Chapter 7

Processes

The model of a communication system that we have been developing is shown in Figure 7.1, where the source is assumed to emit a stream of symbols. The channel may be a physical channel between different points in space, or it may be a memory which stores information for retrieval at a later time, or it may be a computation in which the information is processed in some way.

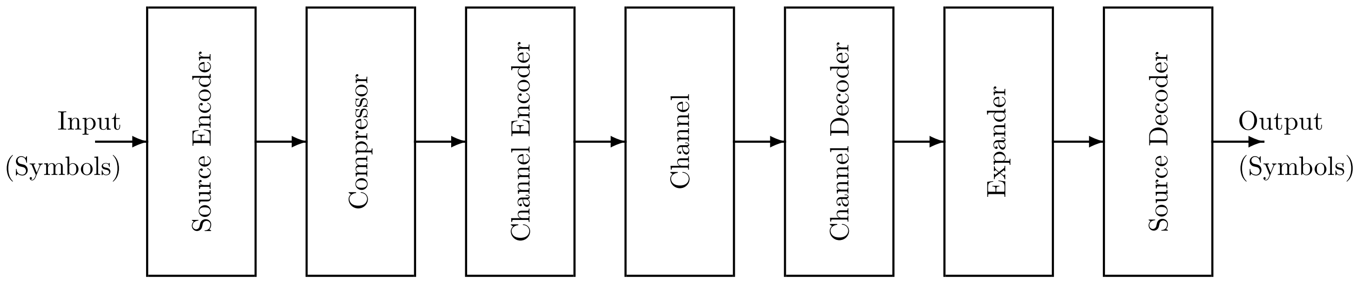


Figure 7.1: Elaborate communication system

Figure 7.1 shows the module inputs and outputs and how they are connected. A diagram like this is very useful in portraying an overview of the operation of a system, but other representations are needed for more detail. In this chapter we develop two abstract models that are general enough to represent each of these boxes in Figure 7.1, but show the flow of information quantitatively.

Because each of these boxes in Figure 7.1 processes information in some way, it is called a **processor** and what it does is called a **process**. The processes we consider here are

- **Discrete:** The inputs are members of a set of mutually exclusive possibilities, only one of which occurs at a time, and the output is one of another discrete set of mutually exclusive events.
- **Finite:** The set of possible inputs is finite in number, as is the set of possible outputs.
- **Memoryless:** The process acts on the input at some time and produces an output based on that input and not on any prior inputs.

- **Nondeterministic:** The process may produce a different output when presented with the same input a second time (the model is also valid for deterministic processes). With nondeterministic processes the output contains random **noise**.
- **Nontransparent:** It may not be possible to "see" the input from the output, i.e., determine the input by observing the output. The model is also valid for transparent processes. Nontransparent processes are called **lossy** because knowledge about the input is lost by the time the output is created.

7.1 Types of Process Diagrams

Different diagrams of processes are useful for different purposes. The four we use here are all **recursive**, meaning that a process may be represented in terms of other, more detailed processes of the same sort, interconnected. Conversely, two or more connected processes may be represented by a single higher-level process with some of the detailed information suppressed. The processes represented can be either deterministic (noiseless) or nondeterministic (noisy), and either transparent (lossless) or nontransparent (lossy).

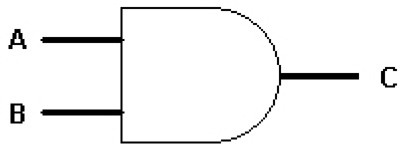


Figure 7.2: Circuit diagram of an *AND* gate

- **Block Diagram:** Figure 7.1 is a block diagram. It shows how the processes are connected, but very little about how the processes achieve their purposes, or how the connections are made. It is useful in viewing the system at a high level. An interconnection in a block diagram can represent many bits.
- **Logic Circuit:** If the system is made of logic gates, a useful diagram is one showing such gates interconnected. For example, Figure 7.2 is an *AND* gate. Each input and output represents a wire with a single logic value, with, for example, a high voltage representing 1 and a low voltage 0. The number of possible bit patterns of a logic gate is greater than the number of physical wires; each wire could have two possible voltages, so for n-input gates there would be 2^n input states.
- **Probability Diagram:** A process with n single-bit inputs and m single-bit outputs can be modeled by the probabilities relating the 2^n possible input bit patterns and the 2^m possible output patterns. For example, Figure 7.3 shows a gate with two inputs (four bit patterns) and one output. An example of such a gate is the *AND* gate, and its probability model is shown in Figure 7.4. Probability diagrams are discussed further in Section 7.2.
- **Information Diagram:** A diagram that shows explicitly the information flow between processes is useful. In order to handle processes with noise or loss, the information associated with them can be shown. Information diagrams are discussed further in Section 7.6.

