_2 &< I(X_2; Y | X_1, Q), \\ R_1 + R_2 &< I(X_1, X_2; Y | Q) \end{aligned} \tag{15.89}$$

for some choice of the joint distribution $p(q)p(x_1|q)p(x_2|q)p(y|x_1, x_2)$ *with* $|\mathcal{Q}| \leq 4$.

Proof: We will show that every rate pair lying in the region defined in (15.89) is achievable (i.e., it lies in the convex closure of the rate pairs satisfying Theorem 15.3.1). We also show that every point in the convex closure of the region in Theorem 15.3.1 is also in the region defined in (15.89).

Consider a rate point $\mathbf{R}$ satisfying the inequalities (15.89) of the theorem. We can rewrite the right-hand side of the first inequality as

$$I(X_1; Y|X_2, Q) = \sum_{q=1}^{m} p(q) I(X_1; Y|X_2, Q = q) \tag{15.90}$$

$$= \sum_{q=1}^{m} p(q) I(X_1; Y|X_2)_{p_{1q}, p_{2q}}, \tag{15.91}$$

where m is the cardinality of the support set of Q. We can expand the other mutual informations similarly.

For simplicity in notation, we consider a rate pair as a vector and denote a pair satisfying the inequalities in (15.58) for a specific input product distribution $p_{1q}(x_1)p_{2q}(x_2)$ as $\mathbf{R}_{p_1,p_2}$ as $\mathbf{R}_q$. Specifically, let $\mathbf{R}_q = (R_{1q}, R_{2q})$ be a rate pair satisfying

$$R_{1q} < I(X_1; Y|X_2)_{p_{1q}(x_1)p_{2q}(x_2)}, \tag{15.92}$$

$$R_{2q} < I(X_2; Y|X_1)_{p_{1q}(x_1)p_{2q}(x_2)}, \tag{15.93}$$

$$R_{1q} + R_{2q} < I(X_1, X_2; Y)_{p_{1q}(x_1)p_{2q}(x_2)}. \tag{15.94}$$

Then by Theorem 15.3.1, $\mathbf{R}_q = (R_{1q}, R_{2q})$ is achievable. Then since $\mathbf{R}$ satisfies (15.89) and we can expand the right-hand sides as in (15.91), there exists a setof pairs $\mathbf{R}_q$ satisfying (15.94) such that

$$\mathbf{R} = \sum_{q=1}^{m} p(q)\mathbf{R}_q. \tag{15.95}$$

Since a convex combination of achievable rates is achievable, so is $\mathbf{R}$. Hence, we have proven the achievability of the region in the theorem. The same argument can be used to show that every point in the convex closure of the region in (15.58) can be written as the mixture of points satisfying (15.94) and hence can be written in the form (15.89).

The converse is proved in the next section. The converse shows that all achievable rate pairs are of the form (15.89), and hence establishes that this is the capacity region of the multiple-access channel. The cardinality bound on the time-sharing random variable Q is a consequence of Carathéodory's theorem on convex sets. See the discussion below. □

The proof of the convexity of the capacity region shows that any convex combination of achievable rate pairs is also achievable. We can continue this process, taking convex combinations of more points. Do we need to use an arbitrary number of points ? Will the capacity region be increased? The following theorem says no.

Theorem 15.3.5 (*Carathéodory*) *Any point in the convex closure of a compact set A in a d-dimensional Euclidean space can be represented as a convex combination of $d + 1$ or fewer points in the original set A.*

Proof: The proof may be found in Eggleston [183] and Grünbaum [263]. □

This theorem allows us to restrict attention to a certain finite convex combination when calculating the capacity region. This is an important property because without it, we would not be able to compute the capacity region in (15.89), since we would never know whether using a larger alphabet $\mathcal{Q}$ would increase the region.

In the multiple-access channel, the bounds define a connected compact set in three dimensions. Therefore, all points in its closure can be defined as the convex combination of at most four points. Hence, we can restrict the cardinality of Q to at most 4 in the above definition of the capacity region.

Remark Many of the cardinality bounds may be slightly improved by introducing other considerations. For example, if we are only interested in the boundary of the convex hull of A as we are in capacity theorems, a point on the boundary can be expressed as a mixture of d points of A, since a point on the boundary lies in the intersection of A with a $(d - 1)$-dimensional support hyperplane.

15.3.4 Converse for the Multiple-Access Channel

We have so far proved the achievability of the capacity region. In this section we prove the converse.

Proof: (*Converse to Theorems 15.3.1 and 15.3.4*). We must show that given any sequence of $((2^{nR_1}, 2^{nR_2}), n)$ codes with $P_e^{(n)} \to 0$, the rates must satisfy

$$\begin{aligned} R_1 &\le I(X_1; Y|X_2, Q), \\ R_2 &\le I(X_2; Y|X_1, Q), \\ R_1 + R_2 &\le I(X_1, X_2; Y|Q) \end{aligned} \tag{15.96}$$

for some choice of random variable Q defined on $\{1, 2, 3, 4\}$ and joint distribution $p(q)p(x_1|q)p(x_2|q)p(y|x_1, x_2)$. Fix n. Consider the given code of block length n. The joint distribution on $\mathcal{W}_1 \times \mathcal{W}_2 \times \mathcal{X}_1^n \times \mathcal{X}_2^n \times \mathcal{Y}^n$ is well defined. The only randomness is due to the random uniform choice of indices W_1 and W_2 and the randomness induced by the channel. The joint distribution is

$$p(w_1, w_2, x_1^n, x_2^n, y^n) = \frac{1}{2^{nR_1}} \frac{1}{2^{nR_2}} p(x_1^n|w_1) p(x_2^n|w_2) \prod_{i=1}^{n} p(y_i|x_{1i}, x_{2i}), \tag{15.97}$$

where $p(x_1^n|w_1)$ is either 1 or 0, depending on whether $x_1^n = \mathbf{x}_1(w_1)$, the codeword corresponding to w_1, or not, and similarly, $p(x_2^n|w_2) = 1$ or 0, according to whether $x_2^n = \mathbf{x}_2(w_2)$ or not. The mutual informations that follow are calculated with respect to this distribution.

By the code construction, it is possible to estimate (W_1, W_2) from the received sequence Y^n with a low probability of error. Hence, the conditional entropy of (W_1, W_2) given Y^n must be small. By Fano's inequality,

$$H(W_1, W_2|Y^n) \le n(R_1 + R_2)P_e^{(n)} + H(P_e^{(n)}) \overset{\triangle}{=} n\epsilon_n. \tag{15.98}$$

It is clear that $\epsilon_n \to 0$ as $P_e^{(n)} \to 0$. Then we have

$$H(W_1|Y^n) \le H(W_1, W_2|Y^n) \le n\epsilon_n, \tag{15.99}$$

$$H(W_2|Y^n) \le H(W_1, W_2|Y^n) \le n\epsilon_n. \tag{15.100}$$

We can now bound the rate R_1 as

$$nR_1 = H(W_1) \tag{15.101}$$

$$= I(W_1; Y^n) + H(W_1|Y^n) \tag{15.102}$$

$$\overset{(a)}{\le} I(W_1; Y^n) + n\epsilon_n \tag{15.103}$$

$$\overset{(b)}{\leq} I(X_1^n(W_1); Y^n) + n\epsilon_n \tag{15.104}$$

$$= H(X_1^n(W_1)) - H(X_1^n(W_1)|Y^n) + n\epsilon_n \tag{15.105}$$

$$\overset{(c)}{\leq} H(X_1^n(W_1)|X_2^n(W_2)) - H(X_1^n(W_1)|Y^n, X_2^n(W_2)) + n\epsilon_n \tag{15.106}$$

$$= I(X_1^n(W_1); Y^n|X_2^n(W_2)) + n\epsilon_n \tag{15.107}$$

$$= H(Y^n|X_2^n(W_2)) - H(Y^n|X_1^n(W_1), X_2^n(W_2)) + n\epsilon_n \tag{15.108}$$

$$\overset{(d)}{=} H(Y^n|X_2^n(W_2)) - \sum_{i=1}^{n} H(Y_i|Y^{i-1}, X_1^n(W_1), X_2^n(W_2)) + n\epsilon_n \tag{15.109}$$

$$\overset{(e)}{=} H(Y^n|X_2^n(W_2)) - \sum_{i=1}^{n} H(Y_i|X_{1i}, X_{2i}) + n\epsilon_n \tag{15.110}$$

$$\overset{(f)}{\leq} \sum_{i=1}^{n} H(Y_i|X_2^n(W_2)) - \sum_{i=1}^{n} H(Y_i|X_{1i}, X_{2i}) + n\epsilon_n \tag{15.111}$$

$$\overset{(g)}{\leq} \sum_{i=1}^{n} H(Y_i|X_{2i}) - \sum_{i=1}^{n} H(Y_i|X_{1i}, X_{2i}) + n\epsilon_n \tag{15.112}$$

$$= \sum_{i=1}^{n} I(X_{1i}; Y_i|X_{2i}) + n\epsilon_n, \tag{15.113}$$

where

(a) follows from Fano's inequality
(b) follows from the data-processing inequality
(c) follows from the fact that since W_1 and W_2 are independent, so are $X_1^n(W_1)$ and $X_2^n(W_2)$, and hence $H(X_1^n(W_1)|X_2^n(W_2)) = H(X_1^n(W_1))$, and $H(X_1^n(W_1)|Y^n, X_2^n(W_2)) \leq H(X_1^n(W_1)|Y^n)$ by conditioning
(d) follows from the chain rule
(e) follows from the fact that Y_i depends only on X_{1i} and X_{2i} by the memoryless property of the channel
(f) follows from the chain rule and removing conditioning
(g) follows from removing conditioning

Hence, we have

$$R_1 \leq \frac{1}{n} \sum_{i=1}^{n} I(X_{1i}; Y_i|X_{2i}) + \epsilon_n. \tag{15.114}$$

Similarly, we have

$$R_2 \le \frac{1}{n}\sum_{i=1}^{n} I(X_{2i}; Y_i|X_{1i}) + \epsilon_n. \tag{15.115}$$

To bound the sum of the rates, we have

$$\begin{aligned}
n(R_1 + R_2) &= H(W_1, W_2) && (15.116)\\
&= I(W_1, W_2; Y^n) + H(W_1, W_2|Y^n) && (15.117)\\
&\overset{(a)}{\le} I(W_1, W_2; Y^n) + n\epsilon_n && (15.118)\\
&\overset{(b)}{\le} I(X_1^n(W_1), X_2^n(W_2); Y^n) + n\epsilon_n && (15.119)\\
&= H(Y^n) - H(Y^n|X_1^n(W_1), X_2^n(W_2)) + n\epsilon_n && (15.120)\\
&\overset{(c)}{=} H(Y^n) - \sum_{i=1}^{n} H(Y_i|Y^{i-1}, X_1^n(W_1), X_2^n(W_2)) + n\epsilon_n && (15.121)\\
&\overset{(d)}{=} H(Y^n) - \sum_{i=1}^{n} H(Y_i|X_{1i}, X_{2i}) + n\epsilon_n && (15.122)\\
&\overset{(e)}{\le} \sum_{i=1}^{n} H(Y_i) - \sum_{i=1}^{n} H(Y_i|X_{1i}, X_{2i}) + n\epsilon_n && (15.123)\\
&= \sum_{i=1}^{n} I(X_{1i}, X_{2i}; Y_i) + n\epsilon_n, && (15.124)
\end{aligned}$$

where

(a) follows from Fano's inequality
(b) follows from the data-processing inequality
(c) follows from the chain rule
(d) follows from the fact that Y_i depends only on X_{1i} and X_{2i} and is conditionally independent of everything else
(e) follows from the chain rule and removing conditioning

