Liar Games and Coding Theory

Rachel Watkinson 328506
School of Mathematics,
University of Birmingham

March 27, 2006

I warrant that the content of this dissertation is the direct result of my own work and that any use made in it of published or unpublished material is fully and correctly referenced.

Signed ..... **Date**.....

Abstract

In a liar game we have two players, the Questioner and the Responder. The Responder thinks of an integer x between 1 and n and the Questioner attempts to identify x in q questions, however, when the Responder replies to the questions they are allowed to tell up to k lies. This project discusses strategies for which it is possible to solve the liar game using the minimum number of questions and further shows links between the liar game and coding theory.

The project begins by highlighting the different types of liar games and the particular liar game that is of focus within this project. The basic concepts of coding theory are introduced in the second chapter using the work of Welsh [11]. In particular, we will see that there is a code which optimally solves the liar game with one lie and $n = 10^6$. Using Spencer [8], in Chapter 3 it is shown that it is possible to solve asymptotically a general version of the liar game for the parameters n,q,k, which represent the size of the search space, the number of questions and the number of lies respectively. Chapter 4 first provides various different strategies of solving the liar game with one lie and $n = 10^6$, one of which is optimal. This chapter further discusses a result by Pelc [3] regarding the minimum number of questions required to solve the liar game with one lie and arbitrary n. For n, where n is a power of 2, we give a simple proof of a result of Pelc which determines the minimum number of questions required by the Questioner when 1 lie is allowed.

Contents

Chapter 1

Introduction

Consider a game between two players, known as the Questioner and the Responder, where the Responder chooses an integer x from a search space of size n and within q questions the Questioner must determine what the value of x is to win the game, otherwise the Responder wins. In the twentieth century two mathematicians, Rényi [5] and Ulam [10], independently discussed the above two player game except with a variation - what happens if the Responder is allowed to lie when answering some of the questions? This type of game is referred to as a liar game or a searching game with errors[1].

In [4], Pelc sets out four principles in which the Questioner and the Responder must agree upon before play of the liar game begins. These are:

1. The size of the search space, denoted by n, and the number of questions q in which the Questioner must determine the integer x to win the game,
2. The form of limitation on the way in which the Responder is permitted to lie,
3. The format of the questions that the Questioner asks,
4. The degree of interactivity between the Questioner and Responder during play[2].

For different variations of the liar game, we consider different limitations on the lies permitted and the format of the questions asked.

Referring to item 2, there exist several different ways in which it is possible to define limitations upon the number of lies the Responder is permitted to tell. One option is to fix an upper bound on the number of lies k the Responder can use. For example, in Ulam's description of the liar game, Ulam fixes k such that the Responder is allowed to lie once or twice. Rényi's most common description of the liar game provides an alternative limitation on the number of lies permitted. Rényi suggests that the Responder lies at most a given percentage of the total number of questions i.e. if the game lasts q questions, as agreed in item 1, the Responder can lie at most pq times, where $0 \leq p < 1$, such that p is fixed. A slight variation of this limitation could be to specify that the Responder is allowed to lie p times within the first j questions and must tell the truth thereafter i.e. the Responder is allowed to lie pj times in the first j questions, where $0 \leq p < 1$, and

[1] We will refer to this type of game as the liar game throughout this project.

[2] We will assume that the Questioner receives a reply to each question from the Responder before asking the next question.

must not lie to the Questioner in the remaining $q-j$ questions. The study of this type of limitation and Rényi's limitation were initially discussed by Pelc [2] and were then later discussed by Spencer and Winkler in [9]. Alternatively, the Responder could adopt a strategy of random lies. This type of limitation on the liar game was the earliest studied way of limiting the lies permitted, which was also proposed by Rényi [5]. Suppose that the Responder has a biased coin, such that the probability p of tossing a tail is $p < \frac{1}{2}$. The Responder's lying strategy could now be if the coin, when tossed, displays a tail the Responder can choose to lie or not, but must not lie if the coin displays a head i.e. each lie is independent of the other lies and occurs with probability $p < \frac{1}{2}$. Here we consider a search space of size n, which is divided into the subsets A_i and the Questioner asks questions of the form "Is $x \in A_i$?". Rényi's interest in this form of the liar game was to determine the minimum number of questions q that was sufficient to determine x. In [5], Rényi showed

$$q = \frac{\log n + o(\log n)}{1 - I(p)},$$

where[3] $I(p) = -p\log(p) - (1-p)\log(1-p)$ is the entropy of the probability distribution $(p,\, 1-p)$. This is analogous to Shannon's Noisy Coding Theorem, discussed in Chapter 2 Section 2.2.5. Note also that if $p = 0$, then this reduces to $q = \log n$, the number of questions which suffice to solve the liar game if no lies are told.

Similarly, as there exist variations in the limitations of lying, there also exist variations in which the Questioner can ask questions. The most obvious form of questioning is asking questions of the form "Is $x \in A_i$?", where A_i forms part of some partition of n. These types of questions require answers of the form "Yes" or "No". A further variation to this type of questioning is adding in the requirement that each question can not be asked more the once. Alternatively, variable cost questions could be adopted. The idea behind variable cost questions is such that the Questioner receives a fixed budget and the answers "Yes" and "No" are given weights. Each time one of these answers appears, its weight is removed from the budget. For example, every "Yes" answer the Questioner receives, they are charged for it, but they are not charged for "No" answers and the Responder wins if the Questioner uses more than q questions or if the Questioner exceeds the budget. This form of questioning was discussed by Sereno [6].

The liar game that is focused on within this project is the one described by Ulam [10]. We shall consider a two player game, with the players the Questioner and the Responder. In the most restricted version, the Responder chooses an integer x from the set $\{1, \ldots, 10^6\}$ i.e. $n = 10^6$, and during play of the game is allowed to lie at most once i.e. $k \leq 1$. It will be shown that the Questioner can find x using at most $q = 25$ questions, that provide answers of the form "Yes" or "No"(In the well known "Twenty Questions" game, which has $n = 10^6$ with $k = 0$ i.e. the liar game with out the use of any lies, one needs at most 20 questions to identify x). In Chapter 4, we aim to show why the value of q is minimal and formulate strategies in which the Questioner should adopt in order to win the game. In 1984, Spencer [7] commented that "it seems very difficult to determine whether the answer to Ulam's original problem is twenty-five or twenty-six". The main aim of this chapter is to provide a strategy which determines that $q = 25$ is the answer using a result which is shorter than that given by Pelc [3].

[3]Note that all of the logarithms discussed within this project are to the base 2.

Further, the liar game is generalised to one lie and to a search space of size n, where n is defined to be a power of 2. Again we aim to give a simple optimal strategy for this case and give a proof of its optimality, which is considerably shorter than existing proofs.

As an alternative approach to solving the liar game, it is possible to use error-correcting codes to identify the integer x. By considering the lies as equivalent to errors in a code, one is able find the integer x using coding theory and its applications. Chapter 2 provides an introduction to the theory of coding and as an application, discusses the maximal number of questions in which it is possible to identify x using error-correcting codes. This chapter aims to find the existence of a 1 error-correcting code that can be used to optimally solve the liar game with one lie and $n = 10^6$.

As an extension to the liar game discussed in Chapter 1, in Chapter 3 we consider a game in which one has the parameters n, q, k, where n is the size of the search space, q represents the number of questions allowed and k the number of lies the Responder is allowed to tell. Amongst other results, we describe the proof of a result of Spencer [8], which implies that the minimum number of questions q required to identify the answer x (using an optimal strategy) lies between $\log n + k \log(\log n) - c''$ and $\log n + k \log(\log n) + c'$, where c' and c'' are constants dependent only on the number of lies k. Note that if n is large and k is small, this gives an asymptotic solution to the game.

Chapter 2

Coding Theory

This chapter will discuss the effects and results of communication through noisy channels. This chapter is based upon the book 'Codes and Cryptography' by Welsh [11].

It is possible to link liar games with results from coding theory, where communication occurs through a noisy channel. By using error correcting codes it is possible to find a winning strategy for the Questioner in the liar game.

When a coded message is sent by Person A, for some reason the coded message maybe distorted in places by the channel it passes through, so that the message received by Person B contains errors. This channel is known as a **noisy channel**.

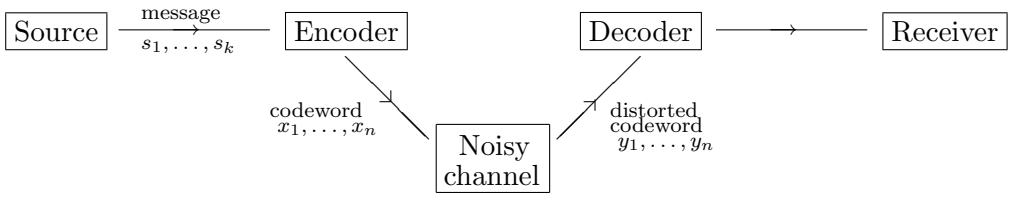


Figure 2.1: Noisy communication channel

In a liar game, the distortion in the received message would be considered as lies told during the play of the game. Hence, by considering the lies of a liar game as errors, it is possible to apply aspects of coding theory to find a winning strategy for the Questioner.

2.1 Some definitions

To begin with it is important to provide a series of definitions to help understand the concepts of coding theory that are discussed within this chapter.

Definition 2.1.1 *Given a random variable X, that takes a finite set of values with probabilities $p_1, \ldots, p_n$, the* ***uncertainty*** *or* ***entropy*** *of X is*

$$H(X) = -\sum_{k=1}^{n} p_k \log p_k \qquad (0 < p_k < 1) \text{ and } \sum_{i=1}^{k} p_i = 1,$$

where we are using logarithms to base 2.

Example 1 *Determine which scenario has the greater uncertainty:*
i) An eight lane athletics race, in which five runners have the probability of $\frac{1}{10}$ *of winning and three runners have the probability of* $\frac{1}{6}$ *of winning;*
ii) A six lane swimming race, where two swimmers have the probability of $\frac{1}{4}$ *of winning and four swimmers have the probability of* $\frac{1}{8}$ *of winning.*

Solution*:*
i) Define the athletes probability of winning the race by the random variable X. Then

$$\begin{aligned} H(X) &= 3 \times \left(-\frac{1}{6}\log\left(\frac{1}{6}\right)\right) + 5 \times \left(-\frac{1}{10}\log\left(\frac{1}{10}\right)\right) \\ &= \frac{1}{2}\log 6 + \frac{1}{2}\log 10 \\ &= \frac{1}{2}\log 60 \\ &\approx 0.859. \end{aligned}$$

ii) Define the swimmers probability of winning the race by the random variable Y. Then

$$\begin{aligned} H(Y) &= 2 \times \left(-\frac{1}{4}\log\left(\frac{1}{4}\right)\right) + 4 \times \left(-\frac{1}{8}\log\left(\frac{1}{8}\right)\right) \\ &= \frac{1}{2}\log 4 + \frac{1}{2}\log 8 \\ &= \frac{1}{2}\log 32 \\ &\approx 0.753. \end{aligned}$$

Since $H(X) > H(Y)$*, the outcome of the athletics race is more uncertain than the outcome of the swimming race.*

Suppose that $\mathbf{U}$ and $\mathbf{V}$ are random vectors. The **conditional entropy** of $\mathbf{U}$ given $\mathbf{V}$ is defined

$$H(\mathbf{U}|\mathbf{V}) = \sum_j H(\mathbf{U}|\mathbf{V} = \mathbf{v}_j) Pr(\mathbf{V} = \mathbf{v}_j),$$

where the sum is over the finite range of values $\mathbf{v}_j$ that $\mathbf{V}$ has a positive probability of taking. The **information** about $\mathbf{U}$ conveyed by $\mathbf{V}$ is defined to be the quantity

$$I(\mathbf{U}|\mathbf{V}) = H(\mathbf{U}) - H(\mathbf{U}|\mathbf{V})$$

i.e. $I(\mathbf{U}|\mathbf{V})$ measures the amount of uncertainty about $\mathbf{U}$ that is removed by $\mathbf{V}$.