7.2 Probability Diagrams

The probability model of a process with n inputs and m outputs, where n and m are integers, is shown in Figure 7.5. The n input states are mutually exclusive, as are the m output states. If this process is implemented by logic gates the input would need at least $\log_2(n)$ wires but not as many as n wires.

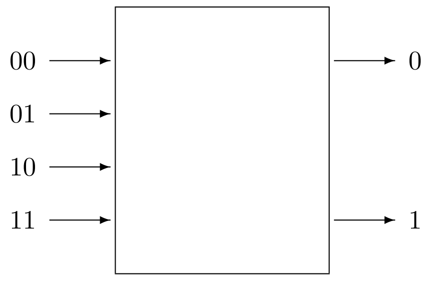


Figure 7.3: Probability model of a two-input one-output gate.

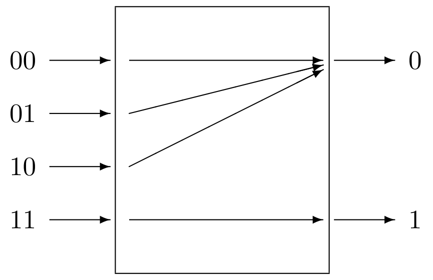


Figure 7.4: Probability model of an *AND* gate

This model for processes is conceptually simple and general. It works well for processes with a small number of bits. It was used for the binary channel in Chapter 6.

Unfortunately, the probability model is awkward for practical calculations. The reason is that the inputs and outputs are represented in terms of mutually exclusive sets of events. If the events describe signals on, say, five wires, each of which can carry a high or low voltage signifying a boolean 1 or 0, there would be 32 possible events. It is much easier to draw a logic gate with five inputs representing physical variables, than a probability process with 32 input states. This "exponential explosion" of the number of possible input states gets even more severe when the process represents the evolution of the state of a physical system with a large number of atoms. For example, the number of molecules in a mole of gas is Avogadro's number $N_A = 6.02 \times 10^{23}$. If each atom had just one associated boolean variable, there would be 2^{N_A} states, far greater than the number of particles in the universe. And there would not be time to even list all the particles, much less do any calculations: the number of microseconds since the big bang is less than 5×10^{23}.

The probability model, introduced in Chapter 6, is reviewed in the next few paragraphs.

We assume that each possible input state of a process can lead to one or more output state. For each input i denote the probability that this input leads to the output j as c_{ji}. These **transition probabilities** c_{ji} can be thought of as a table, or matrix, with as many columns as there are input states, and as many

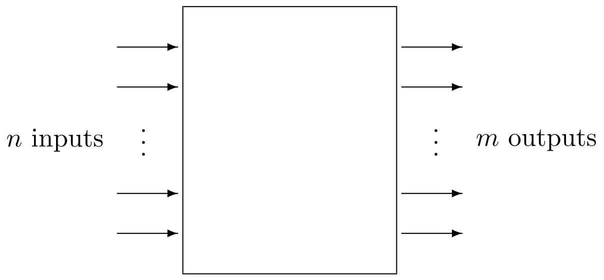


Figure 7.5: Probability model

rows as output states. We will use i as an index over the input states and j over the output states, and denote the event associated with the selection of input i as A_i and the event associated with output j as B_j.

The transition probabilities are properties of the process, and do not depend on the process inputs. The transition probabilities lie between 0 and 1 and, for each i their sum over the output index j is 1, since for each possible input event exactly one output event happens. If the number of input states is the same as the number of output states then c_{ji} is a square matrix; otherwise it has more columns than rows or vice versa.

$$0 \le c_{ji} \le 1 \tag{7.1}$$

$$1 = \sum_j c_{ji} \tag{7.2}$$

This description has great generality. It applies to a deterministic process (although it may not be the most convenient to use). For such a process, each column of the c_{ji} matrix contains one element that is 1 and all the other elements are 0. It also applies to a channel with noise. It applies to the source encoder and decoder, to the compressor and expander, and to the channel encoder and decoder. It applies to logic gates and to devices which perform arbitrary memoryless computation (sometimes called "combinational logic" in distinction to "sequential logic" which can involve prior states). It even applies to transitions taken by a physical system from one of its states to the next. It applies if the number of output states is greater than the number of input states (for example channel encoders) or less (for example channel decoders).