Hence, we have

$$R_1 + R_2 \le \frac{1}{n}\sum_{i=1}^{n} I(X_{1i}, X_{2i}; Y_i) + \epsilon_n. \tag{15.125}$$

The expressions in (15.114), (15.115), and (15.125) are the averages of the mutual informations calculated at the empirical distributions in column i of the codebook. We can rewrite these equations with the new variable Q, where $Q = i \in \{1, 2, \ldots, n\}$ with probability $\frac{1}{n}$. The equations become

$$\begin{aligned}
R_1 &\le \frac{1}{n}\sum_{i=1}^{n} I(X_{1i}; Y_i|X_{2i}) + \epsilon_n && (15.126)\\
&= \frac{1}{n}\sum_{i=1}^{n} I(X_{1q}; Y_q|X_{2q}, Q = i) + \epsilon_n && (15.127)\\
&= I(X_{1Q}; Y_Q|X_{2Q}, Q) + \epsilon_n && (15.128)\\
&= I(X_1; Y|X_2, Q) + \epsilon_n, && (15.129)
\end{aligned}$$

where $X_1 \overset{\triangle}{=} X_{1Q}$, $X_2 \overset{\triangle}{=} X_{2Q}$, and $Y \overset{\triangle}{=} Y_Q$ are new random variables whose distributions depend on Q in the same way as the distributions of X_{1i}, X_{2i} and Y_i depend on i. Since W_1 and W_2 are independent, so are $X_{1i}(W_1)$ and $X_{2i}(W_2)$, and hence

$$\begin{aligned}
&\Pr\left(X_{1i}(W_1) = x_1, X_{2i}(W_2) = x_2\right)\\
&\quad \overset{\triangle}{=} \Pr\{X_{1Q} = x_1|Q = i\}\Pr\{X_{2Q} = x_2|Q = i\}. && (15.130)
\end{aligned}$$

Hence, taking the limit as $n \to \infty$, $P_e^{(n)} \to 0$, we have the following converse:

$$\begin{aligned}
R_1 &\le I(X_1; Y|X_2, Q),\\
R_2 &\le I(X_2; Y|X_1, Q),\\
R_1 + R_2 &\le I(X_1, X_2; Y|Q) && (15.131)
\end{aligned}$$

for some choice of joint distribution $p(q)p(x_1|q)p(x_2|q)p(y|x_1, x_2)$. As in Section 15.3.3, the region is unchanged if we limit the cardinality of $\mathcal{Q}$ to 4.

This completes the proof of the converse. □

Thus, the achievability of the region of Theorem 15.3.1 was proved in Section 15.3.1. In Section 15.3.3 we showed that every point in the region defined by (15.96) was also achievable. In the converse, we showed that the region in (15.96) was the best we can do, establishing that this is indeed the capacity region of the channel. Thus, the region in (15.58) cannot be any larger than the region in (15.96), and this is the capacity region of the multiple-access channel.

15.3.5 *m*-User Multiple-Access Channels

We will now generalize the result derived for two senders to m senders, $m \geq 2$. The multiple-access channel in this case is shown in Figure 15.15.

We send independent indices $w_1, w_2, \ldots, w_m$ over the channel from the senders $1, 2, \ldots, m$, respectively. The codes, rates, and achievability are all defined in exactly the same way as in the two-sender case.

Let $S \subseteq \{1, 2, \ldots, m\}$. Let S^c denote the complement of S. Let $R(S) = \sum_{i \in S} R_i$, and let $X(S) = \{X_i : i \in S\}$. Then we have the following theorem.

Theorem 15.3.6 *The capacity region of the m-user multiple-access channel is the closure of the convex hull of the rate vectors satisfying*

$$R(S) \leq I(X(S); Y|X(S^c)) \quad \textit{for all } S \subseteq \{1, 2, \ldots, m\} \tag{15.132}$$

for some product distribution $p_1(x_1)p_2(x_2)\cdots p_m(x_m)$.

Proof: The proof contains no new ideas. There are now $2^m - 1$ terms in the probability of error in the achievability proof and an equal number of inequalities in the proof of the converse. Details are left to the reader. □

In general, the region in (15.132) is a beveled box.

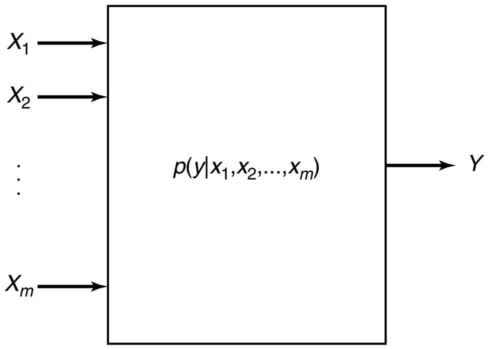


FIGURE 15.15. *m*-user multiple-access channel.

15.3.6 Gaussian Multiple-Access Channels

We now discuss the Gaussian multiple-access channel of Section 15.1.2 in somewhat more detail.

Two senders, X_1 and X_2, communicate to the single receiver, Y. The received signal at time i is

$$Y_i = X_{1i} + X_{2i} + Z_i, \tag{15.133}$$

where $\{Z_i\}$ is a sequence of independent, identically distributed, zero-mean Gaussian random variables with variance N (Figure 15.16). We assume that there is a power constraint P_j on sender j; that is, for each sender, for all messages, we must have

$$\frac{1}{n}\sum_{i=1}^{n} x_{ji}^2(w_j) \le P_j, \qquad w_j \in \{1, 2, \ldots, 2^{nR_j}\}, \qquad j = 1, 2. \tag{15.134}$$

Just as the proof of achievability of channel capacity for the discrete case (Chapter 7) was extended to the Gaussian channel (Chapter 9), we can extend the proof for the discrete multiple-access channel to the Gaussian multiple-access channel. The converse can also be extended similarly, so we expect the capacity region to be the convex hull of the set of rate pairs satisfying

$$R_1 \le I(X_1; Y|X_2), \tag{15.135}$$

$$R_2 \le I(X_2; Y|X_1), \tag{15.136}$$

$$R_1 + R_2 \le I(X_1, X_2; Y) \tag{15.137}$$

for some input distribution $f_1(x_1)f_2(x_2)$ satisfying $EX_1^2 \le P_1$ and $EX_2^2 \le P_2$.

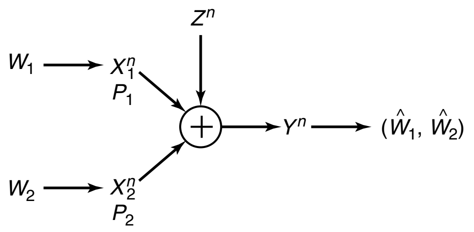


FIGURE 15.16. Gaussian multiple-access channel.

Now, we can expand the mutual information in terms of relative entropy, and thus

$$
\begin{aligned}
I(X_1; Y|X_2) &= h(Y|X_2) - h(Y|X_1, X_2) &(15.138)\\
&= h(X_1 + X_2 + Z|X_2) - h(X_1 + X_2 + Z|X_1, X_2) &(15.139)\\
&= h(X_1 + Z|X_2) - h(Z|X_1, X_2) &(15.140)\\
&= h(X_1 + Z|X_2) - h(Z) &(15.141)\\
&= h(X_1 + Z) - h(Z) &(15.142)\\
&= h(X_1 + Z) - \frac{1}{2}\log(2\pi e)N &(15.143)\\
&\le \frac{1}{2}\log(2\pi e)(P_1 + N) - \frac{1}{2}\log(2\pi e)N &(15.144)\\
&= \frac{1}{2}\log\left(1 + \frac{P_1}{N}\right), &(15.145)
\end{aligned}
$$

where (15.141) follows from the fact that Z is independent of X_1 and X_2, (15.142) from the independence of X_1 and X_2, and (15.144) from the fact that the normal maximizes entropy for a given second moment. Thus, the maximizing distribution is $X_1 \sim \mathcal{N}(0, P_1)$ and $X_2 \sim \mathcal{N}(0, P_2)$ with X_1 and X_2 independent. This distribution simultaneously maximizes the mutual information bounds in (15.135)–(15.137).

Definition We define the channel capacity function

$$C(x) \overset{\triangle}{=} \frac{1}{2}\log(1 + x), \tag{15.146}$$

corresponding to the channel capacity of a Gaussian white-noise channel with signal-to-noise ratio x (Figure 15.17). Then we write the bound on R_1 as

$$R_1 \le C\left(\frac{P_1}{N}\right). \tag{15.147}$$

Similarly,

$$R_2 \le C\left(\frac{P_2}{N}\right) \tag{15.148}$$

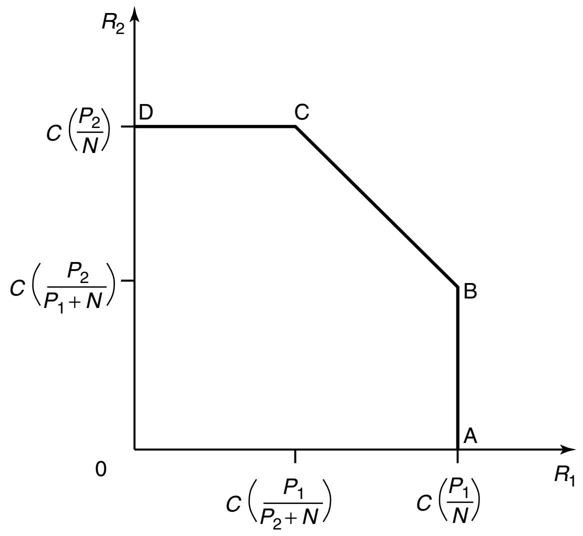


FIGURE 15.17. Gaussian multiple-access channel capacity.

and

$$R_1 + R_2 \le C\left(\frac{P_1 + P_2}{N}\right). \tag{15.149}$$

These upper bounds are achieved when $X_1 \sim \mathcal{N}(0, P_1)$ and $X_2 = \mathcal{N}(0, P_2)$ and define the capacity region. The surprising fact about these inequalities is that the sum of the rates can be as large as $C\left(\frac{P_1+P_2}{N}\right)$, which is that rate achieved by a single transmitter sending with a power equal to the sum of the powers.

The interpretation of the corner points is very similar to the interpretation of the achievable rate pairs for a discrete multiple-access channel for a fixed input distribution. In the case of the Gaussian channel, we can consider decoding as a two-stage process: In the first stage, the receiver decodes the second sender, considering the first sender as part of the noise. This decoding will have low probability of error if $R_2 < C(\frac{P_2}{P_1+N})$. After the second sender has been decoded successfully, it can be subtracted out and the first sender can be decoded correctly if $R_1 < C(\frac{P_1}{N})$. Hence, this argument shows that we can achieve the rate pairs at the corner points of the capacity region by means of single-user operations. This process, called *onion-peeling*, can be extended to any number of users.

If we generalize this to m senders with equal power, the total rate is $C\left(\frac{mP}{N}\right)$, which goes to ∞ as $m \to \infty$. The average rate per sender, $\frac{1}{m}C(\frac{mP}{N})$, goes to 0. Thus, when the total number of senders is very large, so that there is a lot of interference, we can still send a total amount of information that is arbitrarily large even though the rate per individual sender goes to 0.