Example 2 *Suppose that the weather affects the ability of an athlete to win the race in Example 1 i). On the day of the race the temperature is* $29°C$*. Now six runners have the probability* $\frac{1}{12}$ *of winning and two runners have the probability* $\frac{1}{4}$ *of winning.*

Define the affects of weather on the athletes ability by the random variable Z. Then we have

$$\begin{aligned} H(X|Z) &= 6 \times \left(-\frac{1}{12}\log\left(\frac{1}{12}\right)\right) + 2 \times \left(-\frac{1}{4}\log\left(\frac{1}{4}\right)\right) \\ &= \frac{1}{2}\log 12 + \frac{1}{2}\log 4 \\ &\approx 0.841. \end{aligned}$$

Therefore, the information is

$$\begin{aligned} I(X|Z) &= H(X) - H(X|Z) \\ &\approx 0.859 - 0.841 = 0.018. \end{aligned}$$

A **source** is a stream of symbols from some finite alphabet. Let X_i denote the i^{th} symbol produced by a source. For each symbol a_j, the probability

$$Pr(X_i = a_j) = p_j,$$

is independent of i and is independent of all other symbols emitted. Thus the random variables $X_1, X_2, \ldots, X_n$ are independently identically distributed. This is known as a **memoryless source**. By Definition 2.1.1, the entropy of a memoryless source is

$$H = -\sum_j p_j \log p_j,$$

where $0 < p_j < 1$.

If φ is a memoryless source that emits symbols from a known alphabet $\mathbb{S} = \{s_1, \ldots, s_N\}$, with the corresponding probabilities $\{p_1, \ldots, p_N\}$, then the elements of $\mathbb{S}$ are called **source words**. A **message** is any finite string of source words thats are successively encoded, transmitted and then decoded.

A **code** f, over a finite alphabet Σ, is a collection of sequences of symbols from Σ. An **encoding** is a map f from $\{s_1, \ldots, s_N\}$ into Σ^*, where Σ^* is a collection of finite strings of symbols from the alphabet Σ. The code f is **uniquely decipherable** if any finite string from Σ^* is the image of at most one message i.e. f is injective. The strings $f(s_i)$ are called **codewords** and the integers $|f(s_i)|$ are the **word lengths** of f. Further, the **average length** of a code f is

$$\langle f \rangle = \sum_{i=1}^{N} p_i |f(s_i)|.$$

We describe f as an **instantaneous** or **prefix code** if there exist no distinct s_i and s_j, such that $f(s_i)$ is a prefix of $f(s_j)$. It follows by definition that instantaneous codes are uniquely decipherable.

Example 3 *Suppose* $\Sigma = \{0, 1\}$ *and that there exist three source words* s_1, s_2, s_3. *An instantaneous code is*

$$f(s_1) = 1 \qquad f(s_2) = 01 \qquad f(s_3) = 001$$

The message 00101101001101 would be decoded as $s_3 s_2 s_1 s_2 s_3 s_1 s_2$, *since*

$$\underbrace{001}_{s_3}\,\underbrace{01}_{s_2}\,\underbrace{1}_{s_1}\,\underbrace{01}_{s_2}\,\underbrace{001}_{s_3}\,\underbrace{1}_{s_1}\,\underbrace{01}_{s_2}$$

This type of code is known as a ***comma code****, where the 1 indicates the end of a word.*

2.2 Communication through a noisy channel

2.2.1 Discrete Memoryless Channel

A communication channel is essentially a channel that accepts strings of symbols from an input alphabet[1] and ejects strings of symbols from an output alphabet.

Definition 2.2.1 *Define the input alphabet to be* $\Sigma_\alpha = \{\alpha_1, \ldots, \alpha_r\}$ *and the output alphabet* $\Sigma_\beta = \{\beta_1 \ldots, \beta_s\}$. *Further, define the* ***channel matrix*** $P = (p_{ij} : 1 \leq i \leq r,\ 1 \leq j \leq s)$. *These three conditions form the definition of a* ***discrete memoryless channel.***

We note that the input alphabet is certainly discrete since $|\Sigma_\alpha| = r \neq \infty$. Similarly, this can be shown for the output alphabet.

The channel operates by taking a sequence $(x_1, \ldots, x_n)$ of symbols from Σ_α and inputting them into the channel. The output sequence is a string of symbols $(y_1 \ldots, y_n)$ of the same length with $y_k \in \Sigma_\beta$. This has the associated probability

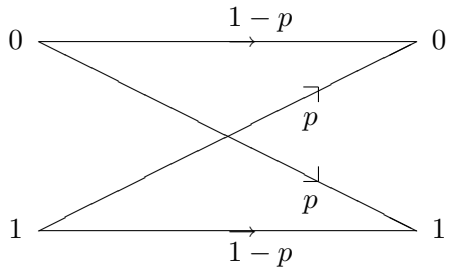
$$Pr(y_k = \beta_j | x_k = \alpha_i) = p_{ij} \qquad (1 \leq i \leq r,\ 1 \leq j \leq s),$$

independently for all k. As with all probabilities $\sum_j p_{ij} = 1$. A matrix with only non-negative entries and with row sum equal to 1 is called a **stochastic matrix**. Since none of the entries of the channel matrix P will be negative and the summation of the row is equal to one, the channel matrix P is a stochastic matrix.

It helps to represent the channel of communication in diagrammatic form. The **binary symmetric channel** is often used when representing the model of communication we are interested in.

> The binary symmetric channel ... is a discrete memoryless channel with input and output alphabets $\Sigma = \{0, 1\}$ and with channel matrix P given by

$$P = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$



> In other words, there is a common probability p of any symbol being transmitted incorrectly, independently for each symbol transmitted. [2]

[1] The input alphabet is regarded as a source.

[2] [11] page 29.

Suppose that we have a memoryless source φ that emits source words $s_1, s_2, \ldots, s_N$, with the respective probabilities $p_1, p_2, \ldots, p_N$. Encode these source words into binary, assuming that encoding into binary is noiseless[3]. Connect the encoded source to a binary symmetric channel with the error probability p. Figure 2.2 represents this diagrammatically.

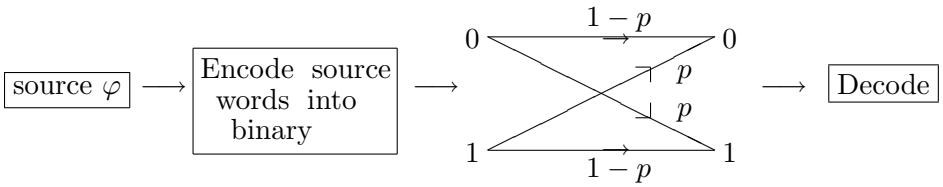


Figure 2.2: Connecting the source to a binary symmetric channel

Suppose that we have a source with 8 source words, such that

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
↓	↓	↓	↓	↓	↓	↓	↓
000	001	010	100	011	110	101	111

Figure 2.3: A possible encoding for 8 source words

By the above encoding and the fact that $1-p$ represents the probability of a source word being transmitted correctly, the probability that any source word is transmitted correctly is $(1-p)^3$. It follows that a message of n source words being transmitted correctly is $(1-p)^{3n}$.

It is possible to improve the probability of the correct transmission of a message. Suppose that we have the same source words $s_1, \ldots, s_8$, as in Figure 2.3, and assume that they all have equal probability of being used by the channel i.e. $Pr(s_1) = \ldots = Pr(s_8) = \frac{1}{8}$. Consider the same encoding as before but instead now "double up" the encoding to give:

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
↓	↓	↓	↓	↓	↓	↓	↓
000000	001001	010010	100100	011011	110110	101101	111111

Figure 2.4: "Double up" encoding

The decoder now applies the rule that they only decode when the first three symbols and the second three symbols of the codeword agree. If they do not agree, then the decoder asks for *"help"*[4]. The probability of an error occurring and remaining unnoticed is reduced, however, the rate at which transmission occurs is also reduced. Therefore, the decoder must decide whether the accuracy of a message or the rate in which they receive it, is of greater importance to them.

[3] Noiseless means that there exists no error when encoding, during transmission or decoding.

[4] "Help", for example, could be asking the source to resend the message.

2.2.2 Codes and decoding rules

Definition 2.2.2 *Given a code ψ of length n with codewords $\mathbf{c}_1, \ldots, \mathbf{c}_N$, a **decoding rule** is any partition of the set of possible received sequences into disjoint sets $R_1, \ldots, R_N$. If the received sequence $\boldsymbol{y} \in R_j$, then it is decoded as the codeword $\mathbf{c}_j$.*

The choice of the decoding rule adopted by the decoder is important to the success of a communication system. The decoder should aim to use a decoding rule that attempts to minimise the chance of error. This implies decoding any received vector $\mathbf{y}$ into a codeword $\mathbf{c}_j$, such that

$$Pr(\mathbf{c}_j \text{ sent}| \mathbf{y} \text{ received}) \geq Pr(\mathbf{c}_i \text{ sent}| \mathbf{y} \text{ received})$$

i.e. take the codeword $\mathbf{c}_j$ with the largest probability of being decoded correctly given that the decoder received vector $\mathbf{y}$. This is known as the **Ideal Observer** or **Minimum Error rule**. It is important to highlight that this rule cannot be used without prior knowledge of the probabilities of the codewords $\mathbf{c}_j$. Furthermore, the Minimum Error rule is not easy to use for a large number of codewords. Thus the Minimum Error decoding rule is quite difficult for the decoder to use as a method of decoding. Therefore, we consider using a rule known as **Maximum Likelihood decoding rule**. This rule decodes a received vector $\mathbf{y}$ into a codeword $\mathbf{c}_j$, such that it maximises

$$Pr(\mathbf{y} \text{ received}|\mathbf{c}_j \text{ sent}).$$

This rule only requires us to look for the codewords that differ slightly from the vector received i.e. it reduces the number of codewords in which the decoder was previously required to look at when using the Minimum Error rule. We note that if the codewords all have equal probability, then the Maximum Likelihood decoding rule is the same as the Minimum Error rule.

2.2.3 Hamming Distance

Definition 2.2.3 *Let V_n denote the set of all n-sequences of 0's and 1's. Consider V_n as an n-dimensional vector space over the field of integers modulo 2. Let $\boldsymbol{x}, \boldsymbol{y} \in V_n$. The **Hamming distance** $d(\boldsymbol{x}, \boldsymbol{y})$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined to be the number of places in which the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ differ.*

Consider the **Minimum Distance decoding rule**. The Minimum Distance decoding rule decodes any received vector $\mathbf{y}$ into a codeword $\mathbf{c}_j$ that has a minimum Hamming distance from $\mathbf{y}$. If there exists more than one such codeword, pick one of these codewords arbitrarily. When considering the binary symmetric channel, the Minimum Distance decoding rule is the obvious choice of decoding rule to apply.