If a process input is determined by random events A_i with probability distribution $p(A_i)$ then the various other probabilities can be calculated. The conditional output probabilities, conditioned on the input, are

$$p(B_j \mid A_i) = c_{ji} \tag{7.3}$$

The unconditional probability of each output $p(B_j)$ is

$$p(B_j) = \sum_i c_{ji} p(A_i) \tag{7.4}$$

Finally, the joint probability of each input with each output $p(A_i, B_j)$ and the backward conditional probabilities $p(A_i \mid B_j)$ can be found using Bayes' Theorem:

$$\begin{aligned} p(A_i, B_j) &= p(B_j)p(A_i \mid B_j) && (7.5)\\ &= p(A_i)p(B_j \mid A_i) && (7.6)\\ &= p(A_i)c_{ji} && (7.7) \end{aligned}$$

7.2.1 Example: *AND* Gate

The *AND* gate is deterministic (it has no noise) but is lossy, because knowledge of the output is not sufficient to infer the input. The transition matrix is

$$\begin{bmatrix} c_{0(00)} & c_{0(01)} & c_{0(10)} & c_{0(11)} \\ c_{1(00)} & c_{1(01)} & c_{1(10)} & c_{1(11)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.8}$$

The probability model for this gate is shown in Figure 7.4.

7.2.2 Example: Binary Channel

The binary channel is well described by the probability model. Its properties, many of which were discussed in Chapter 6, are summarized below.

Consider first a noiseless binary channel which, when presented with one of two possible input values 0 or 1, transmits this value faithfully to its output. This is a very simple example of a discrete memoryless

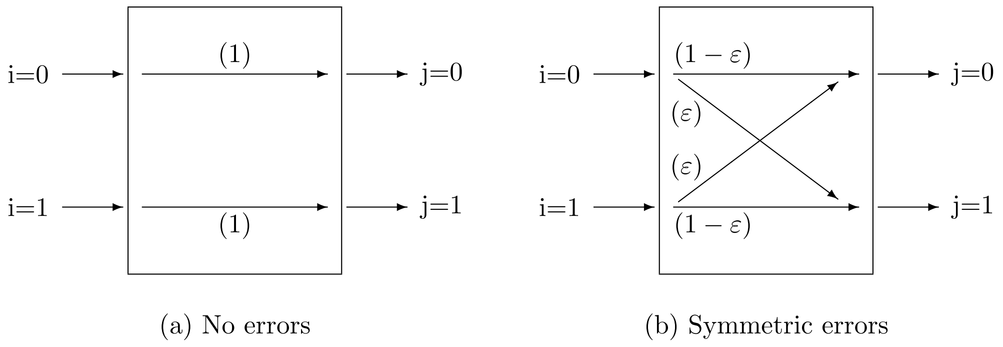


Figure 7.6: Probability models for error-free binary channel and symmetric binary channel

process. We represent this channel by a probability model with two inputs and two outputs. To indicate the fact that the input is replicated faithfully at the output, the inner workings of the box are revealed in the form of two paths, one from each input to the corresponding output. See Figure 7.6(a). The transition matrix for this channel is

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.9}$$

The input information I for this process is 1 bit if the two values are equally likely; more generally

$$I = p(A_0)\log_2\left(\frac{1}{p(A_0)}\right) + p(A_1)\log_2\left(\frac{1}{p(A_1)}\right) \tag{7.10}$$

The output information J has a similar formula, using the output probabilities $p(B_j)$. Since the input and output are the same in this case, it is possible to infer the input when the output has been observed. The amount of information out is the same as the amount in: $J = I$. This noiseless channel is effective for its intended purpose, which is to permit the receiver, at the output, to infer the value at the input.

Next, let us suppose that this channel occasionally makes errors. Thus if the input is 1 the output is not always 1, but with the "bit error probability" ε is flipped to the "wrong" value 0, and hence is "correct" only with probability $1 - \varepsilon$. Similarly, for the input of 0, the probability of error is ε. Then the transition matrix is

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{bmatrix} \tag{7.11}$$

This model, with random behavior, is sometimes called the Symmetric Binary Channel (SBC), symmetric in the sense that the errors in the two directions (from 0 to 1 and vice versa) are equally likely. This channel is pictured in Figure 7.6(b), with two paths leaving from each input, and two paths converging on each output.