The capacity region described above corresponds to *code-division multiple access* (CDMA), where separate codes are used for the different senders and the receiver decodes them one by one. In many practical situations, though, simpler schemes, such as frequency-division multiplexing or time-division multiplexing, are used. With *frequency-division multiplexing*, the rates depend on the bandwidth allotted to each sender. Consider the case of two senders with powers P_1 and P_2 using nonintersecting frequency bands with bandwidths W_1 and W_2, where $W_1 + W_2 = W$ (the total bandwidth). Using the formula for the capacity of a single-user bandlimited channel, the following rate pair is achievable:

$$R_1 = W_1 \log\left(1 + \frac{P_1}{NW_1}\right), \tag{15.150}$$

$$R_2 = W_2 \log\left(1 + \frac{P_2}{NW_2}\right). \tag{15.151}$$

As we vary W_1 and W_2, we trace out the curve as shown in Figure 15.18. This curve touches the boundary of the capacity region at one point, which corresponds to allotting bandwidth to each channel proportional to the power in that channel. We conclude that no allocation of frequency bands to radio stations can be optimal unless the allocated powers are proportional to the bandwidths.

In *time-division multiple access* (TDMA), time is divided into slots, and each user is allotted a slot during which only that user will transmit and every other user remains quiet. If there are two users, each of power P, the rate that each sends when the other is silent is $C(P/N)$. Now if time is divided into equal-length slots, and every odd slot is allocated to user 1 and every even slot to user 2, the average rate that each user achieves is $\frac{1}{2}C(P/N)$. This system is called *naive time-division multiple access* (TDMA). However, it is possible to do better if we notice that since user 1 is sending only half the time, it is possible for him to use twice the power during his transmissions and still maintain the same average power constraint. With this modification, it is possible for each user to send information at a rate $\frac{1}{2}C(2P/N)$. By varying the lengths of the slots allotted to each sender (and the instantaneous power used during the slot), we can achieve the same capacity region as FDMA with different bandwidth allocations.

As Figure 15.18 illustrates, in general the capacity region is larger than that achieved by time- or frequency-division multiplexing. But note that

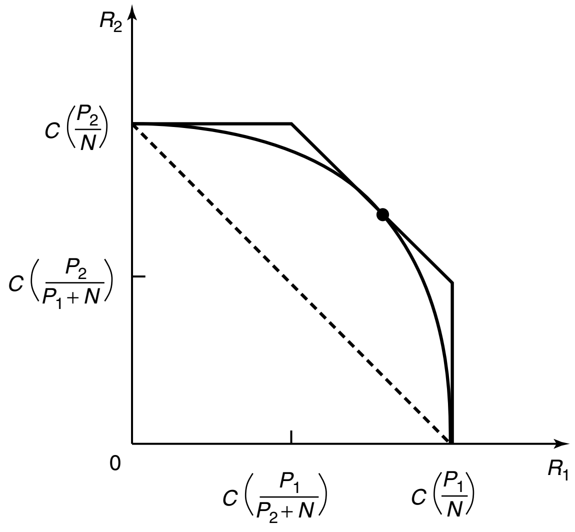


FIGURE 15.18. Gaussian multiple-access channel capacity with FDMA and TDMA.

the multiple-access capacity region derived above is achieved by use of a common decoder for all the senders. However, it is also possible to achieve the capacity region by onion-peeling, which removes the need for a common decoder and instead, uses a sequence of single-user codes. CDMA achieves the entire capacity region, and in addition, allows new users to be added easily without changing the codes of the current users. On the other hand, TDMA and FDMA systems are usually designed for a fixed number of users and it is possible that either some slots are empty (if the actual number of users is less than the number of slots) or some users are left out (if the number of users is greater than the number of slots). However, in many practical systems, simplicity of design is an important consideration, and the improvement in capacity due to the multiple-access ideas presented earlier may not be sufficient to warrant the increased complexity.

For a Gaussian multiple-access system with m sources with powers $P_1, P_2, \ldots, P_m$ and ambient noise of power N, we can state the equivalent of Gauss's law for any set S in the form

$$\sum_{i \in S} R_i = \text{total rate of information flow from } S \tag{15.152}$$

$$\leq C\left(\frac{\sum_{i \in S} P_i}{N}\right). \tag{15.153}$$

15.4 ENCODING OF CORRELATED SOURCES

We now turn to distributed data compression. This problem is in many ways the data compression dual to the multiple-access channel problem. We know how to encode a source X. A rate $R > H(X)$ is sufficient. Now suppose that there are two sources $(X, Y) \sim p(x, y)$. A rate $H(X, Y)$ is sufficient if we are encoding them together. But what if the X and Y sources must be described separately for some user who wishes to reconstruct both X and Y? Clearly, by separately encoding X and Y, it is seen that a rate $R = R_x + R_y > H(X) + H(Y)$ is sufficient. However, in a surprising and fundamental paper by Slepian and Wolf [502], it is shown that a total rate $R = H(X, Y)$ is sufficient even for separate encoding of correlated sources.

Let $(X_1, Y_1), (X_2, Y_2), \ldots$ be a sequence of jointly distributed random variables i.i.d. $\sim p(x, y)$. Assume that the X sequence is available at a location A and the Y sequence is available at a location B. The situation is illustrated in Figure 15.19.

Before we proceed to the proof of this result, we will give a few definitions.

Definition A $((2^{nR_1}, 2^{nR_2}), n)$ *distributed source code* for the joint source (X, Y) consists of two encoder maps,

$$f_1 : \mathcal{X}^n \to \{1, 2, \ldots, 2^{nR_1}\}, \tag{15.154}$$

$$f_2 : \mathcal{Y}^n \to \{1, 2, \ldots, 2^{nR_2}\}, \tag{15.155}$$

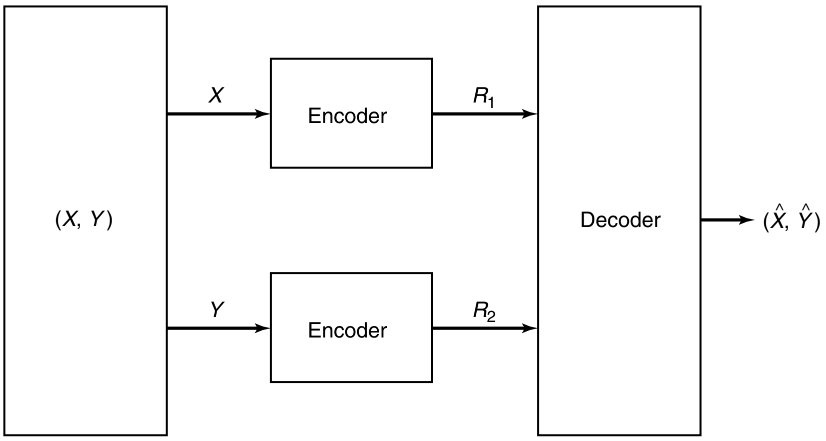


FIGURE 15.19. Slepian–Wolf coding.

and a decoder map,

$$g : \{1, 2, \ldots, 2^{nR_1}\} \times \{1, 2, \ldots, 2^{nR_2}\} \to \mathcal{X}^n \times \mathcal{Y}^n. \tag{15.156}$$

Here $f_1(X^n)$ is the index corresponding to X^n, $f_2(Y^n)$ is the index corresponding to Y^n, and (R_1, R_2) is the rate pair of the code.

Definition The *probability of error* for a distributed source code is defined as

$$P_e^{(n)} = P(g(f_1(X^n), f_2(Y^n)) \neq (X^n, Y^n)). \tag{15.157}$$

Definition A rate pair (R_1, R_2) is said to be *achievable* for a distributed source if there exists a sequence of $((2^{nR_1}, 2^{nR_2}), n)$ distributed source codes with probability of error $P_e^{(n)} \to 0$. The *achievable rate region* is the closure of the set of achievable rates.

Theorem 15.4.1 (*Slepian–Wolf*) *For the distributed source coding problem for the source (X, Y) drawn i.i.d $\sim p(x, y)$, the achievable rate region is given by*

$$R_1 \geq H(X|Y), \tag{15.158}$$

$$R_2 \geq H(Y|X), \tag{15.159}$$

$$R_1 + R_2 \geq H(X, Y). \tag{15.160}$$

Let us illustrate the result with some examples.

Example 15.4.1 Consider the weather in Gotham and Metropolis. For the purposes of our example, we assume that Gotham is sunny with probability 0.5 and that the weather in Metropolis is the same as in Gotham with probability 0.89. The joint distribution of the weather is given as follows:

	Metropolis	
$p(x, y)$	Rain	Shine
Gotham		
Rain	0.445	0.055
Shine	0.055	0.445

Assume that we wish to transmit 100 days of weather information to the National Weather Service headquarters in Washington. We could send all the 100 bits of the weather in both places, making 200 bits in all. If we decided to compress the information independently, we would still need $100H(0.5) = 100$ bits of information from each place, for a total of 200 bits. If, instead, we use Slepian–Wolf encoding, we need only $H(X) + H(Y|X) = 100H(0.5) + 100H(0.89) = 100 + 50 = 150$ bits total.

Example 15.4.2 Consider the following joint distribution:

$p(u, v)$	0	1
0	$\frac{1}{3}$	$\frac{1}{3}$
1	0	$\frac{1}{3}$

In this case, the total rate required for the transmission of this source is $H(U) + H(V|U) = \log 3 = 1.58$ bits rather than the 2 bits that would be needed if the sources were transmitted independently without Slepian–Wolf encoding.

15.4.1 Achievability of the Slepian–Wolf Theorem

We now prove the achievability of the rates in the Slepian–Wolf theorem. Before we proceed to the proof, we introduce a new coding procedure using random bins. The essential idea of random bins is very similar to hash functions: We choose a large random index for each source sequence. If the set of typical source sequences is small enough (or equivalently, the range of the hash function is large enough), then with high probability, different source sequences have different indices, and we can recover the source sequence from the index.

Let us consider the application of this idea to the encoding of a single source. In Chapter 3 the method that we considered was to index all elements of the typical set and not bother about elements outside the typical set. We will now describe the random binning procedure, which indexes all sequences but rejects untypical sequences at a later stage.

Consider the following procedure: For each sequence X^n, draw an index at random from $\{1, 2, \ldots, 2^{nR}\}$. The set of sequences X^n which have the same index are said to form a *bin*, since this can be viewed as first laying down a row of bins and then throwing the X^n's at random into the bins. For decoding the source from the bin index, we look for a typical X^n sequence in the bin. If there is one and only one typical X^n sequence in the bin, we declare it to be the estimate $\hat{X}^n$ of the source sequence; otherwise, an error is declared.

The above procedure defines a source code. To analyze the probability of error for this code, we will now divide the X^n sequences into two types, typical sequences and nontypical sequences. If the source sequence is typical, the bin corresponding to this source sequence will contain at least one typical sequence (the source sequence itself). Hence there will be an error only if there is more than one typical sequence in this bin. If the source sequence is nontypical, there will always be an error. But if the number of bins is much larger than the number of typical sequences, the probability that there is more than one typical sequence in a bin is very small, and hence the probability that a typical sequence will result in an error is very small.

