Chapter One: Error-Correcting Codes

Section 1. From the Alt Code to the Hamming Code

In a war, those who start it are usually at a safe distance from the front. Orders are sent to the troops via binary messages, sequences of 0s and 1s which may be interpreted according to a code by both the transmitter and the receiver. For example, if we want the troops to "Do nothing", we send the message 0. If we want the troops to "Drop the bomb", we send the message 1. This process is called **encoding**.

When the transmitted message is received, it must be interpreted. This process is called **decoding**. Here, 0 is decoded as "Do nothing" and 1 as "Drop the bomb". However, electronic transmissions may contain errors, due to imperfect equipments or enemy sabotage. We will assume that an error consists of a single digit-reversal per transmission.

In other words, the worst that can happen is that one of the 0s may turn into a 1, or one of the 1s may turn into a 0, but neither can happen twice and both cannot happen together in a single transmission. Even this is serious enough. In our example, if a 1 turns into a 0, we may not wake up in time to find that we have lost World War II. On the other hand, if a 0 turns into a 1, we may have started World War III while still fighting World War II.

We would like to have some way of telling whether we can trust what we have received. So we modify our simple code as follows: 00 would mean "Do nothing" and 11 would mean "Drop the bomb". If an error occurs, we will receive either 10 or 01, and we will know something has happened to the message. This is a prototype of what is known as an **error-detecting code**.

When we learn that an error has occurred, the natural thing is to ask headquarters to retransmit the message. If errors occur only infrequently, this is tolerable. If they occur often enough, it is at least a nuisance. Moreover, the request for retransmission is also sent electronically, and errors can occur there too.

What we would like is a code which not only tells us when something has gone wrong, but tells us exactly what has gone wrong. This is asking for a lot, but as is often the case, when we believe that something can be done, there may just be a way to do so. We now modify our code again, with 000 meaning "Do nothing" and 111 meaning "Drop the bomb". If we receive 001, 010 or 100, we do nothing. If we receive 110, 101 or 011, we drop the bomb. This is a prototype of what is known as an **error-correcting code**.

A. Liu, *S.M.A.R.T. Circle Projects*, Springer Texts in Education, DOI 10.1007/978-3-319-56811-9_1

Note that it is unrealistic to increase the length of the message and continue to assume that there is at most one error per transmission. So let us fix the length of any message to 15 binary digits, with at most one digit-reversal per transmission.

In some sense, even this is unrealistic since if one digit-reversal can occur, there is no reason why a second one cannot. What must be emphasized is that our assumption amounts to a mathematical model which simulates reality, but is not reality itself. We can get away with it if the probability of a single digit-reversal is high enough to worry us, but the probability of multiple digit-reversals is low enough to lose any sleep over.

Of the 15 digits, we may view some of them as conveying the intended message while the remaining ones are for security measure. The **efficiency** of a code using this 15-digit transmitter is defined as $\frac{n}{15}$. where n is the number of digits used for the message.

How can we extend our earlier examples with short messages to 15-digit messages? In an error-detecting code, we need to distinguish between two scenarios, whether the message contains an error or not. A single binary digit would allow us to do so. Hence the efficiency of such a code can be as high as $\frac{14}{15}$. Obviously. we cannot have $\frac{15}{15}$ as we will have no protection at all.

How can we encode the intended message 10110100010111? Should we add a 0 or a 1 as the fifteen digit? Let us reexamine the simple example earlier. To the message 0, we add a 0, and to the message 1, we add a 1. Note that we are not copying the message, but arrange for the coded message to contain an even number of 1s. Since 10110100010111 has 8 1s, we add a 0 to yield the coded message 101101000101110. This code is called the **parity-check code**, and the added digit is called the parity-check digit.

Decoding is straight-forward. Simply count the number of 1s in the received message. If no digit-reversal has occurred, this number is even as agreed. If a single digit-reversal has occurred, whether a 0 turned into a 1 or vice versa, the number of 1s will become odd. Thus a received message 001100101010110 contains an error. We do not know the intended message as any of the 15 digits may be the one which has been reversed.

In an error-correcting code on the 15-digit transmitter, we need to distinguish between sixteen scenarios, whether the message contains an error, and if so, which of the 15 digits has been reversed. We need four binary digits to do so because $2^4. = 16$. Thus the efficiency of such a code cannot be higher than $\frac{11}{15}$.