Theorem 1 *For a binary symmetric channel, with error probability $p \leq \frac{1}{2}$, the Minimum Distance decoding rule is equivalent to the Maximum Likelihood decoding rule.*

Proof Let $\mathbf{x}, \mathbf{y} \in V_n$ with the Hamming distance $d(\mathbf{x}, \mathbf{y}) = d$, [5]

$$Pr(\mathbf{y} \text{ received}| \mathbf{x} \text{ sent}) = p^d(1-p)^{n-d}.$$

[5]The probability describes how the received vector differs from the sent vector by looking at the error probability, which occurs d times, and the probability of the vector being sent correctly which occurs $n - d$ times.

When $p \leq \frac{1}{2}$, [6] the above probability is maximised when d is the minimum Hamming distance. Therefore, the Minimum Distance decoding rule is the same as the Maximum Likelihood decoding rule. □

2.2.4 Capacity of a channel

The **capacity** of a communication channel is a measure of the channel's ability to transmit information.

Suppose that our communication channel is a discrete memoryless channel with input alphabet $\Sigma_\alpha = \{\alpha_1, \ldots, \alpha_r\}$ and output alphabet $\Sigma_\beta = \{\beta_1, \ldots, \beta_s\}$, with the channel matrix $P = [p_{ij}] = Pr(\beta_j \; received | \; \alpha_i \; sent)$. Attach a memoryless source φ to this channel, which emits symbols $\alpha_1, \ldots, \alpha_r$ with the respective probabilities $p_1, \ldots, p_r$. Then the output of this channel can be seen as the memoryless source ζ. ζ emits the symbols $\beta_1, \ldots, \beta_s$ with the respective probabilities $q_1, \ldots, q_s$ where,

$$\begin{aligned} q_j &= \sum_{i=1}^{s} Pr(\beta_j \; received | \; \alpha_i \; sent) Pr(\alpha_i \; sent) \\ &= \sum_{i=1}^{s} p_i p_{ij}. \end{aligned}$$

The information about φ given ζ is $I(\varphi|\zeta)$, where

$$\begin{aligned} I(\varphi|\zeta) &= H(\varphi) - H(\varphi|\zeta) \\ &= H(\varphi) + H(\zeta) - H(\varphi, \zeta) \end{aligned}$$

is a function of the source distribution $(p_1, \ldots, p_r)$ and the channel matrix P.

Definition 2.2.4 *The **capacity** C of the channel is*

$$C = \sup I(\varphi|\zeta) \tag{2.1}$$

The supremum is taken over all possible input distributions $(p_1, \ldots, p_r)$.

Note that (2.1) can be rewritten as $C = \max I(\varphi|\zeta)$ [7].

Theorem 2 *The capacity of the binary symmetric channel, in which there is probability p of error, is given by*

$$C(p) = 1 + p \log p + (1-p) \log(1-p) \tag{2.2}$$

Proof See Appendix A.

[6] Take the logarithm of $p^d(1-p)^{n-d}$, such that we have $d \log p + (n-d)\log(1-p)$. Now differentiate with respect to d and set the equation equal to 0. This implies $p = \frac{1}{2}$. This is the maximum value p can take, hence $p \leq \frac{1}{2}$.

[7] C is well defined in the sense that we are seeking the supremum of $f(p)$. f is a continuous function on a closed and bounded subset (the code is discrete and the supremum bounds) of $\mathbb{R}^r$. Using a fundamental theorem of analysis, any continuous function on such a set attains its supremum on the set i.e. the maximum of the set equals the supremum. Hence, (2.1).

2.2.5 Shannon's Noisy Coding Theorem

Definition 2.2.5 *Given a code ψ and any decoding scheme for ψ, the* ***maximum error probability*** *is defined*

$$\hat{e}(\psi) = \max_i Pr(\, error|\ \mathbf{c}_i\ transmitted),$$

where $\mathbf{c}_i$'s are the codewords of ψ.

Obviously we want codes with small error probability and large transmission rate, Shannon's theorem tells us that such codes exist.

Shannon's Theorem *Given a binary symmetric channel of capacity C and any R, such that $0 < R < C$, then if $(M_n; 1 \leq n < \infty)$ is any sequence of integers satisfying*

$$1 \leq M_n \leq 2^{Rn} \tag{2.3}$$

and $\epsilon > 0$, there exists a sequence of codes $(\psi_n : 1 \leq n < \infty)$ and an integer $N_0(\epsilon)$, with ψ_n having M_n codewords of length n and with the maximum error probability

$$\hat{e}(\psi_n) \leq \epsilon \qquad\qquad for\ all\ n \geq N_0(\epsilon).$$

How does Shannon's theorem work? Suppose that we have an error probability such that the channel has capacity $C(p) = 0.70$. Further, suppose that we have a message that we wish to send, which consists of a string of 0's and 1's. By Shannon's theorem, for n sufficiently large and taking $R = \frac{3}{5}$, it follows that there exists a set of $2^{\frac{3}{5}n}$ codewords of length n that have an error probability less than the one given to begin with. In order to encode the message stream from the source, one should use the following steps:

1. Take the message stream and split it into blocks of length m, where m is an integer, such that
$$3\left\lceil \frac{1}{5}n \right\rceil = m \geq \frac{3}{5}N_0(\epsilon),$$
2. Using a codeword of length $\frac{5}{3}m$, encode each m-block into the code ψ_n,
3. Transmit the encoded stream through the channel.

What does Shannon's theorem tell us? Shannon's theorem shows, provided the rate of transmission R is below the channel capacity C, it is possible to achieve arbitrarily high reliability. Hence, by using Shannon's theorem it is possible to obtain a marked reduction in the error probability i.e. Shannon's theorem show that such codes with small maximum error probability exist. The theorem provides a more theoretical result, rather than a practical one. The theorem sets bounds but it only tells us that good codes do exist.

2.3 Error-correcting codes

Although Shannon's theorem tells us that good codes exist[8] it does not tell us how to find them. This section indicates possible different approaches to the problem of finding these codes.

[8] i.e. they have small maximum error probability.

Without loss of generality, assume that the channel has the same alphabet Σ of size q, for both the input and output alphabets. Let ψ be a code over Σ and assume that all the codewords of ψ are of equal length [9]. The use of this type of code makes decoding easier.

2.3.1 Some preliminaries

If the codewords of ψ have length n and $|\Sigma| = q$, then the code is known as a **q-ary code of length** n. We will only consider $q = 2$ i.e. binary codes of length n. The set of all n-sequences of symbols from the alphabet Σ is denoted $V_n(\Sigma)$. The elements of $V_n(\Sigma)$ are called **vectors** or **words**.

Theorem 3 *If a code has minimum distance d, then the Minimum Distance decoding rule will correct up to $\frac{1}{2}(d-1)$ errors.*

Proof Let $e = \lfloor \frac{1}{2}(d-1) \rfloor$ and consider a sphere of radius e about $\mathbf{x}$, where $\mathbf{x}$ is a codeword. This is the set $S_e(\mathbf{x})$ given by

$$S_e(\mathbf{x}) = \{y : \ d(\mathbf{x}, \mathbf{y}) \leq e\}.$$

If $\mathbf{x}$ and $\mathbf{z}$ are distinct codewords, the minimum distance hypothesis implies

$$S_e(\mathbf{x}) \cap S_e(\mathbf{z}) = \emptyset$$

i.e. $\mathbf{x}$ and $\mathbf{z}$ are at least $2e$ apart. This implies that minimum distance decoding will correct up to e errors. □

Definition 2.3.1 *If a code has M codewords of length n and has a minimum distance d, then it is called an (n, M, d)-code.*

Let $A(n, d)$ denote the maximum number of codewords M, such that there exists a binary (n, M, d)-code.

Theorem 4 *When d is an odd integer,*

$$A(n, d) = A(n+1, \, d+1) \tag{2.4}$$

Proof Assume that d is an odd integer. Firstly we wish to show that

$$A(n, d) \leq A(n+1, \, d+1) \tag{2.5}$$

Suppose we take a code ψ that has the parameters $n+1$ and $d+1$. By deleting the last digit from all of the codewords of ψ, clearly we reduce the length of the code by one and we at most reduce the minimum distance by 1. Hence (2.5) holds.

Further, we wish to show that

$$A(n, d) \geq A(n+1, \, d+1) \tag{2.6}$$

Suppose we take a code ψ that has the parameters n and d. By adding a parity check digit to all of the codewords of ψ we obtain the new code ψ'. All the codewords of ψ'

[9] Such codes are known as **block codes**.

have even weight, which implies that the minimum distance d must be even. Since d is odd, we have shown that (2.6) holds. Since (2.5) and (2.6) both hold, this implies (2.4). □

Suppose that you have a code ψ that is an (n, M, d)-code. ψ can be represented by an $M \times n$ array, whose rows are distinct codewords. If a code ψ' can be obtained from a code ψ by a sequence of permutations, then ψ' is called an **equivalent code**.

Definition 2.3.2 *Suppose ψ is a code over $V_n(\Sigma)$, such that for some $t > 0$, the t-spheres around the codewords of ψ are disjoint but their union contains every vector in $V_n(\Sigma)$. A code such as this is called a **perfect code**.*

Suppose ψ is a code which has minimum distance $d = 2e + 1$. By Theorem 3 it is possible to correct up to e errors by the nearest neighbour decoding rule. If $|\psi|$ is large, this method can be time consuming, since it requires comparing a large number of codewords with the received vector. Linear codes offer a solution to this.

Definition 2.3.3 *A **linear code** ψ over Σ is defined to be any subspace of $V_n(\Sigma)$. If ψ is a k-dimensional subspace, then the code is known as [n, k]-code or if the minimum distance d is known, ψ is known as an [n, k, d]-code.*

Definition 2.3.4 *A **generator matrix**, for a linear [n, k]-code ψ, is any $k \times n$ matrix, whose rows constitute of k linearly independent codewords of ψ.*

Theorem 5 *If ψ is any linear [n, k]-code, then there exists an equivalent code ψ' with generator matrix $[I_k, A]$, where I_k is the $k \times k$ identity matrix.*

Proof Omitted.

By using the generator matrix G of the linear $[n, k]$-code ψ, we are able to encode the source words $\mathbf{s}$ of ψ into codewords $\mathbf{c}$ by

$$\mathbf{c} = \mathbf{s}^T G \tag{2.7}$$

The first k digits of a codeword are often the **message digits** and the remaining $n - k$ are the **parity check** digits.

Example 4 *Suppose that ψ is a [7,4]-linear code with generator matrix*

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

By performing a finite sequence of operations[10] on the rows and columns of the generator matrix, by Theorem 5, we obtain[11]

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

[10]The operations that are performed are i) Permuting rows/columns, ii) Multiplying rows/columns by a non-zero scalar iii) adding to a row a scalar multiple of another row.

[11]See Appendix B.

Consider the source words $(0\,1\,1\,0)$, $(0\,1\,0\,1)$, $(1\,0\,1\,1)$. *Using our generator matrix* G *and (2.7), we obtain the codeword* $(0\,1\,1\,0\,1\,1\,0)$ *from the source word* $(0\,1\,1\,0)$ *i.e.*

$$\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Similarly, $(0\,1\,0\,1)$ *is encoded into the codeword* $(0\,1\,0\,1\,0\,1\,0)$ *and* $(1\,0\,1\,1)$ *is encoded into the codeword* $(1\,0\,1\,1\,0\,1\,0)$.

Suppose that we wish to send the message

$$0\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0$$

By splitting the message up into blocks of size 4, such that

$$0\,1\,0\,1 \,|\, 1\,0\,1\,1 \,|\, 0\,1\,1\,0$$

and encoding these blocks, we send the encoded message

$$0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0$$

This increases the length of the message, which decreases the rate of transmission, however, it increases the reliability of the message sent.

