

# 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.

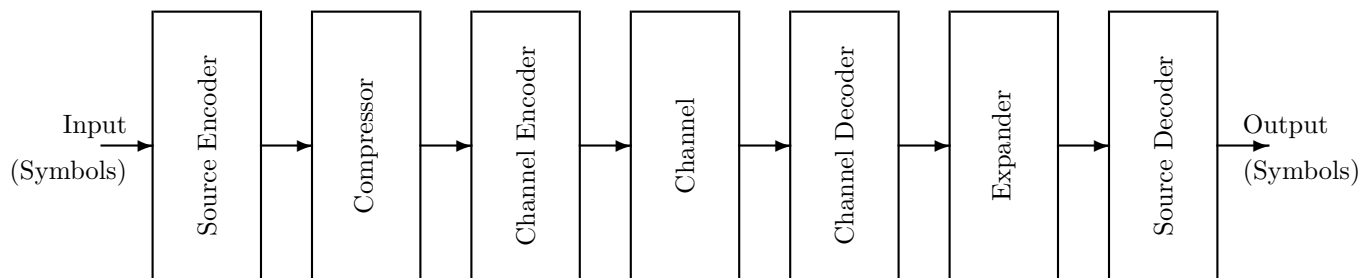


Figure 7.1: Elaborate communication system

In this chapter we present an abstract model that is general enough to represent each of these boxes, and define for such a model the input information, output information, information loss, noise, and mutual information. Formulas bounding the noise and loss of two such boxes connected together are presented.

Because each of these boxes 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 selected from a discrete set of mutually exclusive possibilities; only one of which occurs at a time. The output is one of another discrete set of mutually exclusive values.
- **Finite:** The set of possible inputs is finite. Similarly, the output is one of another finite set of values.
- **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 different outputs when it is presented with the same input again (the model is also valid for deterministic processes). We will see below that nondeterministic processes introduce noise.

---

Author: Paul Penfield, Jr.

Version 1.1.0, March 26, 2004. Copyright © 2004 Massachusetts Institute of Technology

URL: <http://www-mtl.mit.edu/Courses/6.050/notes/chapter7.pdf>

start: <http://www-mtl.mit.edu/Courses/6.050/notes/index.html>

back: <http://www-mtl.mit.edu/Courses/6.050/notes/chapter6.pdf>

next: <http://www-mtl.mit.edu/Courses/6.050/notes/chapter8.pdf>

- **Nontransparent:** It may not be possible to “see” the input from the output, i.e., determine the input state by observing the output state (the model is also valid for transparent processes). We will use the term “lossy” to describe non-transparent processes because knowledge about the input is lost by the time the output is created.

It is useful to represent processes by boxes with inputs and outputs, that can be connected together. One way of doing that is shown above, in Figure 7.1. Another way is useful if the processor is made of electronic parts. Then the input and output may exist physically as one or more wires, with, for example, a high voltage representing logical 1 and a low voltage logical 0. A diagram using these physical wires is called a circuit diagram. For example, the circuit diagram of an AND gate is shown in Figure 7.5.

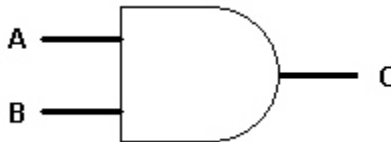


Figure 7.2: Circuit diagram of an *AND* gate

Another process model (one we will emphasize here) shows the set of all possible input states and output states. For example, a two-input one-output gate would have four input possibilities and two output possibilities, as shown in Figure 7.3.

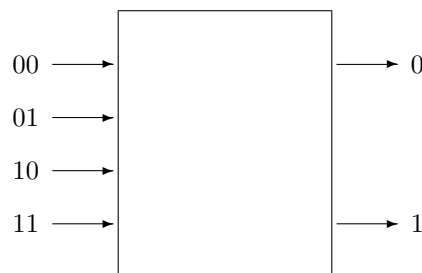


Figure 7.3: Process model of a two-input one-output gate.

Note that the number of possible bit patterns is greater than the number of physical wires connected to a gate; each wire could have two possible states, and so for  $n$ -input gates there would be  $2^n$  input states. The process model for an *AND* gate is shown later.