Clearly the errors in the SBC introduce some uncertainty into the output over and above the uncertainty that is present in the input signal. Intuitively, we can say that noise has been added, so that the output is composed in part of desired signal and in part of noise. Or we can say that some of our information is lost in the channel. Both of these effects have happened, but as we will see they are not always related; processes may introduce noise but have no loss, or vice versa. In Section 7.3 we will find it is possible to calculate the amount of information lost or gained because of noise or loss, in bits.

Loss of information happens because it is no longer possible to tell with certainty what the input signal is, after the output is observed. Loss shows up in drawings like Figure 7.6(b) where two or more paths converge on the same output. Noise happens because the output is not determined precisely by the input. Noise shows up in drawings like Figure 7.6(b) where two or more paths diverge from the same input. Despite noise and loss, however, some information can be transmitted from the input to the output (i.e., observation of the output can allow one to make some inferences about the input).

We now return to our model of a general discrete memoryless nondeterministic lossy process, and derive formulas for noise, loss, and information transfer (which will be called "mutual information"). We will then come back to the symmetric binary channel and interpret these formulas.

7.3 Information, Loss, and Noise

For the general discrete memoryless process, useful measures of the amount of information presented at the input and the amount transmitted to the output can be defined. We suppose the process state is represented by random events A_i with probability distribution $p(A_i)$. The information at the input I is the same as the entropy of this source. (We have chosen to use the letter I for input information not because it stands for "input" or "information" but rather for the index i that goes over the input probability distribution. The output information will be denoted J for a similar reason.)

$$I = \sum_i p(A_i) \log_2 \left(\frac{1}{p(A_i)} \right) \tag{7.12}$$

This is the amount of uncertainty we have about the input if we do not know what it is, or before it has been selected by the source.

A similar formula applies at the output. Thus, expressing the result in terms of the input probability distribution and the channel transition matrix,

$$\begin{aligned} J &= \sum_j p(B_j) \log_2 \left(\frac{1}{p(B_j)} \right) \\ &= \sum_j \left(\sum_i c_{ji} p(A_i) \right) \log_2 \left(\frac{1}{\sum_i c_{ji} p(A_i)} \right) \end{aligned} \tag{7.13}$$

This measure of information at the output, however, does not refer to the identity of the input state, but rather the output state. It represents our uncertainty about the output state before we discover what it is. If our objective is to determine the input, J is not relevant. Instead, we should ask about the uncertainty of our knowledge of the input state. This can be expressed from the vantage point of the output by asking about the uncertainty of the input state given one particular output state, and then averaging over those states. This uncertainty, for each j, is given by a formula like those above but using the reverse conditional probabilities $p(A_i \mid B_j)$

$$\sum_i p(A_i \mid B_j) \log_2 \left(\frac{1}{p(A_i \mid B_j)} \right) \tag{7.14}$$

Then your average uncertainty about the input after learning the output is found by computing the average over the output probability distribution, i.e., by multiplying by $p(B_j)$ and summing over j

$$\begin{aligned} L &= \sum_j p(B_j) \sum_i p(A_i \mid B_j) \log_2 \left(\frac{1}{p(A_i \mid B_j)} \right) \\ &= \sum_{ij} p(A_i, B_j) \log_2 \left(\frac{1}{p(A_i \mid B_j)} \right) \end{aligned} \tag{7.15}$$

Note that the second formula uses the joint probability distribution $p(A_i, B_j)$. We have denoted this average uncertainty by L and will call it "loss." This term is appropriate because it is the amount of information about the input that is not able to be determined by examining the output state, and so got "lost" in the transition from input to output. In the special case that the process allows the input state to be identified uniquely for each possible output state, the process is "lossless" and, as would be expected, $L = 0$.

It was proved in Chapter 6 that $L \leq I$ or, in words, that the uncertainty after learning the output is less than (or perhaps equal to) the uncertainty before. This result was proved using the Gibbs inequality.

The amount of information we learn about the input state upon being told the output state is our uncertainty before being told, which is I, less our uncertainty after being told, which is L. We have just shown that this amount cannot be negative, since $L \leq I$. As was done in Chapter 6, we denote the amount we have learned as $M = I - L$, and call this the "mutual information" between input and output. This is an important quantity because it is the amount of information that gets through the process.