However, there is nothing in our earlier example to suggest how we can encode an 11-digit message. There, we stretch the message 0 to 000 and the message 1 to 111. This time, we are copying the message, twice, so that there are altogether three copies of the intended message. This means that the efficiency of this code is only as high as $\frac{5}{15}$, with the message 10110 encoded as 101101011010110. This code is called the **triple-repetition code**. It is also called the **Alt code** after its inventor (see [2]).

Decoding is based on the simple idea of majority rule. With at most one digit-reversal among three copies of the intended message, at least two copies must be correct. If all three agree, there are no errors. If not, the copy in the minority can safely be discarded. Thus a received message 010010000101001 contains an error in the seventh digit, and the intended message is 01001.

In 1984, Mark Rabenstein was an eighth grade student in Edmonton and a member of the SMART Circle. He had the following complaint about the Alt code: "If at most one digit-reversal can occur, why do we have to have three copies of the message? Wouldn't two be enough?"

"If an error has occurred so that the two copies are different, how do you know which is the correct one without reference to a third copy?" I asked.

"Just tag a parity-check digit to one of the copies, and you can tell if that one is correct."

That was a brilliant observation. In our fifteen-digit transmitter, we can use seven digits for the intended messages, repeat it a second time, and tag a parity-check digit to the second copy. For instance, the intended message 0011001 is encoded as 001100100110011.

Decoding is straight-forward. Compare the two copies of the intended message. If they agree, we can accept it. If not, apply the parity-check to the second copy to decide which one we would accept. For instance, if 101010110111011 is received, we see that 1010101 and 1011101 are different. Applying the parity-check to 10111011, we have an even number of 1s. Thus the received message contains an error in the fourth digit, and the intended message is 1011101.

The efficiency of $\frac{7}{15}$ is a big improvement over the Alt code. The **Rabenstein code** was published in [10].

The next break-through came in 1997. Han-Shian Liu (no relation to the author), then a sixth grade student in Taipei, was a member of the Chiu Chang Mathematical Circle. His (error-free) email message to me contained a gem.

"Mark was really smart to think of using the parity-check digit in an error-correcting code. It works so beautifully. Then I wonder whether I can make even more use of it. After experimenting with the idea for a while, I drew a tic-tac-toe board. (See Figure 1.1.) Each of the nine boxes contained a message digit. Then I added six parity-check digits, one for each row and one for each column."

For example, if the intended message is 011101001, we use the digits A to J are used to convey it, as shown in the grid below on the left. The digits K to Q are chosen so that the number of 1s in each row and each column is even.

A 0	D 1	G 0	N 1
B 1	E 0	H 0	P 1
C 1	F 1	J 1	Q 1
K 0	L 0	M 1	

A 0	D 1	G 0	N 1
B 1	E 0	H 0	P 1
C 0	F 1	J 1	Q 1
K 0	L 0	M 1	

Figure 1.1

Suppose the received message is as shown in the grid above on the right. Then the parity-check fails for the third row and the first column. It follows that the single digit-reversal occur at their intersection, namely, the box containing the digit C.

Han-Shian's code uses parity-check in two dimensions, and has an efficiency of $\frac{9}{15}$. This **Liu code** was published in [8], a paper which also contains an improved version that reaches the efficiency of $\frac{10}{15}$. So we are one step away from a perfect code with efficiency of $\frac{11}{15}$.

To mount this final assault, we use a set-theoretic representation of the tic-tac-toe board. Label the columns with a, b and c from left to right, and the rows with d, e and f from top to bottom. Then each box containing a message digit is represented by a two-element subset of $\{a, b, c, d, e, f\}$ while each box containing a parity-check is represented by a one-element subset. These two groups are separated from each other by a vertical line. Each parity-check digit is chosen so that the total number of 1s under columns containing the element which represents it is even. For example, the message 101111000 is encoded as shown in the chart below.

a	a	a							a					
			b	b	b					b				
						c	c	c				c		
d			d			d							d	
	e			e			e							e	
		f			f			f							f
1	0	1	1	1	1	0	0	0	1	1	0	1	0	0
A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q