Formally, let $f(X^n)$ be the bin index corresponding to X^n. Call the decoding function g. The probability of error (averaged over the random choice ofcodes f) is

$$
\begin{aligned}
P(g(f(\mathbf{X})) \neq \mathbf{X}) &\le P(\mathbf{X} \notin A_\epsilon^{(n)}) + \sum_{\mathbf{x}} P(\exists \mathbf{x}' \neq \mathbf{x} : \mathbf{x}' \in A_\epsilon^{(n)}, f(\mathbf{x}') \\
&= f(\mathbf{x}))p(\mathbf{x}) \\
&\le \epsilon + \sum_{\mathbf{x}} \sum_{\substack{\mathbf{x}' \in A_\epsilon^{(n)} \\ \mathbf{x}' \neq \mathbf{x}}} P(f(\mathbf{x}') = f(\mathbf{x}))p(\mathbf{x}) && (15.161)\\
&\le \epsilon + \sum_{\mathbf{x}} \sum_{\mathbf{x}' \in A_\epsilon^{(n)}} 2^{-nR} p(\mathbf{x}) && (15.162)\\
&= \epsilon + \sum_{\mathbf{x}' \in A_\epsilon^{(n)}} 2^{-nR} \sum_{\mathbf{x}} p(\mathbf{x}) && (15.163)\\
&\le \epsilon + \sum_{\mathbf{x}' \in A_\epsilon^{(n)}} 2^{-nR} && (15.164)\\
&\le \epsilon + 2^{n(H(X)+\epsilon)} 2^{-nR} && (15.165)\\
&\le 2\epsilon && (15.166)
\end{aligned}
$$

if $R > H(X) + \epsilon$ and n is sufficiently large. Hence, if the rate of the code is greater than the entropy, the probability of error is arbitrarily small and the code achieves the same results as the code described in Chapter 3.

The above example illustrates the fact that there are many ways to construct codes with low probabilities of error at rates above the entropy of the source; the universal source code is another example of such a code.

Note that the binning scheme does not require an explicit characterization of the typical set at the encoder; it is needed only at the decoder. It is this property that enables this code to continue to work in the case of a distributed source, as illustrated in the proof of the theorem.

We now return to the consideration of the distributed source coding and prove the achievability of the rate region in the Slepian–Wolf theorem.

Proof: (*Achievability in Theorem 15.4.1*). The basic idea of the proof is to partition the space of $\mathcal{X}^n$ into 2^{nR_1} bins and the space of $\mathcal{Y}^n$ into 2^{nR_2} bins.

Random code generation: Assign every $\mathbf{x} \in \mathcal{X}^n$ to one of 2^{nR_1} bins independently according to a uniform distribution on $\{1, 2, \ldots, 2^{nR_1}\}$. Similarly, randomly assign every $\mathbf{y} \in \mathcal{Y}^n$ to one of 2^{nR_2} bins. Reveal the assignments f_1 and f_2 to both the encoder and the decoder.

Encoding: Sender 1 sends the index of the bin to which $\mathbf{X}$ belongs. Sender 2 sends the index of the bin to which $\mathbf{Y}$ belongs.

Decoding: Given the received index pair (i_0, j_0), declare $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = (\mathbf{x}, \mathbf{y})$ if there is one and only one pair of sequences $(\mathbf{x}, \mathbf{y})$ such that $f_1(\mathbf{x}) = i_0$, $f_2(\mathbf{y}) = j_0$ and $(\mathbf{x}, \mathbf{y}) \in A_\epsilon^{(n)}$. Otherwise, declare an error. The scheme is illustrated in Figure 15.20. The set of X sequences and the set of Y sequences are divided into bins in such a way that the pair of indices specifies a product bin.

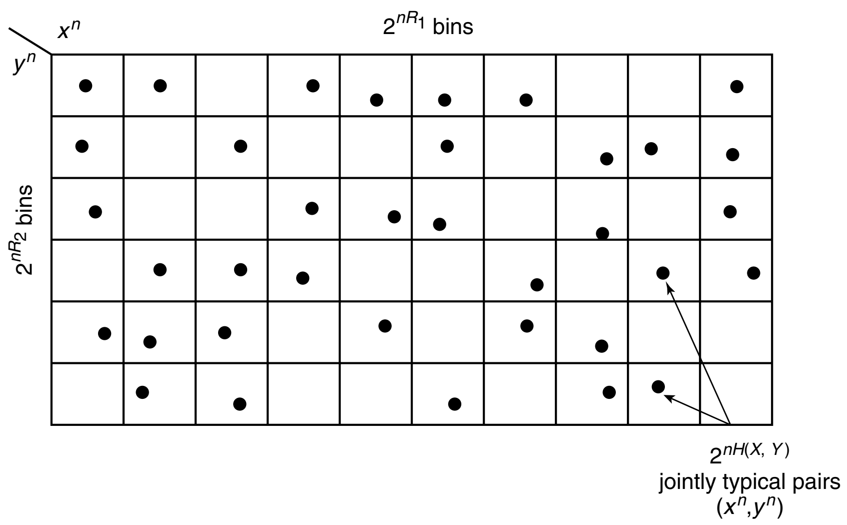


FIGURE 15.20. Slepian–Wolf encoding: the jointly typical pairs are isolated by the product bins.

Probability of error: Let $(X_i, Y_i) \sim p(x, y)$. Define the events

$$E_0 = \{(\mathbf{X}, \mathbf{Y}) \notin A_\epsilon^{(n)}\}, \tag{15.167}$$

$$E_1 = \{\exists \mathbf{x}' \neq \mathbf{X} : f_1(\mathbf{x}') = f_1(\mathbf{X}) \text{ and } (\mathbf{x}', \mathbf{Y}) \in A_\epsilon^{(n)}\}, \tag{15.168}$$

$$E_2 = \{\exists \mathbf{y}' \neq \mathbf{Y} : f_2(\mathbf{y}') = f_2(\mathbf{Y}) \text{ and } (\mathbf{X}, \mathbf{y}') \in A_\epsilon^{(n)}\}, \tag{15.169}$$

and

$$\begin{aligned} E_{12} = \{\exists (\mathbf{x}', \mathbf{y}') : \mathbf{x}' \neq \mathbf{X}, \mathbf{y}' \neq \mathbf{Y}, f_1(\mathbf{x}') \\ = f_1(\mathbf{X}), f_2(\mathbf{y}') = f_2(\mathbf{Y}) \text{ and } (\mathbf{x}', \mathbf{y}') \in A_\epsilon^{(n)}\}. \end{aligned} \tag{15.170}$$

Here $\mathbf{X}, \mathbf{Y}$, f_1, and f_2 are random. We have an error if $(\mathbf{X}, \mathbf{Y})$ is not in $A_\epsilon^{(n)}$ or if there is another typical pair in the same bin. Hence by the union of events bound,

$$\begin{aligned} P_e^{(n)} &= P(E_0 \cup E_1 \cup E_2 \cup E_{12}) && (15.171) \\ &\leq P(E_0) + P(E_1) + P(E_2) + P(E_{12}). && (15.172) \end{aligned}$$

First consider E_0. By the AEP, $P(E_0) \to 0$ and hence for n sufficiently large, $P(E_0) < \epsilon$. To bound $P(E_1)$, we have

$$\begin{aligned} P(E_1) &= P\{\exists \mathbf{x}' \neq \mathbf{X} : f_1(\mathbf{x}') = f_1(\mathbf{X}), \text{ and } (\mathbf{x}', \mathbf{Y}) \in A_\epsilon^{(n)}\} && (15.173) \\ &= \sum_{(\mathbf{x},\mathbf{y})} p(\mathbf{x}, \mathbf{y}) P\{\exists \mathbf{x}' \neq \mathbf{x} : f_1(\mathbf{x}') = f_1(\mathbf{x}), \quad (\mathbf{x}', \mathbf{y}) \in A_\epsilon^{(n)}\} && (15.174) \\ &\leq \sum_{(\mathbf{x},\mathbf{y})} p(\mathbf{x}, \mathbf{y}) \sum_{\substack{\mathbf{x}' \neq \mathbf{x} \\ (\mathbf{x}', \mathbf{y}) \in A_\epsilon^{(n)}}} P(f_1(\mathbf{x}') = f_1(\mathbf{x})) && (15.175) \\ &= \sum_{(\mathbf{x},\mathbf{y})} p(\mathbf{x}, \mathbf{y}) 2^{-nR_1} |A_\epsilon(X|\mathbf{y})| && (15.176) \\ &\leq 2^{-nR_1} 2^{n(H(X|Y)+\epsilon)} \qquad \text{(by Theorem 15.2.2)}, && (15.177) \end{aligned}$$

which goes to 0 if $R_1 > H(X|Y)$. Hence for sufficiently large n, $P(E_1) < \epsilon$. Similarly, for sufficiently large n, $P(E_2) < \epsilon$ if $R_2 > H(Y|X)$ and $P(E_{12}) < \epsilon$ if $R_1 + R_2 > H(X, Y)$. Since the average probability of error is $< 4\epsilon$, there exists at least one code (f_1^*, f_2^*, g^*) with probability of error $< 4\epsilon$. Thus, we can construct a sequence of codes with $P_e^{(n)} \to 0$, and the proof of achievability is complete. □

15.4.2 Converse for the Slepian–Wolf Theorem

The converse for the Slepian–Wolf theorem follows obviously from the results for a single source, but we will provide it for completeness.

Proof: (*Converse to Theorem 15.4.1*). As usual, we begin with Fano's inequality. Let f_1, f_2, g be fixed. Let $I_0 = f_1(X^n)$ and $J_0 = f_2(Y^n)$. Then

$$H(X^n, Y^n|I_0, J_0) \le P_e^{(n)} n(\log|\mathcal{X}| + \log|\mathcal{Y}|) + 1 = n\epsilon_n, \tag{15.178}$$

where $\epsilon_n \to 0$ as $n \to \infty$. Now adding conditioning, we also have

$$H(X^n|Y^n, I_0, J_0) \le n\epsilon_n, \tag{15.179}$$

and

$$H(Y^n|X^n, I_0, J_0) \le n\epsilon_n. \tag{15.180}$$

We can write a chain of inequalities

$$n(R_1 + R_2) \overset{(a)}{\ge} H(I_0, J_0) \tag{15.181}$$
$$= I(X^n, Y^n; I_0, J_0) + H(I_0, J_0|X^n, Y^n) \tag{15.182}$$
$$\overset{(b)}{=} I(X^n, Y^n; I_0, J_0) \tag{15.183}$$
$$= H(X^n, Y^n) - H(X^n, Y^n|I_0, J_0) \tag{15.184}$$
$$\overset{(c)}{\ge} H(X^n, Y^n) - n\epsilon_n \tag{15.185}$$
$$\overset{(d)}{=} nH(X, Y) - n\epsilon_n, \tag{15.186}$$

where

(a) follows from the fact that $I_0 \in \{1, 2, \ldots, 2^{nR_1}\}$ and $J_0 \in \{1, 2, \ldots, 2^{nR_2}\}$
(b) follows from the fact the I_0 is a function of X^n and J_0 is a function of Y^n
(c) follows from Fano's inequality (15.178)
(d) follows from the chain rule and the fact that (X_i, Y_i) are i.i.d.

