The same counting argument works fine for nonlinear codes too. When the decoder receives $\mathbf{r} = \mathbf{t} + \mathbf{n}$, his aim is to deduce both $\mathbf{t}$ and $\mathbf{n}$ from $\mathbf{r}$. If it is the case that the sender can select any transmission $\mathbf{t}$ from a code of size $S_{\mathbf{t}}$, and the channel can select any noise vector from a set of size $S_{\mathbf{n}}$, and those two selections can be recovered from the received bit string $\mathbf{r}$, which is one of at most $2^N$ possible strings, then it must be the case that

$$S_{\mathbf{t}} S_{\mathbf{n}} \leq 2^N. \tag{1.57}$$

So, for a $(N, K)$ two-error-correcting code, whether linear or nonlinear,

$$2^K \left[ \binom{N}{2} + \binom{N}{1} + \binom{N}{0} \right] \leq 2^N. \tag{1.58}$$

Solution to exercise 1.11 (p.14). There are various strategies for making codes that can correct multiple errors, and I strongly recommend you think out one or two of them for yourself.

If your approach uses a linear code, e.g., one with a collection of $M$ parity checks, it is helpful to bear in mind the counting argument given in the previous exercise, in order to anticipate how many parity checks, $M$, you might need.

Examples of codes that can correct any two errors are the $(30, 11)$ dodecahedron code on page 20, and the $(15, 6)$ pentagonful code to be introduced on p.221. Further simple ideas for making codes that can correct multiple errors from codes that can correct only one error are discussed in section 13.7.

Solution to exercise 1.12 (p.16). The probability of error of $\mathrm{R}_3^2$ is, to leading order,

$$p_{\mathrm{b}}(\mathrm{R}_3^2) \simeq 3 \left[ p_{\mathrm{b}}(\mathrm{R}_3) \right]^2 = 3(3f^2)^2 + \cdots = 27f^4 + \cdots, \tag{1.59}$$

whereas the probability of error of $\mathrm{R}_9$ is dominated by the probability of five flips,

$$p_{\mathrm{b}}(\mathrm{R}_9) \simeq \binom{9}{5} f^5 (1-f)^4 \simeq 126 f^5 + \cdots. \tag{1.60}$$

The $\mathrm{R}_3^2$ decoding procedure is therefore suboptimal, since there are noise vectors of weight four that cause it to make a decoding error.

It has the advantage, however, of requiring smaller computational resources: only memorization of three bits, and counting up to three, rather than counting up to nine.

This simple code illustrates an important concept. Concatenated codes are widely used in practice because concatenation allows large codes to be implemented using simple encoding and decoding hardware. Some of the best known practical codes are concatenated codes.