Suppose the message in the chart below has been received. To decode it, we count the total numbers of 1s under the columns containing the elements a, b, c, d, e and f. They are 0, 3, 2, 2, 3 and 2 respectively. Since parity-check fails for the elements b and e, the error occurs at the element under the subset $\{b, e\}$. Hence the intended message is 000101010.

a	a	a							a					
			b	b	b					b				
						c	c	c				c		
d			d			d							d	
	e			e			e							e	
		f			f			f							f
0	0	0	1	1	1	0	1	0	0	0	1	1	1	1

It is now clear why there is still room for improvement. Han-Shian had only made use of some of the non-empty subsets but not all of them. If we cut the set down to $\{a, b, c, d\}$, there are exactly fifteen non-empty subsets, four of which are singletons that give rise to the parity-check digits. For example, the message 10111100011 is encoded as shown in the chart below.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	0	1	1	1	1	0	0	0	1	1	0	1	0	0

Decoding is by the same method as in the Liu code. Suppose the message in the chart below has been received. The total numbers of 1s under the columns containing the elements a, b, c and d are 3, 5, 4 and 6 respectively. Since parity-check fails for the elements a and b, the error occurs at the element under the subset $\{a, b\}$. Hence the intended message is 10111100011.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	0	1	1	1	0	0	0	0	1	1	0	1	0	0

This perfect code is due independently to Golay [3] and Hamming [4] and is commonly known just as the **Hamming code**. The description given here is from [1]. Since it was already known early in the history of error-correcting codes, the Rabenstein code and the Liu code are irrelevant from an application point of view. However, they are the work of students and have high pedagogical value. They form a sequence of plausible reasoning that could eventually lead a lesser mortal from the simplistic Alt code to the same discovery by the founding fathers of coding theory.

In summary, the first step is to replace three copies of the messages by two, and incorporating parity-check. The second step is to perform parity-check in two dimensions instead of one dimension. The third step is to represent the code in set-theoretic format instead of geometric format. The chronology is given in [5].

Section 2. Two Applications of the Hamming Code

We give two unexpected applications of the Hamming code (see [7]). Both of them have strong recreational flavor.

Alice and Michael, along with thirteen of their friends, enter in a team competition organized by a certain hi-tech company. They will be put respectively into rooms A to Q (there are no rooms I and O). Each room is considered to be in one of two states, 0 or 1, assigned completely at random. Once they are isolated in their rooms, the team members will be informed of the state of each room except their own. Simultaneously, each must either pass, or declare the state of her or his room. They will have no further communication with their teammates, and are not aware of the action taken by any of them. If everybody passes, the team will be disqualified. If at least one declaration is incorrect, the team will also be disqualified. On the other hand, if there is at least one declaration, and all declarations are correct, the team wins a prize.

Alice, Michael and friends are given a short time to come up with some strategy. For instance, they could designate Alice as the guesser and have everyone else pass. The probability of winning a prize would then be $\frac{1}{2}$. However, they would like to do better. Alice and Michael come up with the following strategy based on the Hamming code.

We first give an illustration. Suppose Alice is in Room G and Michael is in Room N, and the actual states are as shown in the chart below.

A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q
a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
0	1	1	1	0	1	1	0	0	1	1	0	0	1	0

Alice is in Room G which corresponds to the subset $\{a, c\}$. So she applies the parity-check to the element b and to the element d. Both tests pass. Now she applies the parity-check to the element a and to the element c, without taking the information on her room, which is unknown to her. Both will pass if the state of Room G is 0. So Alice will declare the opposite state 1.

Michael is in Room N which corresponds to the subset $\{b\}$. So he applies the parity-check to the element a. It fails. So there is no need to apply the parity-check to the elements c and d. Michael just passes.

This illustrates how things work in general. A team member declares if, and only if, the state of her or his room can be chosen to correct the single error in the corresponding Hamming code, but the opposite state is declared.

If the original set-up contains no errors when treated as a Hamming code, every team member will make an incorrect declaration. If the original set-up contains an error, only the team member in the room corresponding to where the error occurs will declare, and the declaration will be correct.

Recall that in the Hamming code, the 4 protection digits are uniquely determined by the 11 message digits. Of the $2^4 = 16$ possible sequences for those 4 digits, only the one contains no errors. Hence $\frac{15}{16}$ of the time, the original set-up contains an error. It follows that the probability of winning a prize is $\frac{15}{16}$.