Theorem 6 *The minimum distance of a linear code* ψ *is the minimum weight of a non-zero vector in* ψ.

Proof Let d be the minimum distance of ψ and suppose that $\mathbf{x}$ and $\mathbf{y}$ are codewords, with $d(\mathbf{x}, \mathbf{y}) = d$. Since ψ is a linear subspace, the vector $\mathbf{x} - \mathbf{y}$ is also a codeword of ψ. But the weight of $\mathbf{x} - \mathbf{y}$ is $w(\mathbf{x} - \mathbf{y}) = d$, showing that the minimum weight is at most d. However, it can not be strictly less than d since, if $\mathbf{z} \neq \mathbf{0}$, such that the $w(\mathbf{z}) < d$

$$\Rightarrow \qquad w(\mathbf{z}) = d(\mathbf{z}, \mathbf{0}) < d,$$

which is a contradiction to d being the minimum distance of the linear code ψ. $\square$

Definition 2.3.5 *The matrix*

$$H = [-A^T, I_{n-k}]$$

is called the ***parity check matrix****.*

Claim: *A vector* $\boldsymbol{z}$ *is a codeword of* ψ *if and only if*

$$H\boldsymbol{z}^T = \boldsymbol{0}.$$

Proof Let G be the generator matrix for a linear $[n, k]$-code ψ. Since permuting the rows and columns of G does not affect the codewords, it is possible to rewrite G such that

$$G = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_k \end{bmatrix} = [I_k, A],$$

where $\mathbf{r}_1, \ldots, \mathbf{r}_k$ are the linearly independent vectors of ψ and $\mathbf{A}$ is a $k \times (n-k)$ matrix.

Suppose that the message sequence is given by $\mathbf{s} = (s_1, \ldots, s_k)$. Encode $\mathbf{s}$ by the codeword $\mathbf{c} = (c_1, \ldots, c_n)$ i.e.

$$\mathbf{c} = \mathbf{s}^T G = \mathbf{s}^T [I_k,\, A] = [\mathbf{s}^T,\, \mathbf{s}^T A] \tag{2.8}$$

Thus $c_i = s_i$, for all $i \leq k$. So

$$\left[\begin{array}{ccc} c_1 & \ldots & c_k \end{array}\right] \mathbf{A} = \left[\begin{array}{ccc} s_1 & \ldots & s_k \end{array}\right] \mathbf{A} \overset{(2.8)}{=} \left[\begin{array}{ccc} c_{k+1} & \ldots & c_n \end{array}\right].$$

Rearranging gives

$$\left[\begin{array}{ccc} c_{k+1} & \ldots & c_n \end{array}\right] - \left[\begin{array}{ccc} c_1 & \ldots & c_k \end{array}\right] \mathbf{A} = \mathbf{0}.$$

But

$$\left[\begin{array}{ccc} c_{k+1} & \ldots & c_n \end{array}\right] = \left[\begin{array}{ccc} c_{k+1} & \ldots & c_n \end{array}\right] I_{n-k}.$$

Hence we obtain

$$\mathbf{c}\,[I_{n-k}, -A] = \mathbf{0}$$

Taking the transpose of this gives

$$\left[-A^T, I_{n-k}\right] \mathbf{c}^T = \mathbf{0}.$$

The parity check matrix H, is defined $H = [-A^T, I_{n-k}]$. □

Thus the parity check matrix clearly defines the code equally as well as the generator matrix.

2.3.2 Binary Hamming codes

Definition 2.3.6 *Let r be a positive integer and take $n = 2^r - 1$. Take H to be the $r \times (2^r - 1)$ matrix, whose columns are the distinct non-zero vectors of V_r. Then H is the parity check matrix of a binary $[n, k]$-code, where*

$$n = 2^r - 1 \quad \textit{and} \quad k = n - r.$$

This code is called the [n, k] ***Hamming code****.*

Lemma 2.3.7 *Let $\boldsymbol{z}$ be a codeword of a code ψ i.e. for a parity matrix H of ψ*

$$H\boldsymbol{z}^T = \boldsymbol{0}.$$

Suppose we receive a vector $\boldsymbol{z}_*$ such that, $\boldsymbol{z}_* = \boldsymbol{z} + \boldsymbol{e}_j$, where the error vector

$$\boldsymbol{e}_j = (0 \ldots 1 \ldots 0),$$

where the 1 appears in the j^{th} position i.e. one error was made during transmission. Then $H\boldsymbol{z}_*^T = j^{th}$ column of H. This implies we can use the Minimum Distance decoding rule, which works if there is exactly one error.

Proof

$$\begin{aligned} H\mathbf{z}_*^T &= H(\mathbf{y}+\mathbf{e}_j)^T \\ &= H\mathbf{y}^T + H\mathbf{e}_j^T \\ &= \mathbf{0} + H\mathbf{e}_j^T. \end{aligned}$$ □

Example 5 *Let $r = 3$, then we obtain a [7,4] - Hamming code, which has the parity check matrix*

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

If we know during transmission a coded message becomes distorted by a noisy channel in precisely one place, it is possible to find the error and correct it. Consider that you receive the vector

$$\boldsymbol{z}_* = (1\ 0\ 0\ 0\ 1\ 0\ 1).$$

For $\boldsymbol{z}_*$ to be a codeword we require $H\boldsymbol{z}_*^T = \boldsymbol{0}$. However,

$$H\boldsymbol{z}_*^T = (0\ 1\ 1).$$

We notice that $(0\ 1\ 1)$ is identical to the 3^{rd} column in the parity check matrix. By using Lemma 2.3.7 we conclude that the error occurs in the 3^{rd} place of $\boldsymbol{z}_*$. Hence we decode $\boldsymbol{z}_*$ as

$$\boldsymbol{z} = (1\ 0\ 1\ 0\ 1\ 0\ 1),$$

which satisfies $H\boldsymbol{z}^T = \boldsymbol{0}$.

We know from Definition 2.3.5 that a vector $\boldsymbol{z}$ is a codeword of [7,4] - Hamming code if and only if

$$H\boldsymbol{z}^T = \boldsymbol{0}.$$

For the [7,4] - Hamming code, there exist 16 codewords, since [12]

$$\frac{2^7}{1+\binom{7}{1}} = \frac{2^7}{2^3} = 2^4.$$

Suppose that $\boldsymbol{z} = (a\ b\ c\ d\ e\ f\ g)$ is a codeword. Therefore, solving $H\boldsymbol{z}^T = \boldsymbol{0}$ implies solving

$$\begin{aligned} a+b+d+e &= 0 \\ a+c+d+f &= 0 \\ b+c+d+g &= 0, \end{aligned}$$

[12] Suppose we have a $[n,q]$-Hamming code and we wish to find out the number of codewords that exist for this code. There are 2^n possible codewords. For each codeword $\mathbf{z}$ there exist $\binom{n}{1}$ other words $\mathbf{x}$ of length n with Hamming distance $d(\mathbf{x},\mathbf{z}) = 1$. Hence the total number of codewords is expressed by the equation

$$\frac{2^n}{1+\binom{n}{1}}.$$

remembering that we are working in modulo 2.

This gives the following 16 codewords

$$\begin{aligned}
\boldsymbol{z}_0 &= (0\,0\,0\,0\,0\,0\,0)\\
\boldsymbol{z}_1 &= (1\,1\,1\,1\,1\,1\,1)\\
\boldsymbol{z}_2 &= (1\,1\,1\,0\,0\,0\,0)\\
\boldsymbol{z}_3 &= (1\,1\,0\,0\,0\,1\,1)\\
\boldsymbol{z}_4 &= (1\,0\,0\,1\,0\,0\,1)\\
\boldsymbol{z}_5 &= (1\,0\,1\,1\,0\,1\,0)\\
\boldsymbol{z}_6 &= (1\,0\,1\,0\,1\,0\,1)\\
\boldsymbol{z}_7 &= (1\,0\,0\,0\,1\,1\,0)\\
\boldsymbol{z}_8 &= (1\,1\,0\,1\,1\,0\,0)\\
\boldsymbol{z}_9 &= (0\,1\,1\,1\,0\,0\,1)\\
\boldsymbol{z}_{10} &= (0\,1\,0\,1\,0\,1\,0)\\
\boldsymbol{z}_{11} &= (0\,1\,1\,0\,1\,1\,0)\\
\boldsymbol{z}_{12} &= (0\,1\,0\,0\,1\,0\,1)\\
\boldsymbol{z}_{13} &= (0\,0\,1\,1\,1\,0\,0)\\
\boldsymbol{z}_{14} &= (0\,0\,0\,1\,1\,1\,1)\\
\boldsymbol{z}_{15} &= (0\,0\,1\,0\,0\,1\,1).
\end{aligned}$$

Example 6 *Suppose that $r = 4$, then we have a [15,11] - Hamming code, which has the parity check matrix*

$$H = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0\\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0\\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0\\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}$$

For the [15,11] - Hamming code there exist 2048 codewords, since

$$\frac{2^{15}}{1+\binom{15}{1}} = \frac{2^{15}}{2^4} = 2^{11}.$$

Suppose that

$$\boldsymbol{z} = (a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ p)$$

satisfies $H\boldsymbol{z}^T = \boldsymbol{0}$. This implies solving the equations

$$\begin{aligned}
a+b+c+g+h+i+k+l &= 0\\
a+d+e+g+i+j+k+m &= 0\\
b+d+f+g+h+j+k+n &= 0\\
c+e+f+h+i+j+k+p &= 0
\end{aligned}$$

Clearly it will take a long time to obtain all the codewords for [15,11] - Hamming code, so listed below are a sample of codewords.

$$\begin{aligned}
\boldsymbol{z}_0 &= (0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0)\\
\boldsymbol{z}_1 &= (1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1)\\
\boldsymbol{z}_2 &= (1\,1\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0)\\
\boldsymbol{z}_3 &= (1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,0)\\
\boldsymbol{z}_4 &= (0\,1\,0\,1\,0\,1\,1\,1\,0\,1\,1\,0\,0\,1\,0)\\
\boldsymbol{z}_5 &= (0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1)\\
\boldsymbol{z}_6 &= (0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1)\\
\boldsymbol{z}_7 &= (1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0).
\end{aligned}$$

The following theorem is an important property of Hamming codes.

Theorem 7 *Any Hamming code is a perfect single error correcting code.*

Proof Since ψ is a linear code, by Theorem 6, the minimum distance $d(\psi)$ equals the minimum weight vector in ψ. Suppose that ψ has a codeword $\mathbf{u}$ of weight 1, with a non-zero entry in the i^{th} position. Then

$$H\mathbf{u}^T = \mathbf{0},$$

which implies that the i^{th} column of the parity check matrix H has all zero entries. However, H does not contain a column of zero entries, hence the codeword does not have a weight of 1.

Suppose that ψ has a codeword of weight 2, with non-zero entries in the i^{th} and j^{th} positions (where $i \neq j$). Then this implies that

$$\mathbf{h}_i + \mathbf{h}_j = \mathbf{0},$$

where $\mathbf{h}_i$ denotes the i^{th} column of H. However, this implies H has two identical columns, which is not possible since $i \neq j$ [13]. Thus $d(\psi) \geq 3$ i.e. the minimum distance of a Hamming code is at least 3.