Still another representation is introduced below, in which the information transmitted to and from the process is shown explicitly.

## 7.1 Process Model

A process that can have  $n$  inputs and  $m$  outputs, where  $n$  and  $m$  are integers, is shown in Figure 7.4. The  $n$  input states are mutually exclusive, as are the  $m$  output states.

This model for processes is conceptually simple and general. However, it can be difficult to use for practical calculation. The reason for this 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 discrete process with 32 inputs. This “exponential explosion” of the number of possible input states gets even more severe when the process represents the

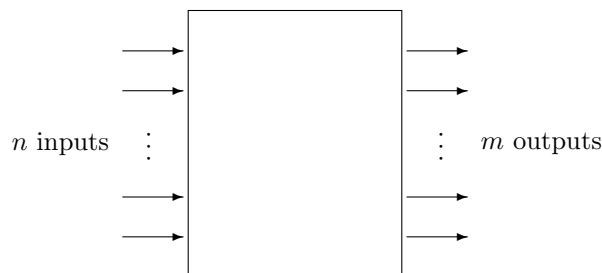


Figure 7.4: Process Model

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 raised to that number of states. To put all this in perspective, the number of microseconds since the big bang is thought to be less than  $5 \times 10^{23}$ .

We assume that each possible input state of a process can lead to one or more of the output states. For each input  $i$  denote the probability that this input leads to the output  $j$  as  $c_{ji}$ . These parameters can be thought of as a table, or matrix, with as many columns as there are input states, and as many rows as output states. We will use  $i$  as an index over the input states and  $j$  over the output states, and we will 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 parameters”  $c_{ji}$  are properties of the process, not properties of whatever drives the process, i.e., whatever inputs are provided to the process. Thus in one sense these parameters are probabilities, but they do not depend on the probability distribution of the input  $p(A_i)$ . Therefore we are denoting them with a letter other than  $p$ .

The transition parameters 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 is selected.

$$0 \leq c_{ji} \leq 1 \quad (7.1)$$

$$1 = \sum_j c_{ji} \quad (7.2)$$

Note that this description has great generality. It applies to a deterministic process, for which the output is determined by the input (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 (although usually in this case the number of possible states is so large that this representation is impractical). It applies whether 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. In particular, the conditional output probabilities, conditioned on the input, are

$$p(B_j | A_i) = c_{ji} \quad (7.3)$$

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

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

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

$$p(A_i, B_j) = p(B_j)p(A_i | B_j) \quad (7.5)$$

$$= p(A_i)p(B_j | A_i) \quad (7.6)$$

$$= p(A_i)c_{ji} \quad (7.7)$$

### 7.1.1 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

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

The process model for this gate is shown in Figure 7.5.

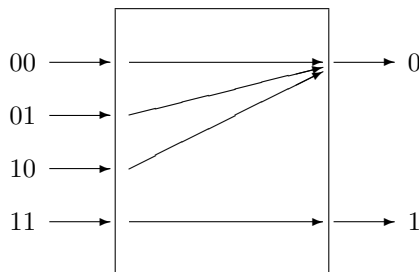


Figure 7.5: Process model of an AND gate

### 7.1.2 Example: Binary Channel

Consider 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 process. We represent this channel by a process 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.

The transition matrix for this channel is

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

The input information  $I_{in}$  for this process is 1 bit if the two values are equally likely; more generally

$$I_{in} = p(A_0) \log_2 \left( \frac{1}{p(A_0)} \right) + p(A_1) \log_2 \left( \frac{1}{p(A_1)} \right) \quad (7.10)$$

The output information  $I_{out}$  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:  $I_{out} = I_{in}$ . 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”  $\epsilon$  is flipped to the “wrong” value 0, and hence is “correct”