This application is given in [12] as the problem titled *Crowning the Minotaur*. A special case was presented in [6]. The next application, in which we continue the story of Alice and Michael, is based on a problem in the Fall Round of the 2007 International Mathematics Tournament of the Towns. Only the former source mentions the Hamming code.

To celebrate their success in the team competition, the fifteen friends have a party, during which Alice and Michael perform a magic trick. While Michael is out of the room, the audience chooses one of the fifteen letters from A to Q inclusive, but excluding I and O. Then the audience places fifteen coins in a row, arbitrarily deciding whether each should be heads or tails. Alice either leave them alone or turns over exactly one coin, and leaves the room while Michael is brought back in. By looking at the coins and without knowing which one Alice has turned over, Michael determines the letter chosen by the audience.

Let us give an illustration. We use 0 to stand for a coin which is heads and 1 for a coin which is tails. Suppose the audience chooses the letter K and places the coins as shown in the chart below.

A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q
a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	0	1	1	1	0	0	0	0	1	1	0	1	0	0

Applying the parity-check, Alice finds that it fails for a and b but passes for c and d. Since the letters under column K are b and d, Alice wants the parity-check to fail for b and d but pass for a and c. So she needs to reverse the parity for a and d, This can be done by flipping the coin under the subset $\{a, d\}$. Hence she changes the 0 under column H to 1. Had the audience chosen the letter F, Alice would have left the coins alone.

When Michael returns, he applies the parity-check, and finds that it passes for a and c but fails for b and d. This tells him that the letter chosen by the audience is the one associated with $\{b, d\}$, namely K.

Section 3. Reed-Muller Code

We now take up the issue of the correction of multiple errors. Our primary example is the Reed-Muller code (see [9] and [11]), which may be considered as an extension of the Hamming code. For a fifteen-digit transmitter, it can correct up to three errors. We set up a chart as in the Hamming code, except that there is now an additional vertical line separating the two-element subsets from the others, as shown below.

Suppose the intended message is 00101. We will now add ten digits for protection. To see what digit we must add under the column $\{a, b\}$, we consider the digits under the other columns which contain $\{a, b\}$, namely, $\{a, b, c, d\}, \{a, b, c\}$ and $\{a, b, d\}$. Since they are 0, 0 and 1 respectively, and we want an even number of 1s, we add the digit 1 under the column $\{a, b\}$. The digits under the next five columns are chosen in an analogous manner, and the chart is them completed as in the Hamming code, as shown below.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
0	0	1	0	1	1	0	1	1	0	1	1	0	1	0

Suppose the following transmission via the Reed-Muller code has been received, with up to three errors.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
0	0	1	0	0	0	0	1	1	0	1	0	1	1	0

We now perform a parity-check on the subset $\{a, b\}$. The digits under the columns $\{a, b, c, d\}, \{a, b, c\}$, $\{a, b, d\}$ and $\{a, b\}$ are 0, 0, 1 and 0. Hence the test fails. Note that this is not saying that the digit under $\{a, b\}$ is an error. It says that either one of these four digits is an error, or three of the four are errors. Performing parity-checks on the other two-element subsets, we find that the test fails for $\{b, c\}, \{b, d\}$ and $\{c, d\}$ but passes for $\{a, c\}$ and $\{a, d\}$.

We now perform a parity-check on the subset $\{a\}$. The digits under the columns $\{a, b, c, d\}, \{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, $\{a, b\}, \{a, c\}$, $\{a, d\}$ and $\{a\}$ are 0, 0, 1, 0, 0, 0, 1 and 0. Hence the test passes. Performing parity-checks on the remaining subsets, we find that the test fails for $\{b\}$, $\{c\}$ and $\{d\}$.

We use P, Q and R to denote the three possible subsets of $\{a, b, c, d\}$ which are errors. Then an odd number of them contain $\{a, b\}$, $\{b, c\}, \{b, d\}$, $\{c, d\}$, $\{b\}$, $\{c\}$ and $\{d\}$ while an even number of them contain $\{a, c\}$, $\{a, d\}$ and $\{a\}$.