Similarly, using (15.179), we have

$$nR_1 \overset{(a)}{\geq} H(I_0) \tag{15.187}$$

$$\geq H(I_0|Y^n) \tag{15.188}$$

$$= I(X^n; I_0|Y^n) + H(I_0|X^n, Y^n) \tag{15.189}$$

$$\overset{(b)}{=} I(X^n; I_0|Y^n) \tag{15.190}$$

$$= H(X^n|Y^n) - H(X^n|I_0, J_0, Y^n) \tag{15.191}$$

$$\overset{(c)}{\geq} H(X^n|Y^n) - n\epsilon_n \tag{15.192}$$

$$\overset{(d)}{=} nH(X|Y) - n\epsilon_n, \tag{15.193}$$

where the reasons are the same as for the equations above. Similarly, we can show that

$$nR_2 \geq nH(Y|X) - n\epsilon_n. \tag{15.194}$$

Dividing these inequalities by n and taking the limit as $n \to \infty$, we have the desired converse. □

The region described in the Slepian–Wolf theorem is illustrated in Figure 15.21.

15.4.3 Slepian–Wolf Theorem for Many Sources

The results of Section 15.4.2 can easily be generalized to many sources. The proof follows exactly the same lines.

Theorem 15.4.2 *Let $(X_{1i}, X_{2i}, \ldots, X_{mi})$ be i.i.d. $\sim p(x_1, x_2, \ldots, x_m)$. Then the set of rate vectors achievable for distributed source coding with separate encoders and a common decoder is defined by*

$$R(S) > H(X(S)|X(S^c)) \tag{15.195}$$

for all $S \subseteq \{1, 2, \ldots, m\}$, where

$$R(S) = \sum_{i \in S} R_i \tag{15.196}$$

and $X(S) = \{X_j : j \in S\}$.

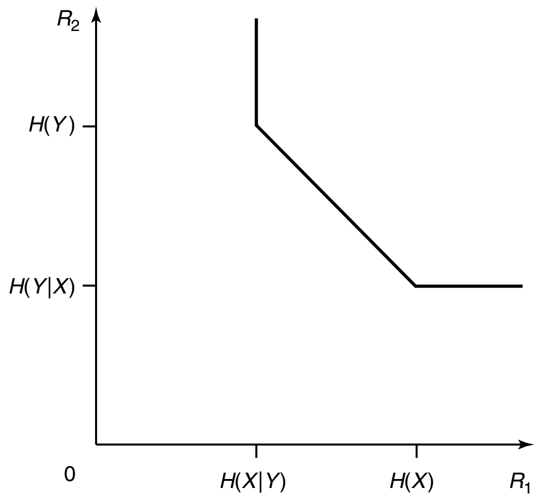


FIGURE 15.21. Rate region for Slepian–Wolf encoding.

Proof: The proof is identical to the case of two variables and is omitted. □

The achievability of Slepian–Wolf encoding has been proved for an i.i.d. correlated source, but the proof can easily be extended to the case of an arbitrary joint source that satisfies the AEP; in particular, it can be extended to the case of any jointly ergodic source [122]. In these cases the entropies in the definition of the rate region are replaced by the corresponding entropy rates.

15.4.4 Interpretation of Slepian–Wolf Coding

We consider an interpretation of the corner points of the rate region in Slepian–Wolf encoding in terms of graph coloring. Consider the point with rate $R_1 = H(X)$, $R_2 = H(Y|X)$. Using $nH(X)$ bits, we can encode X^n efficiently, so that the decoder can reconstruct X^n with arbitrarily low probability of error. But how do we code Y^n with $nH(Y|X)$ bits? Looking at the picture in terms of typical sets, we see that associated with every X^n is a typical "fan" of Y^n sequences that are jointly typical with the given X^n as shown in Figure 15.22.

If the Y encoder knows X^n, the encoder can send the index of the Y^n within this typical fan. The decoder, also knowing X^n, can then construct this typical fan and hence reconstruct Y^n. But the Y encoder does not know X^n. So instead of trying to determine the typical fan, he randomly

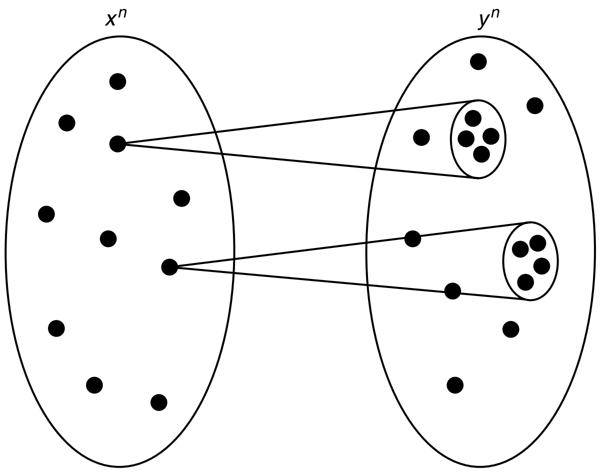


FIGURE 15.22. Jointly typical fans.

colors all Y^n sequences with 2^{nR_2} colors. If the number of colors is high enough, then with high probability all the colors in a particular fan will be different and the color of the Y^n sequence will uniquely define the Y^n sequence within the X^n fan. If the rate $R_2 > H(Y|X)$, the number of colors is exponentially larger than the number of elements in the fan and we can show that the scheme will have an exponentially small probability of error.

15.5 DUALITY BETWEEN SLEPIAN–WOLF ENCODING AND MULTIPLE-ACCESS CHANNELS

With multiple-access channels, we considered the problem of sending independent messages over a channel with two inputs and only one output. With Slepian–Wolf encoding, we considered the problem of sending a correlated source over a noiseless channel, with a common decoder for recovery of both sources. In this section we explore the duality between the two systems.

In Figure 15.23, two independent messages are to be sent over the channel as X_1^n and X_2^n sequences. The receiver estimates the messages from the received sequence. In Figure 15.24 the correlated sources are encoded as “independent” messages i and j. The receiver tries to estimate the source sequences from knowledge of i and j.

In the proof of the achievability of the capacity region for the multiple-access channel, we used a random map from the set of messages to the

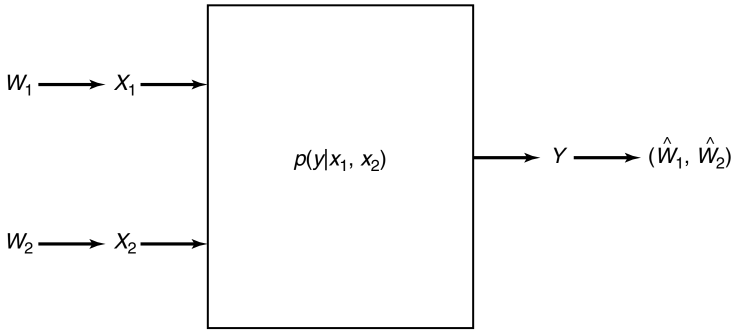


FIGURE 15.23. Multiple-access channels.

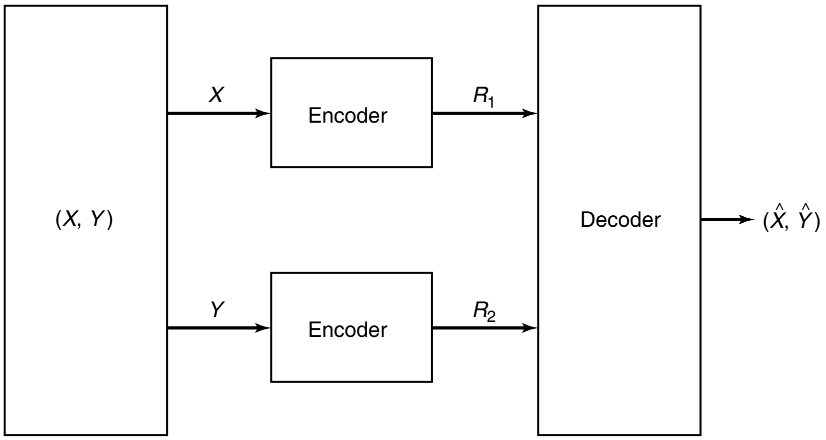


FIGURE 15.24. Correlated source encoding.

sequences X_1^n and X_2^n. In the proof for Slepian–Wolf coding, we used a random map from the set of sequences X^n and Y^n to a set of messages. In the proof of the coding theorem for the multiple-access channel, the probability of error was bounded by

$$P_e^{(n)} \leq \epsilon + \sum_{\text{codewords}} \Pr(\text{codeword jointly typical with sequence received}) \tag{15.197}$$

$$= \epsilon + \sum_{2^{nR_1} \text{ terms}} 2^{-nI_1} + \sum_{2^{nR_2} \text{ terms}} 2^{-nI_2} + \sum_{2^{n(R_1+R_2)} \text{ terms}} 2^{-nI_3}, \tag{15.198}$$

where ϵ is the probability the sequences are not typical, R_i are the rates corresponding to the number of codewords that can contribute to the probability of error, and I_i is the corresponding mutual information that corresponds to the probability that the codeword is jointly typical with the received sequence.

In the case of Slepian–Wolf encoding, the corresponding expression for the probability of error is

$$P_e^{(n)} \le \epsilon + \sum_{\text{jointly typical sequences}} \Pr(\text{ have the same codeword}) \tag{15.199}$$

$$= \epsilon + \sum_{2^{nH_1}\text{ terms}} 2^{-nR_1} + \sum_{2^{nH_2}\text{ terms}} 2^{-nR_2} + \sum_{2^{nH_3}\text{ terms}} 2^{-n(R_1+R_2)}, \tag{15.200}$$

where again the probability that the constraints of the AEP are not satisfied is bounded by ϵ, and the other terms refer to the various ways in which another pair of sequences could be jointly typical and in the same bin as the given source pair.

The duality of the multiple-access channel and correlated source encoding is now obvious. It is rather surprising that these two systems are duals of each other; one would have expected a duality between the broadcast channel and the multiple-access channel.

15.6 BROADCAST CHANNEL

The broadcast channel is a communication channel in which there is one sender and two or more receivers. It is illustrated in Figure 15.25. The basic problem is to find the set of simultaneously achievable rates for communication in a broadcast channel. Before we begin the analysis, let us consider some examples.

Example 15.6.1 (*TV station*) The simplest example of the broadcast channel is a radio or TV station. But this example is slightly degenerate in the sense that normally the station wants to send the same information to everybody who is tuned in; the capacity is essentially $\max_{p(x)} \min_i I(X; Y_i)$, which may be less than the capacity of the worst receiver. But we may wish to arrange the information in such a way that the better receivers receive extra information, which produces a better picture or sound, while the worst receivers continue to receive more basic information. As TV stations introduce high-definition TV (HDTV), it may be necessary to encode the information so that bad receivers will receive the

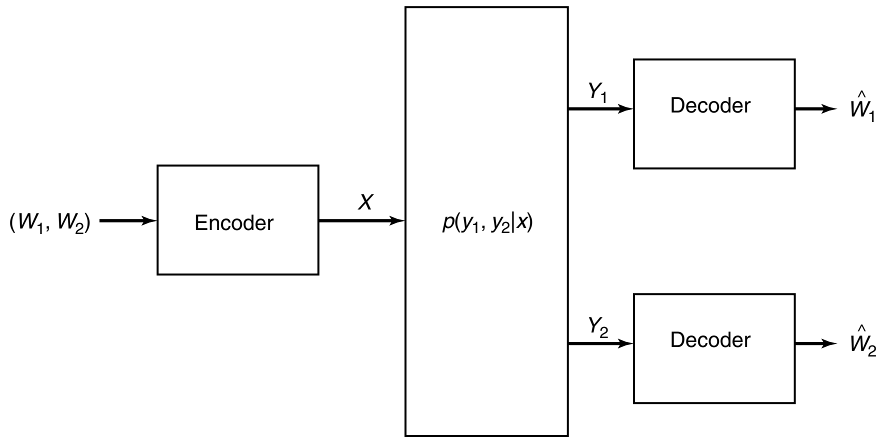


FIGURE 15.25. Broadcast channel.

regular TV signal, while good receivers will receive the extra information for the high-definition signal. The methods to accomplish this will be explained in the discussion of the broadcast channel.