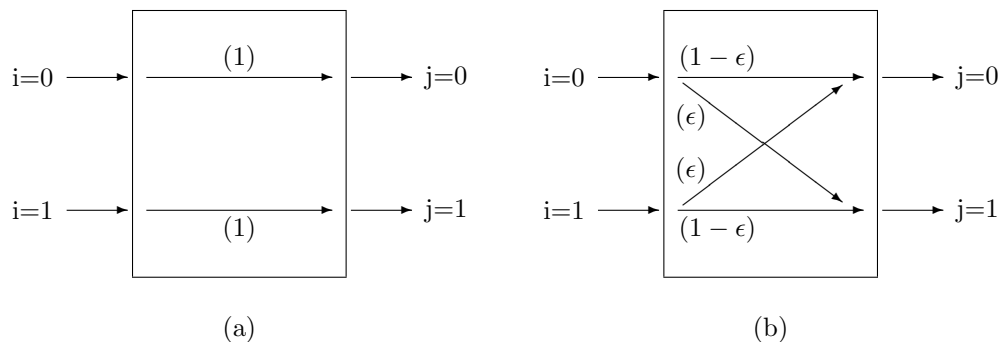


Figure 7.6: (a) Binary channel without errors (b) Symmetric binary channel with errors

only with probability  $1 - \epsilon$ . Similarly, for the input of 0, the probability of error is  $\epsilon$ . Then the transition matrix is

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} 1 - \epsilon & \epsilon \\ \epsilon & 1 - \epsilon \end{bmatrix} \quad (7.11)$$

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; we will see examples of other processes that introduce noise but have no loss, and vice versa.

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 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 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.

## 7.2 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  $I_{in}$  is the same as the entropy of this source.

$$I_{in} = \sum_i p(A_i) \log_2 \left( \frac{1}{p(A_i)} \right) \quad (7.12)$$

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,

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

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,  $I_{out}$  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  $p(A_i | B_j)$

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

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

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

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 is easy to demonstrate that  $L \leq I_{in}$  or, in words, that the uncertainty after learning the output is less than (or perhaps equal to) the uncertainty before. To prove this, use the Gibbs inequality, covered in Chapter 5:

$$\sum_i p_i \log_2 \left( \frac{1}{p_i} \right) \leq \sum_i p_i \log_2 \left( \frac{1}{p'_i} \right) \tag{7.16}$$

where  $p_i$  and  $p'_i$  are any two probability distributions. We use  $p(A_i | B_j)$  for  $p_i$  and  $p(A_i)$  for  $p'_i$ . Then we multiply the result by  $p(B_j)$  and sum over  $j$ . The left side is then just  $L$  and with help from Bayes' Theorem the right side evaluates to

$$\begin{aligned}
\sum_j p(B_j) \sum_i p(A_i | B_j) \log_2 \left( \frac{1}{p(A_i)} \right) &= \sum_{ji} p(B_j) p(A_i | B_j) \log_2 \left( \frac{1}{p(A_i)} \right) \\
&= \sum_{ji} p(A_i, B_j) \log_2 \left( \frac{1}{p(A_i)} \right) \\
&= \sum_{ij} p(B_j | A_i) p(A_i) \log_2 \left( \frac{1}{p(A_i)} \right) \\
&= \sum_i p(A_i) \log_2 \left( \frac{1}{p(A_i)} \right) \\
&= I_{in}
\end{aligned} \tag{7.17}$$

The amount of information we learn about the input state upon being told the output state is our uncertainty before being told, which is  $I_{in}$ , less our uncertainty after being told, which is  $L$ . We have just shown that this amount cannot be negative, since  $L \leq I_{in}$ . Let us denote the amount we have learned as  $M = I_{in} - L$ , 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:

$$I_{in} = \sum_i p(A_i) \log_2 \left( \frac{1}{p(A_i)} \right) \tag{7.18}$$

$$L = \sum_j p(B_j) \sum_i p(A_i | B_j) \log_2 \left( \frac{1}{p(A_i | B_j)} \right) \tag{7.19}$$

$$M = I_{in} - L \tag{7.20}$$

$$0 \leq M \leq I_{in} \tag{7.21}$$

$$0 \leq L \leq I_{in} \tag{7.22}$$

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.