So $\{a\}$ appears either 0 or 2 times in P, Q and R while each of $\{b\}$, $\{c\}$ and $\{d\}$ appears either 1 or 3 times. Because $\{a, b\}$ appears either 1 or 3 times and it cannot appear without $\{a\}$, $\{a\}$ must appear exactly 2 times, say in P and Q. Also, $\{b\}$ cannot appear 3 times as otherwise $\{a, b\}$ will appear twice, but it must appear together with $\{a\}$. We may assume that it appears only in P.

Since each of $\{b, c\}$ and $\{b, d\}$ appears 1 or 3 times, both $\{c\}$ and $\{d\}$ must appear in P. Since each of $\{a, c\}$ and $\{a, d\}$ appears 0 or 2 times, both $\{c\}$ and $\{d\}$ must appear in Q. Since $\{c, d\}$ appears 1 or 3 times, both $\{c\}$ and $\{d\}$ must appear in R also. This is consistent with each of them appearing 1 or 3 times. Hence the errors are $P = \{a, b, c, d\}$, $Q = \{a, c, d\}$ and $R = \{c, d\}$, and the correct message is 10110.

While the decoding procedure in our example seems rather *ad hoc*, it does have a firm theoretical basis (see [1]). Let us give a more mathematical analysis of the above example.

For any set S, we define S^2 to be the collections of all non-empty subsets of S of size up to 2. In the above example,

$$\begin{aligned} P^2 &= \{\{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}, \\ Q^2 &= \{\{a\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}, \{c, d\}\}, \\ R^2 &= \{\{c\}, \{d\}, \{c, d\}\}. \end{aligned}$$

The symmetric difference of a number of collections consists of all elements which belong to an odd number of these collections. The symbol for symmetric difference is Δ. In the above example,

$$P^2 \Delta Q^2 \Delta R^2 = \{\{b\}, \{c\}, \{d\}, \{a, b\}, \{b, c\}, \{b, d\}, \{c, d\}\}.$$

The subsets in this collection are precisely those for which the parity-check fails. Thus $P^2 \Delta Q^2 \Delta R^2$ may be considered as the pattern of parity disturbance caused by the errors P, Q and R.

This pattern of parity disturbance may be caused by a different group of errors, namely $\{b, c, d\}$, $\{a, b\}$, $\{a\}$ and $\{b\}$, in that

$$\begin{aligned} & \{b, c, d\}^2 \Delta \{a, b\}^2 \Delta \{a\}^2 \Delta \{b\}^2 \\ = \; & \{\{b\}, \{c\}, \{d\}, \{b, c\}, \{b, d\}, \{c, d\}\} \Delta \{\{a\}, \{b\}, \{a, b\}\} \Delta \{\{a\}\} \Delta \{\{b\}\} \\ = \; & \{\{b\}, \{c\}, \{d\}, \{a, b\}, \{b, c\}, \{b, d\}, \{c, d\}\}. \end{aligned}$$

However, this does not invalidate the Reed-Muller Code as the second group contains four errors, more than the three we are allowed for a 15-digit transmitter. We claim that if two different groups of errors cause the same pattern of parity disturbance in a 15-digit transmitter, then one of them will consist of four or more sets.

Since they cause the same pattern of parity disturbance independently, they must cause no parity disturbance when acting together. After removing common sets from the two groups, we are left with a non-empty collection since the two groups of errors are different. We now prove that in a 15-digit transmitter, the number of sets in any non-empty collection which causes no parity disturbance is at least seven. It follows that four or more of them must come from the same group of errors, and our claim would be justified. We consider four cases.

Case 1. The collection consists only of 1-element sets.
Since there is at least one of them, the parity for the lone element in this set will be disturbed.

Case 2. The collection contains at least one 2-element set but no 3-element or 4-element sets.
The parity for the pair of elements in a 2-element set will be disturbed.

Case 3. The collection contains at least one 3-element set but not $\{a, b, c, d\}$. We may assume that the 3-element set is $\{a, b, c\}$. By itself, it will disturb the parity of $\{a, b\}$. To nullify this, the collection must contain either $\{a, b, d\}$ or $\{a, b\}$, but not both. Similarly, the collection must contain exactly one of $\{a, c, d\}$ and $\{a, c\}$, and exactly one of $\{b, c, d\}$ and $\{b, c\}$. Thus there are four sets in the collection with 2 or 3 elements. Collectively, they will disturb the parity of each of $\{a\}$, $\{b\}$ and $\{c\}$, and we need to include these three 1-element sets, bringing the total to seven sets.