A sphere of radius 1 surrounding any codeword $\mathbf{x} \in \psi$ will contain $1 + n = 2^r$ vectors. Since ψ contains $2^k = 2^{n-r}$ codewords, the union of the spheres of radius 1 is the complete set of 2^n vectors in V_n i.e. the Hamming code is a perfect code. □

2.4 The liar game using coding theory

Consider the liar game with one lie and search space of size $n = 10^6$. The Responder chooses an integer x from the set $\{1, \ldots, 10^6\}$ and we will see that coding theory tells us that within 25 questions it is possible for the Questioner to win the game[14]. We can use applications of coding theory to solve the liar game, since if we consider the lie told in the liar game as an error, it is possible to use error-correcting codes to find x.

[13] Remember since we are working in binary, this implies we are working in modulo 2 i.e. $1 + 1 = 0$.

[14] It is shown in Chapter 4 Section 4.4 that the value $q = 25$ is the maximal number of questions required to solve this liar game.

Firstly, we consider all integers in the set $\{1, \ldots, 10^6\}$ in their binary representation. We wish encode each of these numbers into a codeword by using a code ψ, such that ψ is an (n, M, d)-code. This means that we require $M \geq 10^6$ to ensure that all of the binary numbers can be encoded. Further the code ψ must be able to correct one lie i.e. ψ must be a 1 error-correcting code. As a consequence of the (n, M, d)-code ψ being an 1 error-correcting, the code must have minimum distance $d \geq 3$. We note that the questions from the Questioner take the form "What is the i^{th} digit of the codeword?".

We have seen that a binary Hamming code is a type of error-correcting code, however, it is not suitable for solving this liar game. When using binary Hamming codes we notice that when $r = 4$, we have a binary $[15, 11]$-code that has 2048 codewords[15], which is insufficient. If we consider $r = 5$, this implies we have a binary $[31, 26]$-code which has 2^{26} codewords [16]. Clearly $2^{26} > 10^6$, however, by using the binary Hamming code with $r = 5$ to solve the liar game implies we require 31 questions to identify x, thus the code is inefficient.

If we assume that $d = 3$, we apply Theorem 2.4 to obtain $d = 4$. Hence we are looking for a (n, M, d)-code ψ, which has $M \geq 10^6$ and $d = 4$, but we are yet to determine the value of n. [12] provides a table of values for the best known bounds on $A(n, d)$. The table states $A(26, 4) \geq 1048576$ [17]. We note that $d = 4$ implies such a code is certainly 1 error-correcting. Hence there exists a code such that the liar game can be solved using at most 26 questions with $d = 4$. However, Theorem 2.4 tells us that $A(n, d) = A(n + 1, d + 1)$, where d is odd. Therefore, there even exists a 1 error-correcting code ψ with $n = 25$ and $d = 3$ that contains more than 10^6 codewords. Hence, it is possible using 1 error-correcting codes to solve the liar game, using 25 questions to identify the integer x.

However, it is not always the case that 1 error-correcting codes are the best way to solve the liar game with one lie. Using [12] we see that with $q = 27$ questions, we can distinguish between 4194304 and 4793472 codewords (since $4194304 \leq A(27, 3) \leq 4793472$) [18]. However, using a result described in Chapter 4 Section 4.4[19], if

$$\frac{2^q}{q+1} > n,$$

where n refers to the number of codewords, then there exists a better strategy to solve the liar game than by using 1 error-correcting codes. Since

$$\frac{2^{27}}{28} > 4194304,$$

there exists a better strategy to solve the liar game than that which uses the best existing

[15] See Example 6.

[16] Since

$$\frac{2^{31}}{1 + \binom{31}{1}} = \frac{2^{31}}{2^5} = 2^{26}.$$

[17] Although the lower bound is larger that 10^6, we note that if $n = 25$ and $d = 4$ the code would contain at most 599184 codewords, which is significantly less than 10^6.

[18] The result in [12] refers to $q = 28$, but we apply Theorem 2.4 to obtain $q = 27$.

[19] This result is also analogous to Theorem 8 described in Chapter 3.

codes. Further we note that

$$\frac{2^{27}}{28} > 4793472.$$

Hence in this case the desired codes cannot exist so it is inappropriate to use 1 error-correcting codes to solve the liar game with one lie and $q = 27$.

Chapter 3

An asymptotic solution for the liar game

In this chapter we are concerned with finding a winning position for the Questioner when playing a general version of the liar game, where we consider the three parameters n,q,k [1]. The results discussed in this chapter follow the results discussed by Spencer in [8].

3.1 The liar game in terms of chips

When evaluating the general version of the game we consider three parameters: n - the size of the search space; q - the number of rounds in the game i.e. the number of questions the Questioner is allowed to ask; and k - the number of lies the Responder is permitted to tell [2]. The values of each of these parameters are known to both players. The Responder thinks of an integer x that lies within the search space of size n and the Questioner has q questions in which to correctly guess the integer x. The questions asked by the Questioner must take the form "Is $x \in A_i$?", where $A_i \subseteq \{1, \ldots, n\}$. This type of game is fully adaptive game i.e. the Questioner learns the answer to each question before asking the next one. This means that the Questioner is able to use the results of previous questions before asking the next one. The Responder is allowed to apply an adversary strategy to the game when replying to questions. This is where the Responder does not actually pick an integer x at the beginning of the game but answers the questions so that there exists at least one x that they could have picked from the search space. By this condition and given n, q, k, we can conclude that either the Questioner will win the game, if they identify x correctly within q rounds, or the Responder wins, if the Questioner is unable to identify correctly the value of x. This type of game is known as a $[n, q, k]$-game.

In order to analyse the above game, we consider a more general game, where we replace the search space with a sequence of nonnegative integers $x_0, x_1, \ldots x_k$. Define A_i, $0 \leq i \leq k$, to be disjoint sets, such that $|A_i| = x_i$, and let these sets be known to both the Questioner and the Responder. The Responder initially chooses $x \in A_0 \cup A_1 \cup \ldots \cup A_k$ and if x lies in A_i, then the Responder is allowed to lie about the value of x at most

[1] Where n represents the size of the search space, q denotes the number of questions the Questioner is allowed to ask and k denotes the number of lies the Responder is permitted to tell.

[2] We will consider k as a fixed integer, with n and q which are considered to be much larger than k.

$k - i$ times in subsequent questions[3].

In [8], Spencer introduces the game in terms of chips, where we consider a board with positions marked $0, 1, \ldots, k$ from left to right. There exists one chip for each integer x. If a chip representing x is placed on position i, where $0 \leq i \leq k$, then the Responder can lie about that chip $k - i$ more times. We can represent the number of chips on each position by x_i, for each i. The Questioner selects a set A_i of chips and asks the question "Is $x \in A_i$?". If the Responder replies "No", but x was an element of A_i, then this corresponds to x being lied about one more time. This is equivalent to moving all of the elements of A_i one position to the right. The chips in A_i that were on position k are removed from the board, since they have been lied about more than k times. If, however, the Responder replied "Yes", this is equivalent to moving all of the elements not in A_i one position to the right. The Responder is not allowed to remove all chips from the board i.e. there must remain at least one chip on the board, which represents the true value of x. The Questioner wins the game if, after q questions, there is exactly one chip remaining on the board.

The state of the game is defined to be the vector $P = (x_0, \ldots, x_k)$, which in terms of the chips is the picture with x_i chips on the i^{th} position shown in Figure 3.1. Note that with every response to a question the state of the game changes.

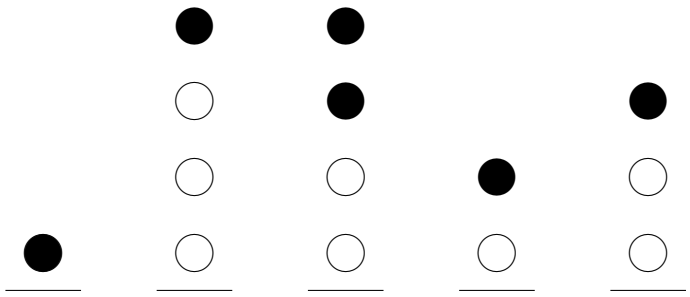


Figure 3.1: An example of the chips on the board, $P = (1, 4, 4, 2, 3)$. Suppose we have this state, where the black chips are the elements of set A_i.

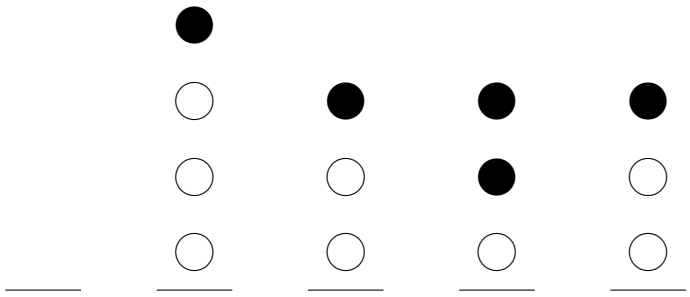


Figure 3.2: If the Responder says that x is not an element of A_i, move all chips in A_i one position to the right to obtain the above position $P = (0, 4, 3, 3, 3)$.

[3]Note that the Responder is still adopting the adversary strategy.

Define the weight of an individual chip on the i^{th} position by

$$p_i^* = Pr[B(q, 0.5) \leq k - i] = \sum_{i=0}^{k} \binom{q}{k-i} \times 2^{-q}, \tag{3.1}$$

where $B(q, 0.5)$ denotes a random variable that has a binomial distribution with parameters q, $\frac{1}{2}$. The weight of a state can be defined by the summation of the weights of the individual chips i.e.

$$w^* = \sum_{i=0}^{k} x_i p_i^* \tag{3.2}$$

The motivation to discuss the weight of a chip arises from the fact that if in each step $Pr[a\ chip\ moves\ to\ the\ right] = \frac{1}{2}$, then the weight of a chip is equal to the expectation of the number of the chips left on the board after q steps [4].

Theorem 8 *If a state has weight greater than 1 then the Responder has a winning strategy.*

Proof The Questioner asks "Is $x \in A_i$?". Suppose the Responder decides to use a random strategy i.e. the Responder flips a fair coin to decide whether to move all of the chips in A_i one position to the right or all of the chips not in A_i one position to the right[5]. The coin is flipped after each question has been asked. Let c be a chip. For each c, we define X_c to be the indicator random variable for c to remain on the board by the end of the game. Regardless of the choice of the strategy used by the Questioner, if $c \in A_i$, then c will move forward with the probability of 0.5 each round. Each movement of the chips is independent from all the previous movements of the chips. If c begins the game at position j, it follows that by the end of the game c will have moved to

1. position $j + B(q, 0.5)$ if $j + B(q, 0.5) \leq k$
2. off of the board if $j + B(q, 0.5) > k$ [6].

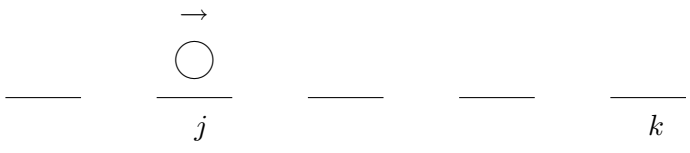


Figure 3.3: Chips moving towards the k^{th} position

Since X_c is an indicator random variable, $\mathbb{E}[X_c]$ is the probability of chip c remaining on the board.

$$\begin{aligned}
\mathbb{E}[X_c] &= Pr[chip\ remains\ on\ the\ board] \\
&= Pr[q\ coin\ tosses\ yield\ \leq k - j\ heads] \\
&= Pr[B\left(q, \frac{1}{2}\right) \leq k - j] = p_i^*.
\end{aligned}$$

[4] Refer to the proof of Theorem 8 for more details of this.