To recapitulate the relations among these information quantities:

$$I = \sum_i p(A_i) \log_2 \left(\frac{1}{p(A_i)} \right) \tag{7.16}$$

$$L = \sum_j p(B_j) \sum_i p(A_i \mid B_j) \log_2 \left(\frac{1}{p(A_i \mid B_j)} \right) \tag{7.17}$$

$$M = I - L \tag{7.18}$$

$$0 \leq M \leq I \tag{7.19}$$

$$0 \leq L \leq I \tag{7.20}$$

Processes with outputs that can be produced by more than one input have loss. These processes may also be nondeterministic, in the sense that one input state can lead to more than one output state. The symmetric binary channel with loss is an example of a process that has loss and is also nondeterministic. However, there are some processes that have loss but are deterministic. An example is the *AND* logic gate, which has four mutually exclusive inputs 00 01 10 11 and two outputs 0 and 1. Three of the four inputs lead to the output 0. This gate has loss but is perfectly deterministic because each input state leads to exactly one output state. The fact that there is loss means that the *AND* gate is not reversible.

There is a quantity similar to L that characterizes a nondeterministic process, whether or not it has loss. The output of a nondeterministic process contains variations that cannot be predicted from knowing the input, that behave like noise in audio systems. We will define the noise N of a process as the uncertainty in the output, given the input state, averaged over all input states. It is very similar to the definition of loss, but with the roles of input and output reversed. Thus

$$\begin{aligned} N &= \sum_i p(A_i) \sum_j p(B_j \mid A_i) \log_2 \left(\frac{1}{p(B_j \mid A_i)} \right) \\ &= \sum_i p(A_i) \sum_j c_{ji} \log_2 \left(\frac{1}{c_{ji}} \right) \end{aligned} \tag{7.21}$$

Steps similar to those above for loss show analogous results. What may not be obvious, but can be proven easily, is that the mutual information M plays exactly the same sort of role for noise as it does for loss. The formulas relating noise to other information measures are like those for loss above, where the mutual information M is the same:

$$J = \sum_i p(B_j) \log_2 \left(\frac{1}{p(B_j)} \right) \tag{7.22}$$

$$N = \sum_i p(A_i) \sum_j c_{ji} \log_2 \left(\frac{1}{c_{ji}} \right) \tag{7.23}$$

$$M = J - N \tag{7.24}$$

$$0 \le M \le J \tag{7.25}$$

$$0 \le N \le J \tag{7.26}$$

It follows from these results that

$$J - I = N - L \tag{7.27}$$

7.3.1 Example: Symmetric Binary Channel

For the SBC with bit error probability ε, these formulas can be evaluated, even if the two input probabilities $p(A_0)$ and $p(A_1)$ are not equal. If they happen to be equal (each 0.5), then the various information measures for the SBC in bits are particularly simple:

$$I = J = 1 \text{ bit} \tag{7.28}$$

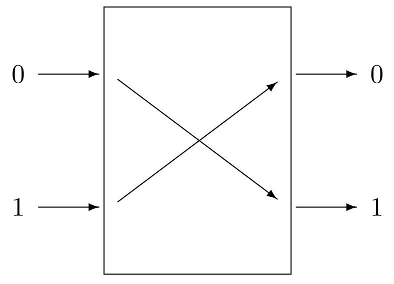
$$L = N = \varepsilon \log_2 \left(\frac{1}{\varepsilon} \right) + (1 - \varepsilon) \log_2 \left(\frac{1}{1 - \varepsilon} \right) \tag{7.29}$$

$$M = 1 - \varepsilon \log_2 \left(\frac{1}{\varepsilon} \right) - (1 - \varepsilon) \log_2 \left(\frac{1}{1 - \varepsilon} \right) \tag{7.30}$$

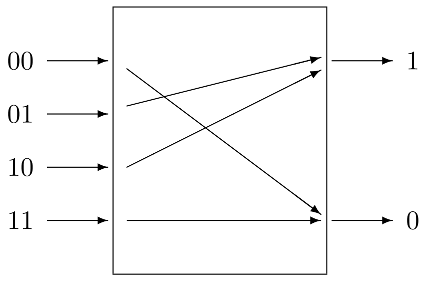
The errors in the channel have destroyed some of the information, in the sense that they have prevented an observer at the output from knowing with certainty what the input is. They have thereby permitted only the amount of information $M = I - L$ to be passed through the channel to the output.