Case 4. The collection contains $\{a, b, c, d\}$.
As in Case 3, the collection must contain either one or three of the set in each column of the chart below.

$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, d\}$	$\{a, b, c\}$	$\{a, b, d\}$	$\{a, c, d\}$
$\{a, b, d\}$	$\{a, c, d\}$	$\{a, c, d\}$	$\{b, c, d\}$	$\{b, c, d\}$	$\{b, c, d\}$
$\{a, b\}$	$\{a, c\}$	$\{a, d\}$	$\{b, c\}$	$\{b, d\}$	$\{c, d\}$

We consider five subcases.
Subcase 4(a). There are no 3-element sets in the collection.
Then we must include all six 2-element sets, bringing the total to seven sets.
Subcase 4(b). There is only one 3-element set in the collection.
We may assume that it is $\{a, b, c\}$. Then we must include $\{a, d\}$, $\{b, d\}$ and $\{c, d\}$. This in turn forces the inclusion of $\{a\}$, $\{b\}$ and $\{c\}$, bringing the total to eight sets.

Subcase 4(c). There are exactly two 3-element sets in the collection.
We may assume that they are $\{a, b, c\}$ and $\{a, b, d\}$. Then we must include $\{a, b\}$ and $\{c, d\}$. This in turn forces the inclusion of $\{c\}$ and $\{d\}$, bringing the total to seven sets.
Subcase 4(d). There are exactly three 3-element sets in the collection.
We may assume that they are $\{a, b, c\}$, $\{a, b, d\}$ and $\{a, c, d\}$. Then we must include $\{a, b\}$. $\{a, c\}$ and $\{a, d\}$, already bringing the total to seven sets.
Subcase 4(e). All four 3-element sets are in the collection.
Then we must also include all six 2-element sets, already bringing the total to eleven sets.

We give some additional examples on the Reed-Muller Code. As before, we use P, Q and R to denote the three possible subsets of $\{a, b, c, d\}$ which are errors.

Example 1.
Decode the received message sent under the Reed-Muller code.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	1	1	1	1	0	1	1	0	0	1	0	0	0	0

Solution:
The parity-check fails for $\{a, b\}$, $\{b, c\}$ and $\{b, d\}$. Thus each of $\{a\}$, $\{b\}$, $\{c\}$ and $\{d\}$ appears an even number of times. Since all are featured in $\{a, b\}$, $\{b, c\}$ or $\{b, d\}$, each appears exactly twice. Let $\{b\}$ appear in P and Q. Each of $\{a\}$, $\{c\}$ and $\{d\}$ appears an odd number of times with $\{b\}$, which means exactly once. Hence they must appear together in R. Since each of $\{a, c\}$, $\{a, d\}$ and $\{c, d\}$ appears an even number of times, $\{a\}$, $\{c\}$ and $\{d\}$ must appear together again, this time with $\{b\}$. Hence the errors are $P = \{a, b, c, d\}$, $Q = \{b\}$ and $R = \{a, c, d\}$, and the correct message is 01101.

Example 2.
Decode the received message sent under the Reed-Muller code.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	0	1	0	1	1	0	1	1	0	1	1	0	1	0

Solution:
All ten parity-checks fail. Hence $P = \{a, b, c, d\}$ and $Q = R = \emptyset$, and the correct message is 00101.

Example 3.
Decode the received message sent under the Reed-Muller code.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	1	0	0	1	1	0	1	1	1	0	0	1	1	0

Solution:
The parity-check fails for $\{b\}$, $\{c\}$, $\{a, b\}$ and $\{b, d\}$. Thus each of $\{a\}$, and $\{d\}$ appears an even number of times. Since the parity-check for $\{a, d\}$ passes, they must appear together twice, say in P and Q. Now $\{b\}$ must appear with $\{a\}$ an odd number of times and with $\{d\}$ an odd number of times. Hence it appears in exactly one of P and Q, sat P, and not in R. Since the parity-check for $\{b, c\}$ pass, $\{c\}$ does not appear together with $\{b\}$, and similarly, it does not appear together with $\{a\}$ or with $\{c\}$. Hence the errors are $P = \{a, b, d\}$, $Q = \{a, d\}$ and $R = \{c\}$, and the correct message is 11101.