[5] If the Responder removes all of the chips from the board using the random strategy then we conclude that the Responder has lost.

[6] This would occur if a chip had been lied about more than $k - j$ times.

Define $X = \sum X_c$ to be the sum over all the chips c. $\mathbb{E}[X]$ is defined to be the expected number of chips that remain on the board. Now

$$\mathbb{E}[X] = \mathbb{E}\Big[\sum X_c\Big] = \sum \mathbb{E}[X_c].$$

But $\sum \mathbb{E}[X_c]$ is defined to be the weight of the state, therefore, by assumption

$$\mathbb{E}[X] > 1.$$

The expectation value of X is an average value of X over all possible outcomes, so there must exist an outcome where X takes a value greater than 1. X is integer valued, hence there exists an outcome such that $X \geq 2$ i.e. there are at least 2 chips on the board at the end of the game. Since each outcome of X corresponds to a strategy, this implies that there exists a strategy which leaves at least 2 chips on the board regardless of the strategy that the Questioner chooses to follow. Since the Questioner and the Responder are playing a perfect information game[7] , either the Questioner or the Responder will win. Since there exists no strategy with which the Questioner will always win, there does exist a strategy with which the Responder will always win. □

We introduce the following notation, which will help us when evaluating the main result of [8].

$$\binom{j}{\leq s} = \sum_{t=0}^{s} \binom{j}{t}, \tag{3.3}$$

where $\binom{j}{\leq 0} = 1$ and if $s \geq j$ then $\binom{j}{\leq s} = 2^j$. This implies that (3.1) can be expressed as

$$Pr[B(j, 0.5) \leq s] = \binom{j}{\leq s} 2^{-j} \tag{3.4}$$

Let $j \geq 0$. We define the weight function

$$w_j(x_0, x_1, \ldots, x_k) = \sum_{i=0}^{k} x_i \binom{j}{\leq k-i} \tag{3.5}$$

In a game with j questions this is 2^j times the weight that was defined by (3.2). Note that the weight function is expressed as an integer.

By definition we have q to be the total number of questions in the game. We now define j to be the number of questions remaining at some point during the game.

3.2 The Main Theorem

If we consider the initial state of the game, we have the search space of size n and q questions remaining, then the weight function of this initial state is defined to be

$$w_q(x_0, x_1, \ldots, x_k) = \sum_{i=0}^{k} x_i \binom{q}{\leq k-i} \tag{3.6}$$

[7] See Appendix C.

By Theorem 8, if the state has a weight more than 1 the Responder wins. Since (3.6) is 2^q times greater than the weight defined before by (3.2), this implies that the Responder would win the game if the weight function of the initial state is greater than 2^q i.e.

$$\textit{If } w_q(x_0, x_1, \ldots, x_k) > 2^q, \textit{ then the Responder wins.}$$

It is possible to see by the above statement that if the weight function of the initial state is less than or equal to 2^q it might be possible for the Questioner to win. Hence the converse of Theorem 8 would be

$$\textit{If } w_q(x_0, \ldots, x_k) \leq 2^q, \textit{ then the Questioner wins.}$$

This condition, although necessary, is not sufficient. This is shown in Example 7, where we suppose that the converse holds.

Example 7 *Let $k = 1$, $n = 5$ and $q = 5$. This yields the initial state $(x_0 = 5, x_1 = 0)$. The weight of the initial state is*

$$\begin{aligned} w_5(5,0) &= 5 \times \binom{5}{\leq 1} + 0 \\ &= 5 \times 6 \\ &= 30 \\ &< 2^5. \end{aligned}$$

The Responder chooses a number between 1 and 5. The Questioner's best strategy is to split the size of the search space as evenly as possible. Suppose the Questioner asks the Responder "Is $x \leq 3$?" and receives the answer "Yes". The new state is $(3, 2)$, which has weight

$$\begin{aligned} w_4(3,2) &= 3 \times \binom{4}{\leq 1} + 2 \\ &= 15 + 2 \\ &= 17 \\ &> 2^4. \end{aligned}$$

Therefore, the Responder wins by Theorem 8.

So clearly the converse of Theorem 8 does not hold. We need to add the condition that the Questioner must survive the first k rounds of the game to ensure certainty of winning the game. Therefore, we consider a partial converse of Theorem 8, which is shown by Theorem 9. The focus for the rest of this chapter is on proving Theorem 9, which is the main result of Spencer [8], and evaluating its consequences.

Theorem 9 *There exist constants c, q_0, such that for all $q \geq q_0$, if*

$$w_q(x_0, x_1, \ldots, x_{k-1}, x_k) \leq 2^q$$

and

$$x_k > cq^k,$$

then the Questioner wins.[8]

Proof If the Questioner is able to win for some $(x_0, x_1, \ldots, x_{k-1}, x_k)$, then they win if x_k is reduced to any $x'_k < x_k$. Therefore, we only need to show that the Questioner wins under the assumption

$$w_q(x_0, x_1, \ldots, x_{k-1}, x_k) = 2^q \tag{3.7}$$

Define $P = (x_0, x_1, \ldots, x_{k-1}, x_k)$ and $\nu = (\nu_0, \nu_1, \ldots, \nu_{k-1}, \nu_k)$, where both P and ν are vectors. Define

$$\begin{aligned} Yes(P, \nu) &= (\nu_0, \nu_1 + x_0 - \nu_0, \ldots, \nu_k + x_{k-1} - \nu_{k-1}) \\ No(P, \nu) = Yes(P, P - \nu) &= (x_0 - \nu_0, x_1 - \nu_1 + \nu_0, \ldots, x_k - \nu_k + \nu_{k-1}) \end{aligned}$$

When P is the current state of the game, the Questioner selects a set of chips consisting of ν_i chips on the i^{th} position. This implies that $Yes(P, \nu)$ is the new position if the Responder replies "Yes". Similarly, if the Responder replies "No", then the new position is $No(P, \nu)$.

Let $j > 0$ and for all P and ν, we are able to calculate[9]

$$\begin{aligned} w_j(Yes(P, \nu)) + w_j(No(P, \nu)) &= w_j(Yes(P, \nu) + No(P, \nu)) \\ &= w_j(x_0, x_0 + x_1, \ldots, x_{k-1} + x_k) \\ &= x_0 \binom{j}{\leq k} + (x_0 + x_1)\binom{j}{\leq k-1} + \ldots \\ &\quad + (x_{k-2} + x_{k-1})\binom{j}{\leq 1} + (x_{k-1} + x_k)\binom{j}{\leq 0} \\ &= \sum_{i=0}^{k} x_i \left(\binom{j}{\leq k-i} + \binom{j}{\leq k-i-1} \right) \\ &= \sum_{i=0}^{k} x_i \binom{j+1}{\leq k-i}. \end{aligned}$$

Hence,

$$w_j(Yes(P, \nu)) + w_j(No(P, \nu)) = \sum_{i=0}^{k} x_i \binom{j+1}{\leq k-i} = w_{j+1}(P),$$

where the last equality follows by definition.

We also define

$$\Delta_j(P, \nu) = w_j(Yes(P, \nu)) - w_j(No(P, \nu)), \tag{3.8}$$

which will be evaluated later on in the proof.

At the start of play, by assumption (3.7), we have $w_q(P) = 2^q$. When playing the game if, with j moves remaining, we have $w_j(P) > 2^j$, then the Responder has won[10]. Suppose if we have $j + 1$ moves remaining and assume that $w_{j+1}(P) = 2^{j+1}$. The Questioner now selects ν and the Responder now has the choice of where the new position of the chip should be by choosing either $Yes(P, \nu)$ or $No(P, \nu)$. If $\Delta_j(P, \nu) \neq 0$, this implies that either

$$w_j(Yes(P, \nu)) > 2^j$$

[8] [8] page 310.

[9] Refer to Appendix D for proof of the final line.

[10] See Theorem 8.

or

$$w_j(No(P,\nu)) > 2^j.$$

Hence, the Responder selects the weight greater than 2^j and thus wins the game by Theorem 8. Therefore, the Questioner, with $j+1$ moves remaining, should select the ν with $\Delta_j(P,\nu) = 0$. If the Questioner is able to do this, then $w_j(P,\nu) = 2^j$, where P is the state of the game with j questions remaining.

So we refer back to (3.8) and expand this expression.

$$\begin{aligned}
\Delta_j(P,\nu) &= w_j(Yes(P,\nu)) - w_j(No(P,\nu)) \\
&= w_j(\nu_0 - (x_0 - \nu_0), \nu_1 + x_0 - \nu_0 - (x_1 - \nu_1 + \nu_0), \ldots, \nu_{k-1} + x_{k-2} - \nu_{k-2} \\
&\quad - (x_{k-1} - \nu_{k-1} + \nu_{k-2}), \nu_k + x_{k-1} - \nu_{k-1} - (x_k - \nu_k + \nu_{k-1})) \\
&= (\nu_0 - (x_0 - \nu_0))\binom{j}{\leq k} + (\nu_1 + x_0 - \nu_0 - (x_1 - \nu_1 + \nu_0))\binom{j}{\leq k-1} + \ldots \\
&\quad + (\nu_{k-1} + x_{k-2} - \nu_{k-2} - (x_{k-1} - \nu_{k-1} + \nu_{k-2}))\binom{j}{\leq 1} \\
&\quad + (\nu_k + x_k - 1 - \nu_{k-1} - (x_k - \nu_k + \nu_{k-1}))\binom{j}{\leq 0} \\
&= 2\nu_0\binom{j}{\leq k} - x_0\binom{j}{\leq k} + 2\nu_1\binom{j}{\leq k-1} + x_0\binom{j}{\leq k-1} - 2\nu_0\binom{j}{\leq k-1} \\
&\quad - x_1\binom{j}{\leq k-1} + \ldots + 2\nu_{k-1}\binom{j}{\leq 1} + x_{k-2}\binom{j}{\leq 1} - 2\nu_{k-2}\binom{j}{\leq 1} \\
&\quad - x_{k-1}\binom{j}{\leq 1} + 2\nu_k\binom{j}{\leq 0} + x_{k-1}\binom{j}{\leq 0} - 2\nu_{k-1}\binom{j}{\leq 0} - x_k\binom{j}{\leq 0} \\
&= 2\nu_0\binom{j}{k} - x_0\binom{j}{k} + 2\nu_1\binom{j}{k-1} - x_1\binom{j}{k-1} + \ldots + 2\nu_{k-1}\binom{j}{1} \\
&\quad - x_{k-1}\binom{j}{1} + 2\nu_k\binom{j}{0} - x_k\binom{j}{0} \\
&= \sum_{i=0}^{k}(2\nu_i - x_i)\binom{j}{k-i}.
\end{aligned}$$

$$\therefore \qquad \Delta_j(P,\nu) = \sum_{i=0}^{k}(2\nu_i - x_i)\binom{j}{k-i} \tag{3.9}$$

The Questioner should now think in terms of deciding, for each chip c, whether to place the chip in A_i or not. Suppose that the chip c is in position i. If the Questioner places c in A_i, then they add $\binom{j}{k-i}$ to Δ_j. If the Questioner leaves c out of A_i, then they subtract $\binom{j}{k-i}$ from Δ_j. The Questioner's objective is to make his decisions such that their effects balance out.

Definition 3.2.1 *The chips placed at position k are known as* ***pennies*** *(See Figure 3.4).*

Pennies have an important function, if the Questioner places a penny in or out of A_i, they will either add or subtract 1 from Δ_j.

We now introduce two types of play: Fictitious play and Perfect play. Assume that there are $j+1$ moves remaining in the game.