Example 15.6.2 (*Lecturer in classroom*) A lecturer in a classroom is communicating information to the students in the class. Due to differences among the students, they receive various amounts of information. Some of the students receive most of the information; others receive only a little. In the ideal situation, the lecturer would be able to tailor his or her lecture in such a way that the good students receive more information and the poor students receive at least the minimum amount of information. However, a poorly prepared lecture proceeds at the pace of the weakest student. This situation is another example of a broadcast channel.

Example 15.6.3 (*Orthogonal broadcast channels*) The simplest broadcast channel consists of two independent channels to the two receivers. Here we can send independent information over both channels, and we can achieve rate R_1 to receiver 1 and rate R_2 to receiver 2 if $R_1 < C_1$ and $R_2 < C_2$. The capacity region is the rectangle shown in Figure 15.26.

Example 15.6.4 (*Spanish and Dutch speaker*) To illustrate the idea of superposition, we will consider a simplified example of a speaker who can speak both Spanish and Dutch. There are two listeners: One understands only Spanish and the other understands only Dutch. Assume for simplicity that the vocabulary of each language is 2^{20} words and that the speaker speaks at the rate of 1 word per second in either language. Then he

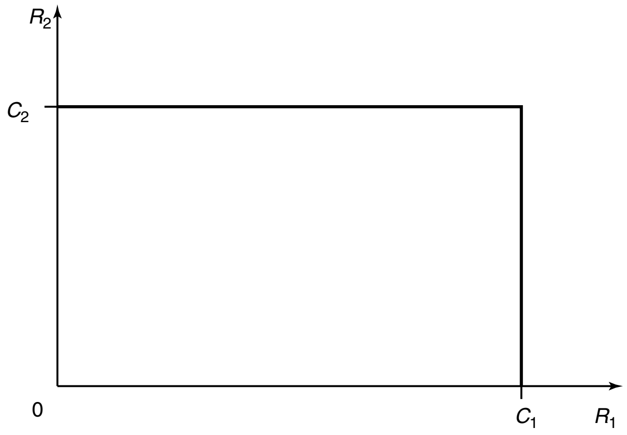


FIGURE 15.26. Capacity region for two orthogonal broadcast channels.

can transmit 20 bits of information per second to receiver 1 by speaking to her all the time; in this case, he sends no information to receiver 2. Similarly, he can send 20 bits per second to receiver 2 without sending any information to receiver 1. Thus, he can achieve any rate pair with $R_1 + R_2 = 20$ by simple time-sharing. But can he do better?

Recall that the Dutch listener, even though he does not understand Spanish, can recognize when the word is Spanish. Similarly, the Spanish listener can recognize when Dutch occurs. The speaker can use this to convey information; for example, if the proportion of time he uses each language is 50%, then of a sequence of 100 words, about 50 will be Dutch and about 50 will be Spanish. But there are many ways to order the Spanish and Dutch words; in fact, there are about $\binom{100}{50} \approx 2^{100H(\frac{1}{2})}$ ways to order the words. Choosing one of these orderings conveys information to both listeners. This method enables the speaker to send information at a rate of 10 bits per second to the Dutch receiver, 10 bits per second to the Spanish receiver, and 1 bit per second of common information to both receivers, for a total rate of 21 bits per second, which is more than that achievable by simple timesharing. This is an example of superposition of information.

The results of the broadcast channel can also be applied to the case of a single-user channel with an unknown distribution. In this case, the objective is to get at least the minimum information through when the channel is bad and to get some extra information through when the channel is good. We can use the same superposition arguments as in the case of the broadcast channel to find the rates at which we can send information.

15.6.1 Definitions for a Broadcast Channel

Definition A *broadcast channel* consists of an input alphabet $\mathcal{X}$ and two output alphabets, $\mathcal{Y}_1$ and $\mathcal{Y}_2$, and a probability transition function $p(y_1, y_2|x)$. The broadcast channel will be said to be *memoryless* if $p(y_1^n, y_2^n|x^n) = \prod_{i=1}^n p(y_{1i}, y_{2i}|x_i)$.

We define codes, probability of error, achievability, and capacity regions for the broadcast channel as we did for the multiple-access channel. A $((2^{nR_1}, 2^{nR_2}), n)$ code for a broadcast channel with independent information consists of an encoder,

$$X : (\{1, 2, \ldots, 2^{nR_1}\} \times \{1, 2, \ldots, 2^{nR_2}\}) \to \mathcal{X}^n, \tag{15.201}$$

and two decoders,

$$g_1 : \mathcal{Y}_1^n \to \{1, 2, \ldots, 2^{nR_1}\} \tag{15.202}$$

and

$$g_2 : \mathcal{Y}_2^n \to \{1, 2, \ldots, 2^{nR_2}\}. \tag{15.203}$$

We define the average probability of error as the probability that the decoded message is not equal to the transmitted message; that is,

$$P_e^{(n)} = P(g_1(Y_1^n) \neq W_1 \quad \text{or} \quad g_2(Y_2^n) \neq W_2), \tag{15.204}$$

where (W_1, W_2) are assumed to be uniformly distributed over $2^{nR_1} \times 2^{nR_2}$.

Definition A rate pair (R_1, R_2) is said to be *achievable* for the broadcast channel if there exists a sequence of $((2^{nR_1}, 2^{nR_2}), n)$ codes with $P_e^{(n)} \to 0$.

We will now define the rates for the case where we have common information to be sent to both receivers. A $((2^{nR_0}, 2^{nR_1}, 2^{nR_2}), n)$ code for a broadcast channel with common information consists of an encoder,

$$X : (\{1, 2, \ldots, 2^{nR_0}\} \times \{1, 2, \ldots, 2^{nR_1}\} \times \{1, 2, \ldots, 2^{nR_2}\}) \to \mathcal{X}^n, \tag{15.205}$$

and two decoders,

$$g_1 : \mathcal{Y}_1^n \to \{1, 2, \ldots, 2^{nR_0}\} \times \{1, 2, \ldots, 2^{nR_1}\} \tag{15.206}$$

and

$$g_2 : \mathcal{Y}_2^n \to \{1, 2, \ldots, 2^{nR_0}\} \times \{1, 2, \ldots, 2^{nR_2}\}. \tag{15.207}$$

Assuming that the distribution on (W_0, W_1, W_2) is uniform, we can define the probability of error as the probability that the decoded message is not equal to the transmitted message:

$$P_e^{(n)} = P(g_1(Y_1^n) \neq (W_0, W_1) \text{ or } g_2(Z^n) \neq (W_0, W_2)). \tag{15.208}$$

Definition A rate triple (R_0, R_1, R_2) is said to be *achievable* for the broadcast channel with common information if there exists a sequence of $((2^{nR_0}, 2^{nR_1}, 2^{nR_2}), n)$ codes with $P_e^{(n)} \to 0$.

Definition The *capacity region* of the broadcast channel is the closure of the set of achievable rates.

We observe that an error for receiver Y_1^n depends only the distribution $p(x^n, y_1^n)$ and not on the joint distribution $p(x^n, y_1^n, y_2^n)$. Thus, we have the following theorem:

Theorem 15.6.1 *The capacity region of a broadcast channel depends only on the conditional marginal distributions $p(y_1|x)$ and $p(y_2|x)$.*

Proof: See the problems. □

15.6.2 Degraded Broadcast Channels

Definition A broadcast channel is said to be *physically degraded* if $p(y_1, y_2|x) = p(y_1|x)p(y_2|y_1)$.

Definition A broadcast channel is said to be *stochastically degraded* if its conditional marginal distributions are the same as that of a physically degraded broadcast channel; that is, if there exists a distribution $p'(y_2|y_1)$ such that

$$p(y_2|x) = \sum_{y_1} p(y_1|x)p'(y_2|y_1). \tag{15.209}$$

Note that since the capacity of a broadcast channel depends only on the conditional marginals, the capacity region of the stochastically degraded broadcast channel is the same as that of the corresponding physically degraded channel. In much of the following, we therefore assume that the channel is physically degraded.

15.6.3 Capacity Region for the Degraded Broadcast Channel

We now consider sending independent information over a degraded broadcast channel at rate R_1 to Y_1 and rate R_2 to Y_2.

Theorem 15.6.2 *The capacity region for sending independent information over the degraded broadcast channel $X \to Y_1 \to Y_2$ is the convex hull of the closure of all (R_1, R_2) satisfying*

$$R_2 \le I(U; Y_2), \tag{15.210}$$

$$R_1 \le I(X; Y_1|U) \tag{15.211}$$

for some joint distribution $p(u)p(x|u)p(y_1, y_2|x)$, where the auxiliary random variable U has cardinality bounded by $|\mathcal{U}| \le \min\{|\mathcal{X}|, |\mathcal{Y}_1|, |\mathcal{Y}_2|\}$.

Proof: (The cardinality bounds for the auxiliary random variable U are derived using standard methods from convex set theory and are not dealt with here.) We first give an outline of the basic idea of superposition coding for the broadcast channel. The auxiliary random variable U will serve as a cloud center that can be distinguished by both receivers Y_1 and Y_2. Each cloud consists of 2^{nR_1} codewords X^n distinguishable by the receiver Y_1. The worst receiver can only see the clouds, while the better receiver can see the individual codewords within the clouds. The formal proof of the achievability of this region uses a random coding argument: Fix $p(u)$ and $p(x|u)$.

Random codebook generation: Generate 2^{nR_2} independent codewords of length n, $\mathbf{U}(w_2)$, $w_2 \in \{1, 2, \ldots, 2^{nR_2}\}$, according to $\prod_{i=1}^n p(u_i)$. For each codeword $\mathbf{U}(w_2)$, generate 2^{nR_1} independent codewords $\mathbf{X}(w_1, w_2)$ according to $\prod_{i=1}^n p(x_i|u_i(w_2))$. Here $\mathbf{u}(i)$ plays the role of the cloud center understandable to both Y_1 and Y_2, while $\mathbf{x}(i, j)$ is the jth satellite codeword in the ith cloud.

Encoding: To send the pair (W_1, W_2), send the corresponding codeword $\mathbf{X}(W_1, W_2)$.

Decoding: Receiver 2 determines the unique $\hat{\hat{W}}_2$ such that $(\mathbf{U}(\hat{\hat{W}}_2), \mathbf{Y}_2) \in A_\epsilon^{(n)}$. If there are none such or more than one such, an error is declared.

Receiver 1 looks for the unique $(\hat{W}_1, \hat{W}_2)$ such that $(\mathbf{U}(\hat{W}_2), \mathbf{X}(\hat{W}_1, \hat{W}_2), \mathbf{Y}_1) \in A_\epsilon^{(n)}$. If there are none such or more than one such, an error is declared.

Analysis of the probability of error: By the symmetry of the code generation, the probability of error does not depend on which codeword was sent. Hence, without loss of generality, we can assume that the message pair $(W_1, W_2) = (1, 1)$ was sent. Let $P(\cdot)$ denote the conditional probability of an event given that (1,1) was sent.