Example 4.
Decode the received message sent under the Reed-Muller code.

a	a	a	a		a	a	a				a			
b	b	b		b	b			b	b			b		
c	c		c	c		c		c		c			c	
d		d	d	d			d		d	d				d
1	1	0	0	1	0	0	1	0	1	1	1	1	1	1

Solution:
The parity-check fails for $\{b\}$, $\{c\}$, $\{b, c\}$, $\{b, d\}$ and $\{c, d\}$. Thus each of $\{a\}$ and $\{d\}$ appears an even number of times. Since $\{a\}$ is not featured in $\{b, c\}$, $\{b, d\}$ or $\{c, d\}$, it does not appear at all. Since $\{d\}$ is featured in $\{b, d\}$ and $\{c, d\}$, it appears exactly twice, say in P and Q. Each of $\{b\}$ and $\{c\}$ appears once with $\{d\}$. Hence each appears exactly once. Since the parity check for $\{b, c\}$ fails, they appear together once, say in P. Hence the errors are $P = \{b, c, d\}$, $Q = \{d\}$ and $R = \emptyset$, and the correct message is 11000.

If in our chart listing the subsets of $\{a, b, c, d\}$, we put in a third vertical line separating all three-element subsets from the others (well, just $\{a, b, c, d\}$), we can correct up to seven errors. The message now consists of a single digit, and it is easy to see that encoding simply repeats it to yield a total of fifteen copies. We have come full circle and return to the majority rule (eight out of fifteen in this case) which is the basis for the Alt Code.

Exercises

1. Design an error-correcting code with efficiency $\frac{10}{15}$.

2. Random justice is applied to three prisoners. On Decision Day, each prisoner will be given a hat to wear, which may be black or white. He can see the other two hats but not his own. At some point, the Warden will call for a simultaneous declaration from each prisoner, which he must make without the benefit of knowing how the other two will declare. He must declare "pass", "black" or "white". If the declaration is indeed' the color of his hat, he is right. If it is the other color, he is wrong. The prisoners will only go free if at least one of them declares, and all those who declare are right. The three prisoners get together the night before Decision Day and discuss strategies. What can they do to make the probability of their going free as high as $\frac{3}{4}$?

3. For a 15-digit transmitter, the efficiency of the Reed-Muller Code is $\frac{5}{15}$. Can this be improved?

Bibliography

[1] N. Alon and A. Liu, An application of set theory to coding theory, *Math. Mag.* **62** (1989) 233–237.

[2] F. L. Alt, A Bell Telephone Laboratories' computing machine (I), *Math. Comput.* **3** (1948/49) 1–13.

[3] M. J. E, Golay, Notes on digital coding, *Proc. I.E.E.E.* **37** (1949) 657.

[4] R. W. Hamming, Error detecting and correcting codes, *Bell System Tech. J.* **29** (1950) 147–160.

[5] A. Liu, In Search of a Missing Link: A Case Study in Error-Correcting Codes, *Math. Mag.*, **32** (2001) 343–347.

[6] A. Liu, A Magic Trick with Eight Coins, *8th Gathering for Gardner Exchange Book*, Vol. 1 (2006) 131–133.

[7] A. Liu, Two Applications of a Hamming Code, *Coll. Math. J.* **40** (2009) 2–5.

[8] H.-S. Liu and A. Liu, Error-correcting codes (in Chinese), *Math. Media* **91** #3 (1999) 59–63.

[9] D. E. Muller, Application of Boolean algebra to switching circuit design and to error detection, *IEEE Trans. Comput.* **3** (1954) 6–12.

[10] M. Rabenstein, An example of an error correcting code, *Math. Mag.* **58** (1985) 225–226.

[11] I. S. Reed, A class of multiple-error-correcting codes and the decoding scheme, em IEEE Trans. Inf. Theory **4** (1954) 38–49.

[12] D. E. Shasha, *Puzzling Adventures*, W. W. Norton, New York, (2005) 35–37 and 160–163.