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

$$\begin{aligned}
N &= \sum_i p(A_i) \sum_j p(B_j | A_i) \log_2 \left( \frac{1}{p(B_j | A_i)} \right) \\
&= \sum_i p(A_i) \sum_j c_{ji} \log_2 \left( \frac{1}{c_{ji}} \right)
\end{aligned} \tag{7.23}$$

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:

$$I_{out} = \sum_i p(B_j) \log_2 \left( \frac{1}{p(B_j)} \right) \quad (7.24)$$

$$N = \sum_i p(A_i) \sum_j c_{ji} \log_2 \left( \frac{1}{c_{ji}} \right) \quad (7.25)$$

$$M = I_{out} - N \quad (7.26)$$

$$0 \leq M \leq I_{out} \quad (7.27)$$

$$0 \leq N \leq I_{out} \quad (7.28)$$

It follows from these results that

$$I_{out} - I_{in} = N - L \quad (7.29)$$

### 7.2.1 Example: Binary Channel

For the SBC with bit error probability  $\epsilon$ , 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:

$$I_{in} = I_{out} = 1 \text{ bit} \quad (7.30)$$

$$L = N = \epsilon \log_2 \left( \frac{1}{\epsilon} \right) + (1 - \epsilon) \log_2 \left( \frac{1}{(1 - \epsilon)} \right) \quad (7.31)$$

$$M = 1 - \epsilon \log_2 \left( \frac{1}{\epsilon} \right) - (1 - \epsilon) \log_2 \left( \frac{1}{(1 - \epsilon)} \right) \quad (7.32)$$

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  $M = I_{in} - L$  to be passed through the channel to the output.

## 7.3 Deterministic Examples

This process 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 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,  $I_{in} = I_{out}$  and there is no noise or loss:  $N = L = 0$ . The information that gets through the gate is  $M = I_{in}$ . 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

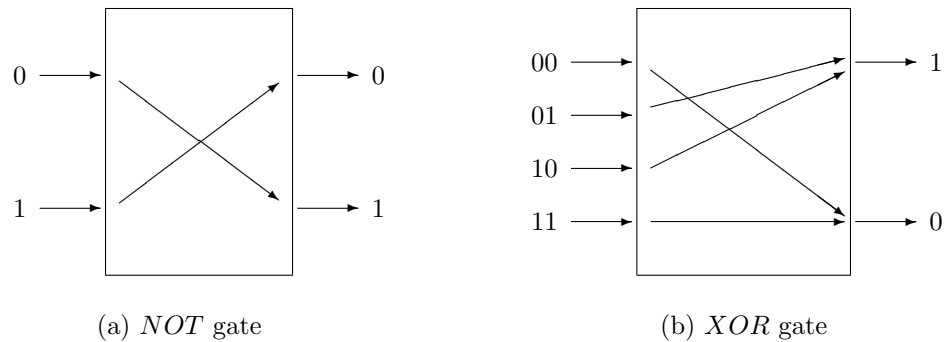


Figure 7.7: Deterministic Examples

as a discrete process using a diagram like Figure 7.6(b), there are four mutually exclusive inputs and two mutually exclusive outputs. See Figure 7.7(b).

If the probabilities of the four inputs are each 0.25, then  $I_{in} = 2$  bits, and the two output probabilities are each 0.5 so  $I_{out} = 1$  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 0 1 or 1 0. 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  $2^n$ .

### 7.3.1 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 1 1 1 and the input 0 to the output 0 0 0. 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  $M = I_{in} = I_{out}$ . 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 0 0 0 is connected only to the outputs 0 0 0 (with high probability) and 0 0 1, 0 1 0, and 1 0 0 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 zero 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 the physics of the channel and also 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 outputs) but no noise (since each input goes to only one output).