Figure 3.4: Pennies

Fictitious Play
The basic idea is to split the number of chips in play in half where possible. When you have an odd number of chips, throughout the game you should alternate between the choice of

$$\left\lceil \frac{x_i}{2} \right\rceil \qquad or \qquad \left\lfloor \frac{x_i}{2} \right\rfloor,$$

the first of which is known as the ceiling choice and the second is known as the floor choice. During fictitious play the Questioner alternates between their choice of floor and ceiling, for those i, where x_i is odd. The Questioner picks ν_k, such that $\Delta_j = 0$.

Example 8 *Let $k = 2$, $j = 6$ and consider the position $P = (1, 3, 75)$ which has*

$$\begin{aligned} w_7(1,3,75) &= 1 \times \binom{7}{\leq 2} + 3 \times \binom{7}{\leq 1} + 75 \times \binom{7}{\leq 0} \\ &= 1 \times (21 + 7 + 1) + 3 \times (7 + 1) + 75 \times (1) \\ &= 128 = 2^7. \end{aligned}$$

Suppose the Questioner selects $\nu_0 = 2$ and $\nu_1 = 3$. We now wish to find the value for ν_2, such that $\Delta_6 = 0$. Using (3.9) we have

$$\begin{aligned} \Delta_6((1,3,75),(2,3,\nu_2)) = (2+1)\binom{6}{2} + 3\binom{6}{1} + (2\nu_2 - 75) &= 0 \\ 0 = 45 + 18 + (2\nu_2 - 75) & \\ \therefore \qquad \nu_2 = 6. & \end{aligned}$$

In general the Questioner will have to solve an equation of the form $\Delta_j = 2\nu_k - A = 0$, where A will always be an even integer. This is true since, for all ν

$$w_j(Yes(P,\nu)) + w_j(No(P,\nu)) = 2^{j+1},$$

which is even. Hence the value of $w_j(Yes(P,\nu)) - w_j(No(P,\nu))$ is also even. Therefore, A must be even.

However, with fictitious play, a problem arises. Although we have the condition that A is even, it may not necessarily be positive i.e. it is possible that the number of pennies will be negative[11].

[11] However, the other co-ordinates remain positive.

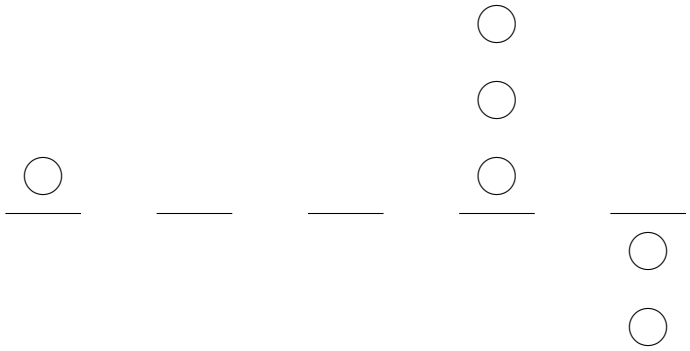


Figure 3.5: An example of what could occur during fictitious play

Example 9 *Let $k = 2$, $j = 6$ and consider the position $P = (1, 6, 51)$.*

$$\begin{aligned} w_7(1, 9, 51) &= 1 \times \binom{6}{\leq 2} + 6 \times \binom{6}{\leq 1} + 51 \times \binom{6}{\leq 0} \\ &= (1 \times 29) + (6 \times 8) + (51 \times 1) \\ &= 128. \end{aligned}$$

If $\nu_0 = 3$ and $\nu_1 = 7$ then,

$$\Delta_6((1, 6, 51), (3, 7, \nu_2)) = 5\binom{6}{2} + 8\binom{6}{1} + (2\nu_2 - 51) = 0$$

$$0 = 75 + 48 + 2\nu_2 - 51$$

$$\therefore \nu_2 = -36.$$

Let

$$fic(j) = (fic_0(j), fic_1(j), \ldots, fic_{k-1}(j), fic_k(j))$$

denote the state P when there are j rounds left in the game. $fic(q)$ is defined to be the initial state of the original game. The values of $fic(j)$ are dependent upon the Questioner's choice of floor or ceiling and the Responder's reply of Yes or No, hence $fic(j)$ can take many possible values[12]. It will be shown later that under the conditions of play we will not end up with negative numbers $fic_k(j)$ of pennies.

Perfect Play

In perfect play you exactly halve the search space, so it is possible to obtain half chips, quarter chips and so forth. When the state is P, the Questioner selects $\nu = \frac{P}{2}$. In perfect play it is clear that $Yes(P, \nu) = No(P, \nu)$ and we can define the state

$$pp(j) = (pp_0(j), pp_1(j), \ldots, pp_{k-1}(j), pp_k(j)),$$

when there are j rounds left in the game. $pp(q)$ is the initial state of the original game and is equal to $fic(q)$. We define $pp(j)$ such that

$$pp(j) = Yes\Big(pp(j+1), \frac{pp(j+1)}{2}\Big).$$

[12]When using inequalities involving $fic_i(j)$ they will hold regardless of the choices by the Questioner or the Responder.

Since we are playing under the conditions of perfect play, the number of chips that move to the right is exactly the same as the expected number if you had flipped an unbiased coin. Therefore,

$$pp_k(j) = \sum_{i=0}^{k} x_i Pr[B(q-j, 0.5) = k-i] \tag{3.10}$$

We will show that fictitious play is relatively close to that of perfect play.

For $0 \leq i \leq k$ and $0 \leq j \leq q$, define the error function

$$e_i(j) = |pp_i(j) - fic_i(j)| \tag{3.11}$$

Lemma 3.2.2 *There exists a constant c_2 such that, for all $j \geq 1$,*

$$e_k(j) \leq c_2 j^k.$$

Proof We know that

$$\begin{aligned} pp_k(j) &= \sum_{i=0}^{k} x_i Pr[B(q-j, 0.5) = k-i] \\ &= \sum_{i=0}^{k} x_i \binom{q-j}{k-i} 2^{-(q-j)}. \end{aligned}$$

Hence

$$pp_0(j) = x_0 2^{-(q-j)}$$

i.e in perfect play the zeroth position is $x_0 2^{-(q-j)}$. With fictitious play the zeroth position is either

$$\left\lfloor x_0 2^{-(q-j)} \right\rfloor \qquad or \qquad \left\lceil x_0 2^{-(q-j)} \right\rceil,$$

since at the end of each round of fictitious play we have to choose between the floor and ceiling integers. At the initial stage, we notice that $e_i(q) = 0$. Let $1 \leq i < k$. By the inductive definition of perfect play

$$pp_i(j) = \frac{1}{2}(pp_i(j+1) + pp_{i-1}(j+1)),$$

which holds since Yes(P,ν) = No(P, ν). This implies, whether the Responder lies or not, the size of the search space remains the same, by the definition of perfect play.

For $1 \leq i < k$,

$$|fic_i(j) - \frac{1}{2}(fic_i(j+1) + fic_{i-1}(j+1))| \leq 1 \tag{3.12}$$

This is holds since ν_i differs by at most $\frac{1}{2}$ from $\frac{x_i}{2}$ and similarly ν_{i-1} differs by at most $\frac{1}{2}$ from $\frac{x_{i-1}}{2}$.

By the definition of the error function

$$\begin{aligned} e_i(j) &= |pp_i(j) - fic_i(j)| \\ &\leq \left|\frac{1}{2}pp_i(j+1) + \frac{1}{2}pp_{i-1}(j+1) + 1 - \left|\frac{1}{2}fic_i(j+1) + \frac{1}{2}fic_{i+1}(j+1)\right|\right| \\ &\leq \frac{1}{2}e_i(j+1) + \frac{1}{2}e_{i-1}(j+1) + 1 \end{aligned}$$

$$\therefore \qquad e_i(j) \leq \frac{1}{2}e_i(j+1) + \frac{1}{2}e_{i-1}(j+1) + 1 \tag{3.13}$$

Define $M_i = 2^{i+1} - 1$, such that $M_0 = 1$ and $M_i \geq 1 + \frac{1}{2}M_i + \frac{1}{2}M_{i-1}$ for $i \geq 1$ [13] . Let $0 \leq i < k$ and $0 \leq j \leq q$, then

$$e_i(j) \leq M_i \tag{3.14}$$

Claim Inequality (3.14) holds.

Proof By using reverse induction we can prove that (3.14) holds. We start with $j = q$ i.e. at the beginning of the game. We wish to show $e_i(j) \leq M_i$. We know

$$\begin{aligned} & e_i(q) = 0 \\ \Rightarrow \qquad & e_i(q) \leq M_i, \end{aligned}$$

since $M_i \geq 1$ by definition.

Now consider the case when $j < q$ and assume that (3.14) holds for $j + 1$.

$$\Rightarrow \qquad e_i(j+1) \leq M_i \qquad for\ all\ \ 0 \leq i < k \tag{3.15}$$

By using (3.15) we wish to show that $e_i(j) \leq M_i$. From (3.13)

$$\begin{aligned} e_i(j) &\leq 1 + \frac{1}{2}e_i(j+1) + \frac{1}{2}e_{i-1}(j+1) \\ &\leq 1 + \frac{1}{2}M_i + \frac{1}{2}M_{i-1} \\ &\leq M_i \qquad\qquad\qquad\qquad if \quad i \geq 1. \end{aligned}$$

If $i = 0$, the inequality still holds since

$$e_0(j) \leq 1 = M_0. \qquad \square$$

Recall Definition 3.2.1, which defines pennies to be the chips on the k^{th} position on the board. During fictitious play we have seen that it is possible for a state to contain a negative number of pennies [14]. When playing fictitiously, once you have chosen $\nu_0, \ldots, \nu_{k-1}$, it is possible to determine the value of ν_k by using (3.16)[15].

$$\Delta_j(P, \nu) = \sum_{i=0}^{k}(2\nu_i - x_i)\binom{j}{k-i} = (2\nu_k - x_k) - \underbrace{\sum_{i=0}^{k-1}(x_i - 2\nu_i)\binom{j}{k-i}}_{(\dagger)} \tag{3.16}$$

[13]

$$1 + \frac{1}{2}M_i + \frac{1}{2}M_{i-1} = 1 + 2^i - \frac{1}{2} + 2^{i-1} - \frac{1}{2} = 2^i + 2^{i-1}.$$

However, for all $i \geq 1$

$$M_i = 2^{i+1} - 1 = 2^i + 2^i - 1 \geq 2^i + 2^{i-1}$$

as required.

[14] Refer to Example 9.

[15] (3.16) is an expanded form of (3.9).