7.4 Deterministic Examples

This probability model applies to any system with mutually exclusive inputs and outputs, whether or not the transitions are random. If all the transition probabilities c_{ji} are equal to either 0 or 1, then the process is deterministic.



(a) *NOT* gate



(b) *XOR* gate

Figure 7.7: Probability models of deterministic gates

A simple example of a deterministic process is the *NOT* gate, which implements Boolean negation. If the input is 1 the output is 0 and vice versa. The input and output information are the same, $I = J$ and there is no noise or loss: $N = L = 0$. The information that gets through the gate is $M = I$. See Figure 7.7(a).

A slightly more complex deterministic process is the exclusive or, *XOR* gate. This is a Boolean function of two input variables and therefore there are four possible input values. When the gate is represented by a circuit diagram, there are two input wires representing the two inputs. When the gate is represented as a discrete process using a probability diagram like Figure 7.6(b), there are four mutually exclusive inputs and two mutually exclusive outputs. If the probabilities of the four inputs are each 0.25, then $I = 2$ bits, and the two output probabilities are each 0.5 so $J = 1$ bit. There is therefore 1 bit of loss, and the mutual information is 1 bit. The loss arises from the fact that two different inputs produce the same output; for example if the output 1 is observed the input could be either 01 or 10. There is no noise introduced into the output because each of the transition parameters is either 0 or 1, i.e., there are no inputs with multiple transition paths coming from them.

Other, more complex logic functions can be represented in similar ways. However, for logic functions with n physical inputs, the discrete process representation is awkward if n is larger than 3 or 4 because the number of inputs is 2^n.

7.4.1 Error Correcting Example

The Hamming Code encoder and decoder can be represented as discrete processes in this form. Consider the (3, 1, 3) code, otherwise known as triple redundancy. The encoder has one 1-bit input (2 values) and a 3-bit output (8 values). The input 1 is wired directly to the output 111 and the input 0 to the output 000. The other six outputs are not connected, and therefore occur with probability 0. See Figure 7.8. The encoder has $N = 0$, $L = 0$, and $M = I = J$. Note that the output information is not three bits even though three physical bits are used to represent it, because of the intentional redundancy.

The output of the triple redundancy encoder is intended to be passed through a channel with the possibility of a single bit error in each block of 3 bits. This noisy channel can be modelled as a nondeterministic process with 8 inputs and 8 outputs (not shown). Each of the 8 inputs is connected with a high-probability connection to the corresponding output, and with low probability connections to the three other values separated by Hamming distance 1 – for example the input 000 is connected only to the outputs 000 (with high probability) and 001, 010, and 100 each with low probability. This channel introduces noise since there are multiple paths coming from each input. In general, when driven with arbitrary bit patterns, there is also loss. However, when driven from the encoder of Figure 7.8, the loss is 0 bits because only two of the eight bit patterns have nonzero probability. The input information to the noisy channel is 1 bit and the output information is greater than 1 bit because of the added noise. This example demonstrates that the value of both noise and loss depend on both the physics of the channel and the probabilities of the input signal.

The decoder, used to recover the signal originally put into the encoder, is shown in Figure 7.8(b). The transition parameters are straightforward – each input is connected to only one output. The decoder has loss (since multiple paths converge on each of the ouputs) but no noise (since each input goes to only one output).

7.5 Capacity

In Chapter 6 of these notes, the channel capacity was defined. This concept can be generalized to other processes.

Call W the maximum rate at which the input state of the process can be detected at the output. Then the rate at which information flows through the process can be as large as WM. However, this product depends on the input probability distribution $p(A_i)$ and hence is not a property of the process itself, but on how it is used. A better definition of process capacity is found by looking at how M can vary with different input probability distributions. Select the largest mutual information discovered, and call that M_{max}. Then the **process capacity** C is defined as