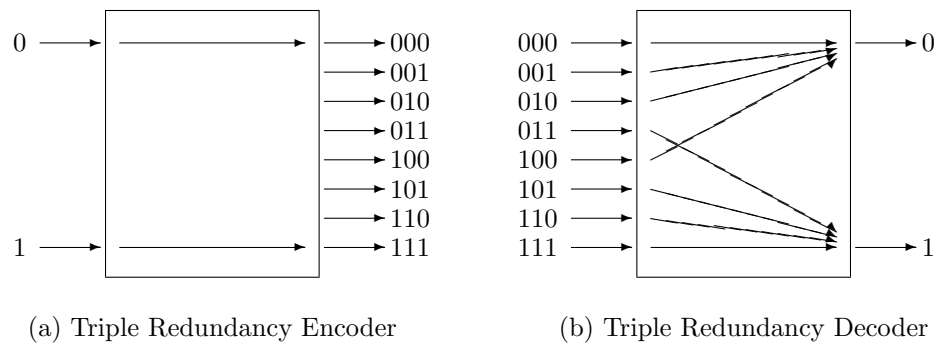


Figure 7.8: Error Correcting Examples

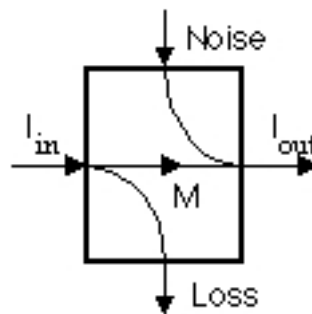


Figure 7.9: General information flow in a discrete memoryless channel

## 7.4 Capacity

In the previous chapter 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

$$C = WM_{max} \quad (7.33)$$

It is easy to see that  $M_{max}$  cannot be arbitrarily large, since  $M \leq I_{in}$  and  $I_{in} \leq \log_2 n$  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.

## 7.5 Observations

It has been shown that all five information measures,  $I_{in}$ ,  $I_{out}$ ,  $L$ ,  $N$ , and  $M$  are nonnegative. It is not necessary that  $L$  and  $N$  be the same, although they are for the symmetric binary channel. 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).

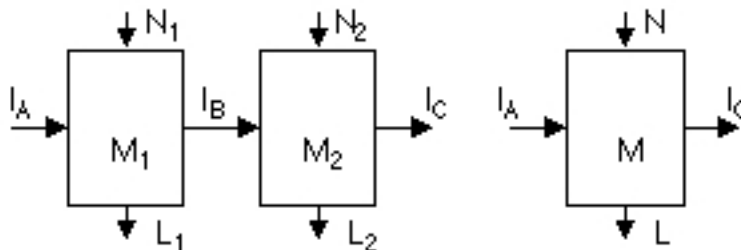


Figure 7.10: Discrete Memoryless Channel: Cascade of Two as a Single Process

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.

## 7.6 Cascaded Processes

It is interesting to consider two processes in cascade. The term “cascade” 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  $I_{in}$ ,  $I_{out}$ ,  $L$ ,  $N$ , and  $M$ . Figure 7.10 shows a cascaded pair of processes on the left, 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.

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. 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  $I_{in}$ ,  $I_{out}$ ,  $L$ ,  $N$ , 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

$$L - N = (L_1 + L_2) - (N_1 + N_2) \quad (7.34)$$

It is then straightforward (though perhaps tedious) to show that the loss  $L$  for the overall process is not

equal to the sum of the losses for the two components  $L_1 + L_2$ , but instead

$$0 \leq L_1 \leq L \leq L_1 + L_2 \quad (7.35)$$

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

$$L_1 + L_2 - N_1 \leq L \leq L_1 + L_2 \quad (7.36)$$

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  $N_1 + N_2$ :

$$0 \leq N_2 \leq N \leq N_1 + N_2 \quad (7.37)$$

$$N_1 + N_2 - L_2 \leq N \leq N_1 + N_2 \quad (7.38)$$

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

$$M_1 - L_2 \leq M \leq M_1 \leq I_{in,1} \quad (7.39)$$

$$M_1 - L_2 \leq M \leq M_1 + N_1 - L_2 \quad (7.40)$$

$$M_2 - N_1 \leq M \leq M_2 \leq I_{out,2} \quad (7.41)$$

$$M_2 - N_1 \leq M \leq M_2 + L_2 - N_1 \quad (7.42)$$

Two other formulas for  $M$  are easily derived from Equation 7.21 applied to the first process and the cascade, and Equation 7.26 applied to the second process and the cascade:

$$\begin{aligned} M &= M_1 + L_1 - L \\ &= M_1 + N_1 + N_2 - N - L_2 \end{aligned} \quad (7.43)$$

$$\begin{aligned} M &= M_2 + N_2 - N \\ &= M_2 + L_2 + L_1 - L - N_1 \end{aligned} \quad (7.44)$$

where the second formula in each case comes from the use of Equation 7.34.

As a special case, note that if the second process is lossless,  $L_2 = 0$  and therefore  $M = M_1$ . 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  $M = M_2$ .