Since we have essentially a single-user channel from U to Y_2, we will be able to decode the U codewords with a low probability of error if $R_2 < I(U; Y_2)$. To prove this, we define the events

$$E_{Yi} = \{(\mathbf{U}(i), \mathbf{Y}_2) \in A_\epsilon^{(n)}\}. \tag{15.212}$$

Then the probability of error at receiver 2 is

$$\begin{aligned} P_e^{(n)}(2) &= P(E_{Y1}^c \bigcup \bigcup_{i\neq 1} E_{Yi}) && (15.213)\\ &\le P(E_{Y1}^c) + \sum_{i\neq 1} P(E_{Yi}) && (15.214)\\ &\le \epsilon + 2^{nR_2} 2^{-n(I(U;Y_2)-2\epsilon)} && (15.215)\\ &\le 2\epsilon && (15.216)\end{aligned}$$

if n is large enough and $R_2 < I(U; Y_2)$, where (15.215) follows from the AEP. Similarly, for decoding for receiver 1, we define the events

$$\begin{aligned} \tilde{E}_{Yi} &= \{(\mathbf{U}(i), \mathbf{Y}_1) \in A_\epsilon^{(n)}\}, && (15.217)\\ \tilde{E}_{Yij} &= \{(\mathbf{U}(i), \mathbf{X}(i, j), \mathbf{Y}_1) \in A_\epsilon^{(n)}\}, && (15.218)\end{aligned}$$

where the tilde refers to events defined at receiver 1. Then we can bound the probability of error as

$$\begin{aligned} P_e^{(n)}(1) &= P\left(\tilde{E}_{Y1}^c \bigcup \tilde{E}_{Y11}^c \bigcup \bigcup_{i\neq 1} \tilde{E}_{Yi} \bigcup \bigcup_{j\neq 1} \tilde{E}_{Y1j}\right) && (15.219)\\ &\le P(\tilde{E}_{Y1}^c) + P(\tilde{E}_{Y11}^c) + \sum_{i\neq 1} P(\tilde{E}_{Yi}) + \sum_{j\neq 1} P(\tilde{E}_{Y1j}). && (15.220)\end{aligned}$$

By the same arguments as for receiver 2, we can bound $P(\tilde{E}_{Yi}) \le 2^{-n(I(U;Y_1)-3\epsilon)}$. Hence, the third term goes to 0 if $R_2 < I(U; Y_1)$. But by the data-processing inequality and the degraded nature of the channel, $I(U; Y_1) \ge I(U; Y_2)$, and hence the conditions of the theorem imply that the third term goes to 0. We can also bound the fourth term in the probability of error as

$$P(\tilde{E}_{Y1j}) = P((\mathbf{U}(1), \mathbf{X}(1, j), \mathbf{Y}_1) \in A_\epsilon^{(n)}) \tag{15.221}$$

$$= \sum_{(\mathbf{U},\mathbf{X},\mathbf{Y}_1)\in A_\epsilon^{(n)}} P((\mathbf{U}(1), \mathbf{X}(1, j), \mathbf{Y}_1)) \tag{15.222}$$

$$= \sum_{(\mathbf{U},\mathbf{X},\mathbf{Y}_1)\in A_\epsilon^{(n)}} P(\mathbf{U}(1))P(\mathbf{X}(1, j)|\mathbf{U}(1))P(\mathbf{Y}_1|\mathbf{U}(1)) \tag{15.223}$$

$$\leq \sum_{(\mathbf{U},\mathbf{X},\mathbf{Y}_1)\in A_\epsilon^{(n)}} 2^{-n(H(U)-\epsilon)}2^{-n(H(X|U)-\epsilon)}2^{-n(H(Y_1|U)-\epsilon)} \tag{15.224}$$

$$\leq 2^{n(H(U,X,Y_1)+\epsilon)}2^{-n(H(U)-\epsilon)}2^{-n(H(X|U)-\epsilon)}2^{-n(H(Y_1|U)-\epsilon)} \tag{15.225}$$

$$= 2^{-n(I(X;Y_1|U)-4\epsilon)}. \tag{15.226}$$

Hence, if $R_1 < I(X; Y_1|U)$, the fourth term in the probability of error goes to 0. Thus, we can bound the probability of error

$$P_e^{(n)}(1) \leq \epsilon + \epsilon + 2^{nR_2}2^{-n(I(U;Y_1)-3\epsilon)} + 2^{nR_1}2^{-n(I(X;Y_1|U)-4\epsilon)} \tag{15.227}$$

$$\leq 4\epsilon \tag{15.228}$$

if n is large enough and $R_2 < I(U; Y_1)$ and $R_1 < I(X; Y_1|U)$. The above bounds show that we can decode the messages with total probability of error that goes to 0. Hence, there exists a sequence of good $((2^{nR_1}, 2^{nR_2}), n)$ codes $\mathcal{C}_n^*$ with probability of error going to 0. With this, we complete the proof of the achievability of the capacity region for the degraded broadcast channel. Gallager's proof [225] of the converse is outlined in Problem 15.11. □

So far we have considered sending independent information to each receiver. But in certain situations, we wish to send common information to both receivers. Let the rate at which we send common information be R_0. Then we have the following obvious theorem:

Theorem 15.6.3 *If the rate pair* (R_1, R_2) *is achievable for a broadcast channel with independent information, the rate triple* $(R_0, R_1 - R_0, R_2 - R_0)$ *with a common rate* R_0 *is achievable, provided that* $R_0 \leq \min(R_1, R_2)$.

In the case of a degraded broadcast channel, we can do even better. Since by our coding scheme the better receiver always decodes all the information that is sent to the worst receiver, one need not reduce the amount of information sent to the better receiver when we have common information. Hence, we have the following theorem:

Theorem 15.6.4 *If the rate pair* (R_1, R_2) *is achievable for a degraded broadcast channel, the rate triple* $(R_0, R_1, R_2 - R_0)$ *is achievable for the channel with common information, provided that* $R_0 < R_2$.

We end this section by considering the example of the binary symmetric broadcast channel.

Example 15.6.5 Consider a pair of binary symmetric channels with parameters p_1 and p_2 that form a broadcast channel as shown in Figure 15.27. Without loss of generality in the capacity calculation, we can recast this channel as a physically degraded channel. We assume that $p_1 < p_2 < \frac{1}{2}$. Then we can express a binary symmetric channel with parameter p_2 as a cascade of a binary symmetric channel with parameter p_1 with another binary symmetric channel. Let the crossover probability of the new channel be α. Then we must have

$$p_1(1-\alpha) + (1-p_1)\alpha = p_2 \tag{15.229}$$

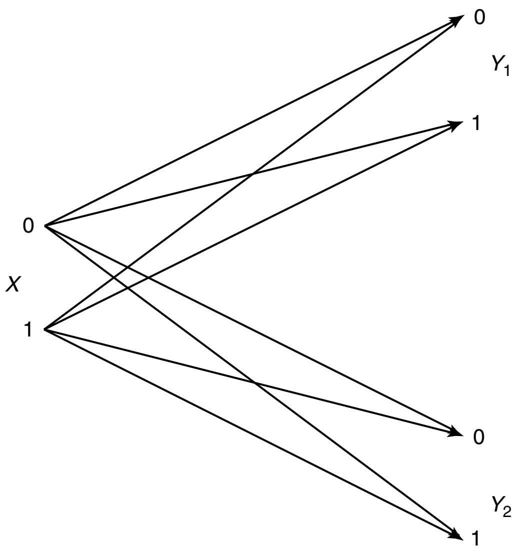


FIGURE 15.27. Binary symmetric broadcast channel.

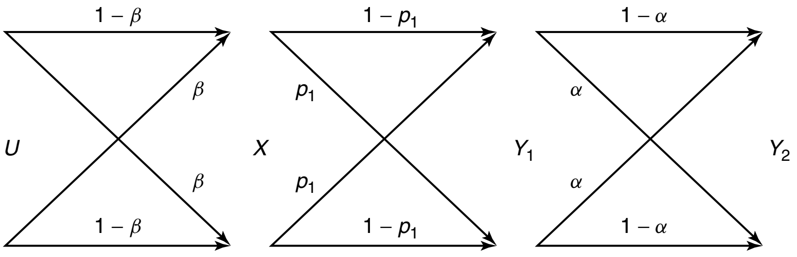


FIGURE 15.28. Physically degraded binary symmetric broadcast channel.

or

$$\alpha = \frac{p_2 - p_1}{1 - 2p_1}. \tag{15.230}$$

We now consider the auxiliary random variable in the definition of the capacity region. In this case, the cardinality of U is binary from the bound of the theorem. By symmetry, we connect U to X by another binary symmetric channel with parameter β, as illustrated in Figure 15.28.

We can now calculate the rates in the capacity region. It is clear by symmetry that the distribution on U that maximizes the rates is the uniform distribution on {0, 1}, so that

$$I(U;Y_2) = H(Y_2) - H(Y_2|U) \tag{15.231}$$

$$= 1 - H(\beta * p_2), \tag{15.232}$$

where

$$\beta * p_2 = \beta(1-p_2) + (1-\beta)p_2. \tag{15.233}$$

Similarly,

$$I(X;Y_1|U) = H(Y_1|U) - H(Y_1|X,U) \tag{15.234}$$

$$= H(Y_1|U) - H(Y_1|X) \tag{15.235}$$

$$= H(\beta * p_1) - H(p_1), \tag{15.236}$$

where

$$\beta * p_1 = \beta(1-p_1) + (1-\beta)p_1. \tag{15.237}$$

Plotting these points as a function of β, we obtain the capacity region in Figure 15.29. When $\beta = 0$, we have maximum information transfer to Y_2 [i.e., $R_2 = 1 - H(p_2)$ and $R_1 = 0$]. When $\beta = \frac{1}{2}$, we have maximum information transfer to Y_1 [i.e., $R_1 = 1 - H(p_1)$] and no information transfer to Y_2. These values of β give us the corner points of the rate region.

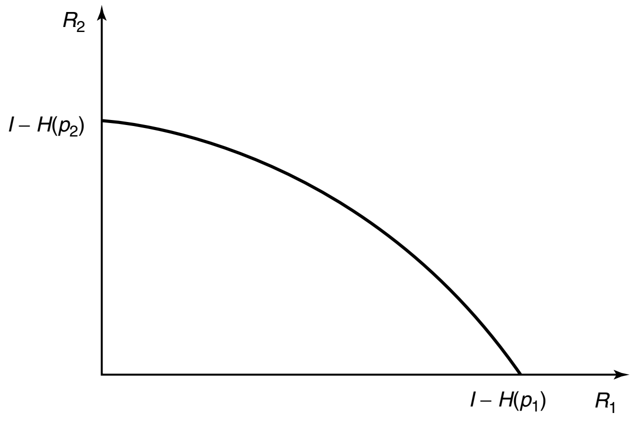


FIGURE 15.29. Capacity region of binary symmetric broadcast channel.

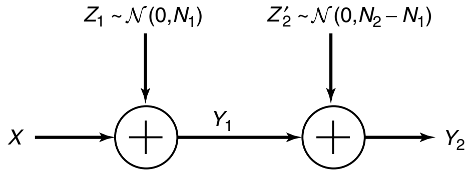

$Z_1 \sim \mathcal{N}(0, N_1)$
$Z_2' \sim \mathcal{N}(0, N_2 - N_1)$
X
Y_1
Y_2

FIGURE 15.30. Gaussian broadcast channel.