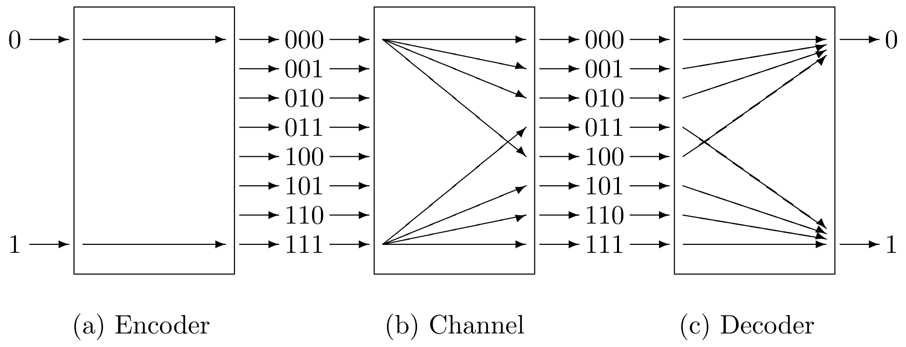


Figure 7.8: Triple redundancy error correction

$$C = WM_{max} \tag{7.31}$$

It is easy to see that M_{max} cannot be arbitrarily large, since $M \leq I$ and $I \leq \log_2 n$ where n is the number of distinct input states.

In the example of symmetric binary channels, it is not difficult to show that the probability distribution that maximizes M is the one with equal probability for each of the two input states.

7.6 Information Diagrams

An information diagram is a connection of processes showing the amount of information passing between them. It is a useful way of representing the input, output, and mutual information and the noise and loss.

It has been shown that all five information measures, I, J, L, N, and M are nonnegative. It is not necessary that L and N be the same, although they are for the symmetric binary channel with inputs with equal probabilities for 0 and 1. It is possible to have processes with loss but no noise (e.g., the XOR gate), or noise but no loss (e.g., the noisy channel for triple redundancy).

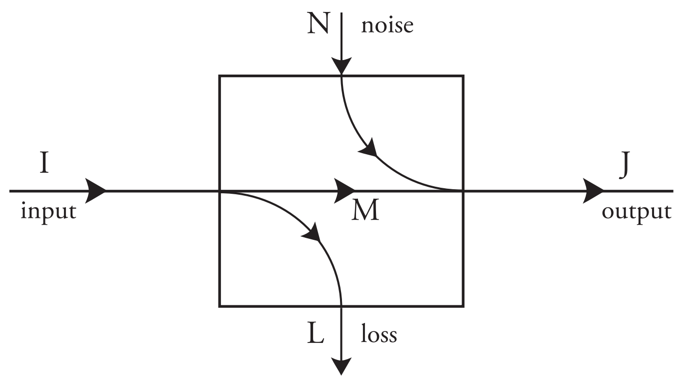


Figure 7.9: Information flow in a discrete memoryless process

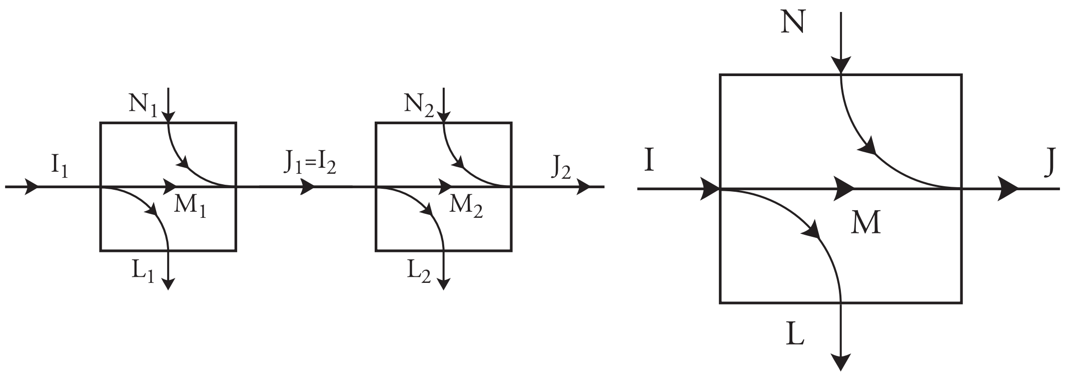
It is convenient to think of information as a physical quantity that is transmitted through this process much the way physical material may be processed in a production line. The material being produced comes in to the manufacturing area, and some is lost due to errors or other causes, some contamination may be added (like noise) and the output quantity is the input quantity, less the loss, plus the noise. The useful product is the input minus the loss, or alternatively the output minus the noise. The flow of information through a discrete memoryless process is shown using this paradigm in Figure 7.9.

An interesting question arises. Probabilities depend on your current state of knowledge, and one observer's knowledge may be different from another's. This means that the loss, the noise, and the information transmitted are all observer-dependent. Is it OK that important engineering quantities like noise and loss depend on who you are and what you know? If you happen to know something about the input that your colleague does not, is it OK for your design of a nondeterministic process to be different, and to take advantage of your knowledge? This question is something to think about; there are times when your knowledge, if correct, can be very valuable in simplifying designs, but there are other times when it is prudent to design using some worst-case assumption of input probabilities so that in case the input does not conform to your assumed probabilities your design still works.

Information diagrams are not often used for communication systems. There is usually no need to account for the noise sources or what happens to the lost information. However, such diagrams are useful in domains where noise and loss cannot occur. One example is reversible computing, a style of computation in which the entire process can, in principle, be run backwards. Another example is quantum communications, where information cannot be discarded without affecting the environment.

7.7 Cascaded Processes

It is interesting to consider two processes in **cascade**. This term refers to having the output from one process serve as the input to another process. Then the two cascaded processes behave just like one larger process, with the states between the two hidden. We have seen that discrete memoryless processes are characterized by values of I, J, L, N, and M. Figure 7.10(a) shows a cascaded pair of processes, each characterized by its own parameters. Of course the parameters of the second process depend on the input probabilities it encounters, which are determined by the transition probabilities (and input probabilities) of the first process.



(a) The two processes (b) Equivalent single process

Figure 7.10: Cascade of two discrete memoryless processes

But the cascade of the two processes is itself a discrete memoryless process and therefore should have its own five parameters, as suggested in Figure 7.10(b). The parameters of the overall model can be calculated either of two ways. First, the transition probabilities of the overall process can be found from the transition probabilities of the two models that are connected together; in fact the matrix of transition probabilities is merely the matrix product of the two transition probability matrices for process 1 and process 2. All the parameters can be calculated from this matrix and the input probabilities .

The other approach is to seek formulas for I, J, L, N, and M of the overall process in terms of the corresponding quantities for the component processes. Unfortunately this approach does not generally work exactly, but it does provide upper and/or lower bounds of performance and is therefore useful in providing insight into the operation of the cascade. Using the notation in the figure above, it can be easily shown that

$$L - N = (L_1 + L_2) - (N_1 + N_2) \tag{7.32}$$

It is then straightforward (though perhaps tedious) to show that the loss L for the overall process is not always equal to the sum of the losses for the two components $L_1 + L_2$, but instead

$$0 \leq L_1 \leq L \leq L_1 + L_2 \tag{7.33}$$

so that the loss is bounded from above and below. Also,

$$L_1 + L_2 - N_1 \leq L \leq L_1 + L_2 \tag{7.34}$$

so that if the first process is noise-free then L is exactly $L_1 + L_2$.

There are similar formulas for N in terms of $N_1 + N_2$:

$$0 \leq N_2 \leq N \leq N_1 + N_2 \tag{7.35}$$

$$N_1 + N_2 - L_2 \leq N \leq N_1 + N_2 \tag{7.36}$$

Similar formulas for the mutual information of the cascade M follow from these results:

$$M_1 - L_2 \leq M \leq M_1 \leq I \tag{7.37}$$
$$M_1 - L_2 \leq M \leq M_1 + N_1 - L_2 \tag{7.38}$$
$$M_2 - N_1 \leq M \leq M_2 \leq J \tag{7.39}$$
$$M_2 - N_1 \leq M \leq M_2 + L_2 - N_1 \tag{7.40}$$

Other formulas for M are easily derived from Equation 7.19 applied to the first process and the cascade, and Equation 7.24 applied to the second process and the cascade:

$$\begin{aligned} M &= M_1 + L_1 - L \\ &= M_1 + N_1 + N_2 - N - L_2 \\ &= M_2 + N_2 - N \\ &= M_2 + L_2 + L_1 - L - N_1 \end{aligned} \tag{7.41}$$

where the second formula in each case comes from the use of Equation 7.32.

Note that M cannot exceed either M_1 or M_2, i.e., $M \leq M_1$ and $M \leq M_2$. This is consistent with the interpretation of M as the information that gets through—information that gets through the cascade must be able to get through the first process and also through the second process.

As a special case, if the second process is lossless, $L_2 = 0$ and then $M = M_1$. In that case, the second process does not lower the mutual information below that of the first process. Similarly if the first process is noiseless, then $N_1 = 0$ and $M = M_2$.

The channel capacity C of the cascade is, similarly, no greater than either the channel capacity of the first process or that of the second process: $C \leq C_1$ and $C \leq C_2$.