Example 15.6.6 (*Gaussian broadcast channel*) The Gaussian broadcast channel is illustrated in Figure 15.30. We have shown it in the case where one output is a degraded version of the other output. Based on the results of Problem 15.10, it follows that all scalar Gaussian broadcast channels are equivalent to this type of degraded channel.

$$Y_1 = X + Z_1, \tag{15.238}$$

$$Y_2 = X + Z_2 = Y_1 + Z_2', \tag{15.239}$$

where $Z_1 \sim \mathcal{N}(0, N_1)$ and $Z_2' \sim \mathcal{N}(0, N_2 - N_1)$.

Extending the results of this section to the Gaussian case, we can show that the capacity region of this channel is given by

$$R_1 < C\left(\frac{\alpha P}{N_1}\right) \tag{15.240}$$

$$R_2 < C\left(\frac{(1-\alpha)P}{\alpha P + N_2}\right), \tag{15.241}$$

where α may be arbitrarily chosen ($0 \le \alpha \le 1$). The coding scheme that achieves this capacity region is outlined in Section 15.1.3.

15.7 RELAY CHANNEL

The relay channel is a channel in which there is one sender and one receiver with a number of intermediate nodes that act as relays to help the communication from the sender to the receiver. The simplest relay channel has only one intermediate or relay node. In this case the channel consists of four finite sets $\mathcal{X}$, $\mathcal{X}_1$, $\mathcal{Y}$, and $\mathcal{Y}_1$ and a collection of probability mass functions $p(y, y_1|x, x_1)$ on $\mathcal{Y} \times \mathcal{Y}_1$, one for each $(x, x_1) \in \mathcal{X} \times \mathcal{X}_1$. The interpretation is that x is the input to the channel and y is the output of the channel, y_1 is the relay's observation, and x_1 is the input symbol chosen by the relay, as shown in Figure 15.31. The problem is to find the capacity of the channel between the sender X and the receiver Y.

The relay channel combines a broadcast channel (X to Y and Y_1) and a multiple-access channel (X and X_1 to Y). The capacity is known for the special case of the physically degraded relay channel. We first prove an outer bound on the capacity of a general relay channel and later establish an achievable region for the degraded relay channel.

Definition A $(2^{nR}, n)$ code for a relay channel consists of a set of integers $\mathcal{W} = \{1, 2, \ldots, 2^{nR}\}$, an encoding function

$$X : \{1, 2, \ldots, 2^{nR}\} \to \mathcal{X}^n, \tag{15.242}$$

a set of relay functions $\{f_i\}_{i=1}^n$ such that

$$x_{1i} = f_i(Y_{11}, Y_{12}, \ldots, Y_{1i-1}), \qquad 1 \le i \le n, \tag{15.243}$$

and a decoding function,

$$g : \mathcal{Y}^n \to \{1, 2, \ldots, 2^{nR}\}. \tag{15.244}$$

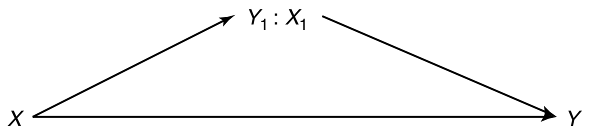


FIGURE 15.31. Relay channel.

Note that the definition of the encoding functions includes the nonanticipatory condition on the relay. The relay channel input is allowed to depend only on the past observations $y_{11}, y_{12}, \ldots, y_{1i-1}$. The channel is memoryless in the sense that (Y_i, Y_{1i}) depends on the past only through the current transmitted symbols (X_i, X_{1i}). Thus, for any choice $p(w)$, $w \in \mathcal{W}$, and code choice $X : \{1, 2, \ldots, 2^{nR}\} \to \mathcal{X}_i^n$ and relay functions $\{f_i\}_{i=1}^n$, the joint probability mass function on $\mathcal{W} \times \mathcal{X}^n \times \mathcal{X}_1^n \times \mathcal{Y}^n \times \mathcal{Y}_1^n$ is given by

$$p(w, \mathbf{x}, \mathbf{x}_1, \mathbf{y}, \mathbf{y}_1) = p(w) \prod_{i=1}^{n} p(x_i|w) p(x_{1i}|y_{11}, y_{12}, \ldots, y_{1i-1}) \times p(y_i, y_{1i}|x_i, x_{1i}). \tag{15.245}$$

If the message $w \in [1, 2^{nR}]$ is sent, let

$$\lambda(w) = \Pr\{g(\mathbf{Y}) \neq w | w \text{ sent}\} \tag{15.246}$$

denote the conditional probability of error. We define the average probability of error of the code as

$$P_e^{(n)} = \frac{1}{2^{nR}} \sum_w \lambda(w). \tag{15.247}$$

The probability of error is calculated under the uniform distribution over the codewords $w \in \{1, \ldots, 2^{nR}\}$. The rate R is said to be *achievable* by the relay channel if there exists a sequence of $(2^{nR}, n)$ codes with $P_e^{(n)} \to 0$. The *capacity* C of a relay channel is the supremum of the set of achievable rates.

We first give an upper bound on the capacity of the relay channel.

Theorem 15.7.1 *For any relay channel $(\mathcal{X} \times \mathcal{X}_1, p(y, y_1|x, x_1), \mathcal{Y} \times \mathcal{Y}_1)$, the capacity C is bounded above by*

$$C \le \sup_{p(x, x_1)} \min \{I(X, X_1; Y), I(X; Y, Y_1|X_1)\}. \tag{15.248}$$

Proof: The proof is a direct consequence of a more general max-flow min-cut theorem given in Section 15.10. □

This upper bound has a nice max-flow min-cut interpretation. The first term in (15.248) upper bounds the maximum rate of information transfer from senders X and X_1 to receiver Y. The second terms bound the rate from X to Y and Y_1.

We now consider a family of relay channels in which the relay receiver is better than the ultimate receiver Y in the sense defined below. Here the max-flow min-cut upper bound in the (15.248) is achieved.

Definition The relay channel $(\mathcal{X} \times \mathcal{X}_1, p(y, y_1|x, x_1), \mathcal{Y} \times \mathcal{Y}_1)$ is said to be *physically degraded* if $p(y, y_1|x, x_1)$ can be written in the form

$$p(y, y_1|x, x_1) = p(y_1|x, x_1)p(y|y_1, x_1). \tag{15.249}$$

Thus, Y is a random degradation of the relay signal Y_1.

For the physically degraded relay channel, the capacity is given by the following theorem.

Theorem 15.7.2 *The capacity C of a physically degraded relay channel is given by*

$$C = \sup_{p(x,x_1)} \min \{I(X, X_1; Y), I(X; Y_1|X_1)\}, \tag{15.250}$$

where the supremum is over all joint distributions on $\mathcal{X} \times \mathcal{X}_1$.

Proof:

Converse: The proof follows from Theorem 15.7.1 and by degradedness, since for the degraded relay channel, $I(X; Y, Y_1|X_1) = I(X; Y_1|X_1)$.

Achievability: The proof of achievability involves a combination of the following basic techniques: (1) random coding, (2) list codes, (3) Slepian–Wolf partitioning, (4) coding for the cooperative multiple-access channel, (5) superposition coding, and (6) block Markov encoding at the relay and transmitter. We provide only an outline of the proof.

Outline of achievability: We consider B blocks of transmission, each of n symbols. A sequence of $B - 1$ indices, $w_i \in \{1, \ldots, 2^{nR}\}$, $i = 1, 2, \ldots, B - 1$, will be sent over the channel in nB transmissions. (Note that as $B \to \infty$ for a fixed n, the rate $R(B - 1)/B$ is arbitrarily close to R.)

We define a doubly indexed set of codewords:

$$\mathcal{C} = \{\mathbf{x}(w|s), \mathbf{x}_1(s)\} : w \in \{1, 2^{nR}\}, s \in \{1, 2^{nR_0}\}, \mathbf{x} \in \mathcal{X}^n, \mathbf{x}_1 \in \mathcal{X}_1^n. \tag{15.251}$$

We will also need a partition

$$\mathcal{S} = \{S_1, S_2, \ldots, S_{2^{nR_0}}\} \text{ of } \mathcal{W} = \{1, 2, \ldots, 2^{nR}\} \tag{15.252}$$

into 2^{nR_0} cells, with $S_i \cap S_j = \phi, i \neq j$, and $\cup S_i = \mathcal{W}$. The partition will enable us to send side information to the receiver in the manner of Slepian and Wolf [502].

Generation of random code: Fix $p(x_1)p(x|x_1)$.

First generate at random 2^{nR_0} i.i.d. n-sequences in $\mathcal{X}_1^n$, each drawn according to $p(\mathbf{x}_1) = \prod_{i=1}^n p(x_{1i})$. Index them as $\mathbf{x}_1(s), s \in \{1, 2, \ldots, 2^{nR_0}\}$. For each $\mathbf{x}_1(s)$, generate 2^{nR} conditionally independent n-sequences $\mathbf{x}(w|s), w \in \{1, 2, \ldots, 2^{nR}\}$, drawn independently according to $p(\mathbf{x}|\mathbf{x}_1(s)) = \prod_{i=1}^n p(x_i|x_{1i}(s))$. This defines the random codebook $\mathcal{C} = \{\mathbf{x}(w|s), \mathbf{x}_1(s)\}$. The random partition $\mathcal{S} = \{S_1, S_2, \ldots, S_{2^{nR_0}}\}$ of $\{1, 2, \ldots, 2^{nR}\}$ is defined as follows. Let each integer $w \in \{1, 2, \ldots, 2^{nR}\}$ be assigned independently, according to a uniform distribution over the indices $s = 1, 2, \ldots, 2^{nR_0}$, to cells S_s.

Encoding: Let $w_i \in \{1, 2, \ldots, 2^{nR}\}$ be the new index to be sent in block i, and let s_i be defined as the partition corresponding to w_{i-1} (i.e., $w_{i-1} \in S_{s_i}$). The encoder sends $\mathbf{x}(w_i|s_i)$. The relay has an estimate $\hat{\hat{w}}_{i-1}$ of the previous index w_{i-1}. (This will be made precise in the decoding section.) Assume that $\hat{\hat{w}}_{i-1} \in S_{\hat{\hat{s}}_i}$. The relay encoder sends $\mathbf{x}_1(\hat{\hat{s}}_i)$ in block i.

Decoding: We assume that at the end of block $i-1$, the receiver knows $(w_1, w_2, \ldots, w_{i-2})$ and $(s_1, s_2, \ldots, s_{i-1})$ and the relay knows $(w_1, w_2, \ldots, w_{i-1})$ and consequently, $(s_1, s_2, \ldots, s_i)$. The decoding procedures at the end of block i are as follows:

1. Knowing s_i and upon receiving $\mathbf{y}_1(i)$, the *relay* receiver estimates the message of the transmitter $\hat{\hat{w}}_i = w$ if and only if there exists a unique w such that $(\mathbf{x}(w|s_i), \mathbf{x}_1(s_i), \mathbf{y}_1(i))$ are jointly ϵ-typical. Using Theorem 15.2.3, it can be shown that $\hat{\hat{w}}_i = w_i$ with an arbitrarily small probability of error if
$$R < I(X; Y_1|X_1) \tag{15.253}$$
and n is sufficiently large.
2. The *receiver* declares that $\hat{s}_i = s$ was sent iff there exists one and only one s such that $(\mathbf{x}_1(s), \mathbf{y}(i))$ are jointly ϵ-typical. From Theorem 15.2.1 we know that s_i can be decoded with arbitrarily small probability of error if
$$R_0 < I(X_1; Y) \tag{15.254}$$
and n is sufficiently large.