In fictitious play we are able to find an upper bound for the summation (†) shown in (3.16). When x_i is odd, our alternating choices of floor and ceiling imply

$$x_i - 2\nu_i = \pm 1$$

i.e. the difference in size between $Yes(P,\nu)$ and $No(P,\nu)$ positions is one chip. When x_i is even, this implies

$$x_i - 2\nu_i = 0$$

i.e. the number of chips in $Yes(P,\nu)$ and $No(P,\nu)$ positions are identical. As a result of the above two equations, the summation (†) is less than or equal to $\sum_{i=0}^{k-1} \binom{j}{k-i}$, implying that

$$\begin{aligned} |2\nu_k - x_k| &\leq \sum_{i=0}^{k-1} \binom{j}{k-i} \\ &\leq \sum_{i=0}^{k-1} j^{k-i} \\ &\leq j^k. \end{aligned}$$

Thus

$$\left|\nu_k - \frac{x_k}{2}\right| < j^k.$$

So it is possible to bound (3.12), when $k = i$, such that

$$|fic_k(j) - \frac{1}{2}\big(fic_k(j+1) + fic_{k-1}(j+1)\big)| \leq j^k + 1.$$

It is possible to do this, since when we have evaluated (3.16) it was evaluated it under the conditions of fictitious play i.e. the alternating choices of floor and ceiling have allowed us to determine of the number of pennies at a point in the game with j questions remaining. Thus it follows

$$e_k(j) \leq j^k + 1 + \frac{1}{2}e_{k-1}(j+1) + \frac{1}{2}e_k(j+1).$$

If $i = k-1$ (which is possible since $i = k-1 < k$), then $e_{k-1}(j+1)$ is bounded from above by (3.14). It is therefore possible to absorb $e_{k-1}(j+1)$ into a constant, say c_1, dependant only upon the value of k, such that

$$e_k(j) \leq j^k + c_1 + \frac{1}{2}e_k(j+1) \leq c_1 j^k + \frac{1}{2}e_k(j+1) \tag{3.17}$$

Further, it is possible to write $\frac{1}{2}e_k(j+1)$ in terms of a summation using (3.17) repeatedly.

$$\begin{aligned} \frac{1}{2}e_k(j+1) &\leq \frac{1}{2}(c_1(j+1)^k + \frac{1}{2}e_k(j+2)) \\ &\leq \frac{1}{2}c_1(j+1)^k + \frac{1}{4}c_1(j+2)^k + \frac{1}{4}e_k(j+3) \\ &\leq \dots \end{aligned}$$

i.e. we can carry on this expansion until we reach the point when $e_k(q) = 0$. Hence we can write the right hand side of (3.17) in terms of the following summation

$$\sum_{m=j}^{q} c_1 m^k 2^{j-m} = c_1 j^k + \frac{1}{2} c_1 (j+1)^k + \frac{1}{4} c_1 (j+2)^k + \ldots,$$

which implies

$$e_k(j) \leq \sum_{m=j}^{q} c_1 m^k 2^{j-m}.$$

By setting $y = m - j$,

$$\begin{aligned}
\sum_{m=j}^{q} c_1 m^k 2^{j-m} &= c_1 \sum_{y=0}^{q} (y+j)^k 2^{-y} \\
&= c_1 j^k \sum_{y=0}^{q} \left(\frac{y+j}{j}\right)^k 2^{-y} \\
&\leq c_1 j^k \sum_{y=0}^{\infty} \left(\frac{y+j}{j}\right)^k 2^{-y}
\end{aligned}$$

This implies

$$e_k(j) \leq c_1 j^k \sum_{y=0}^{\infty} \left(\frac{y+j}{j}\right)^k 2^{-y}.$$

The errors that occurred in the initial rounds of the game effectively disappear due to the halving process of the game. We observe that $\sum_{y=0}^{\infty} \left(\frac{y+j}{j}\right)^k 2^{-y}$ is convergent for all j. We are therefore able to absorb the summation into a constant c_2, such that

$$e_k(j) \leq c_2 j^k.$$

□

The Questioner's Strategy
The Questioner plays fictitious play until there exists at most one nonpenny on the board (see Figure 3.6).

Figure 3.6: An example of where the Questioner wishes to be by the time they reach the Endgame Steps Stage

It is at this point the Questioner begins using a strategy known as the **Endgame Strategy**, which the Questioner uses until the end of the game. The analysis of this

Endgame Strategy requires us to show that fictitious play doesn't leave the Questioner with a negative number of pennies. To work through the strategy, we split the analysis into several stages:

1. FIRST STEPS $\quad 0 \leq q - j < k$ for the first k rounds;
2. MIDDLE STEPS $\quad k \leq q - j$ and $j > (\ln q)^2$;
3. LATE MIDDLE STEPS $\quad \sqrt{\ln q} < j \leq (\ln q)^2$;
4. EARLY END STEPS $\quad \frac{1}{2}\sqrt{\ln q} \leq j \leq \sqrt{\ln q}$;
5. ENDGAME STEPS $\quad 0 \leq j \leq \frac{1}{2}\sqrt{\ln q}$.

Note that we are using logarithms to base 2.

To show that the Questioner is able to play fictitious play, we must show that for all j, $fic_k(j) \geq 0$. This can be done by showing the inequality

$$e_k(j) \leq pp_k(j) \tag{3.18}$$

We will prove this separately for the different stages of the Questioner's strategy.

First steps stage
We know by Lemma 3.2.2, $\quad e_k(j) \leq c_2 j^k$. Since $j \leq q$

$$\Rightarrow \qquad e_k(j) \leq c_2 j^k \leq c_2 q^k$$

In this stage it is our aim to show

$$pp_k(j) \geq pp_k(q)2^{-(q-j)} > x_k 2^{-k} > c2^{-k}q^k, \tag{3.19}$$

where c is the constant referred to in the theorem and is sufficiently large. However, by (3.10)

$$pp_k(j) = \sum_{i=0}^{k} x_i \binom{q-j}{k-i} 2^{-(q-j)} \geq \sum_{i=0}^{k} x_i 2^{-(q-j)} = pp_k(q)2^{-(q-j)},$$

since $pp_k(q) = \sum_{i=0}^{k} x_i$. Also

$$pp_k(q)2^{-(q-j)} \geq x_k 2^{-(q-j)} > x_k 2^{-k},$$

since the First Steps Stage condition specifies $0 \leq q - j < k \quad i.e. \quad 0 \geq -(q-j) > -k$

$$\Rightarrow \qquad pp_k(j)2^{-(q-j)} > x_k 2^{-k}.$$

Using the initial assumption $x_k > cq^k$, stated in the theorem, we clearly have

$$x_k 2^{-k} > cq^k 2^{-k}.$$

Hence we achieve the inequality (3.19). Thus if we select c such that $c2^{-k} \geq c_2$, we can assure that the Questioner will survive the first k rounds of the game.

Middle steps Stage

In this stage the aim is to show

$$e_k(j) \leq c_2 j^k < \frac{2^j}{q^k} \leq pp_k(j) \tag{3.20}$$

We know

$$Pr[B(q-j,0.5) = k-i] = \binom{q-j}{k-i} 2^{-(q-j)} \geq 2^{-(q-j)}.$$

Combining this with (3.10) we have

$$pp_k(j) = \sum_{i=0}^{k} x_i Pr[B(q-j,0.5) = k-i] \geq \sum_{i=0}^{k} x_i 2^{-(q-j)},$$

where $\sum_{i=0}^{k} x_i$ equals the maximum numbers chips at beginning of game. Recall Definition 3.5, which defines the weight of a state. It tells us

$$\begin{aligned} w_j(x_0, x_1, \ldots, x_k) &= \sum_{i=0}^{k} x_i \binom{q}{\leq k-i} \\ &= x_0 \binom{q}{\leq k} + x_1 \binom{q}{\leq k-1} + \ldots + x_k \binom{q}{\leq 0}, \end{aligned}$$

from which it is easily seen that the weight of any chip is at most $\binom{q}{\leq k}$. Further,

$$\binom{q}{\leq k} = \sum_{t=0}^{k} \binom{q}{t} \leq \frac{q^k}{k!} \leq q^k \tag{3.21}$$

We know by assumption $w_j(x_0, x_1, \ldots, x_k) = 2^q$, which represents the total weight of the chips. Hence (3.21) implies that the total number of chips is at least $\frac{2^q}{q^k}$ i.e.

$$\sum_{i=0}^{k} x_i \geq \frac{2^q}{q^k}$$

$$\therefore \qquad pp_k(j) \geq 2^{-(q-j)} \sum_{i=0}^{k} x_i \geq \frac{2^j}{q^k}.$$

We now wish to show that

$$c_2 j^k < \frac{2^j}{q^k}, \tag{3.22}$$

which is equivalent to showing

$$c_2 (jq)^k < 2^j.$$

We note that

$$\begin{aligned} c_2 (qj)^k &\leq c_2 q^{2k} && (3.23)\\ &\leq q^{2k+1} && (3.24)\\ &\leq q^{\ln q} = 2^{(\ln q)^2} && (3.25)\\ &< 2^j && (3.26) \end{aligned}$$

(3.23) is by fact $j \leq q$ and (3.24) uses $c_2 << q$. (3.25) holds since k is a small fixed integer, which implies $2k+1 \leq \ln q$. By condition of the Middle Steps Stage $(\ln q)^2 < j$ and we get (3.26). Hence (3.22) follows. We know by Lemma 3.2.2 that $e_k(j) \leq c_2 j^k$, so (3.20) follows.

Late Middle Steps Stage
Recall the formula for perfect play derived by (3.10)

$$pp_k(j) = \sum_{i=0}^{k} x_i Pr[B(q-j, 0.5) = k-i].$$

We now give a lower bound on this using the two inequalities (3.27) and (3.28). If q is sufficiently large we claim that

$$Pr[B(q-j,0.5) = k-i] > \frac{1}{2} Pr[B(q-j,0.5) \leq k-i] \tag{3.27}$$

$$Pr[B(q-j,0.5) \leq k-i] \geq \frac{1}{2} Pr[B(q,0.5) \leq k-i] 2^j \tag{3.28}$$

Firstly we show (3.27). For this, note that for all n, l,

$$Pr[B(n,0.5) = l] = \binom{n}{l} 2^{-n}$$

$$Pr[B(n,0.5) \leq l] = \sum_{s=0}^{l} \binom{n}{s} 2^{-n}.$$

Further, for all $s < n$,

$$\frac{\binom{n}{s}}{\binom{n}{s+1}} = \frac{\frac{n!}{s!\,(n-s)!}}{\frac{n!}{(s+1)!\,(n-s-1)!}} = \frac{s+1}{(n-s+1)} \leq \frac{1}{2}.$$

Thus for all $s \leq l$,

$$\binom{n}{s} \leq \frac{1}{2}\binom{n}{s+1} \leq \frac{1}{4}\binom{n}{s+2} \leq \dots$$

$$\Rightarrow \qquad \binom{n}{s} \leq 2^{-(l-s)} \binom{n}{l}.$$

Hence

$$\begin{aligned} \sum_{s=0}^{l} \binom{n}{s} &\leq 2^{-n} \binom{n}{l} \sum_{s=0}^{l} 2^{-(l-s)} \\ &< 2^{-n} \binom{n}{l} \sum_{i=0}^{\infty} 2^{-i} = 2 \times 2^{-n} \binom{n}{l} \\ &= 2 Pr[B(n,0.5) = l] \end{aligned}$$

$$\Rightarrow \qquad Pr[B(n,0.5) = l] > \frac{1}{2} Pr[B(n,0.5) \leq l],$$

which completes the proof of (3.27), by substituting $q-j$ for n and $k-i$ for l. Now we need to show (3.28). This equivalent to

$$2^{-(q-j)}\sum_{i=0}^{l}\binom{q-j}{i} \geq \frac{1}{2}2^j 2^{-q}\sum_{i=0}^{l}\binom{q}{i}.$$

We see that [16]

$$\begin{aligned}
\frac{\binom{q-j}{i}}{\binom{q}{i}} &= \frac{(q-j)!}{i!\,(q-i-j)!} \times \frac{i!\,(q-i)!}{q!} \\
&= \frac{q-i}{q} \times \frac{q-i-1}{q-1} \times \ldots \times \frac{q-i-j}{q-j} \\
&\geq \Big(\frac{q-i-j}{q-j}\Big)^j = \Big(1-\frac{i}{q-j}\Big)^j \\
&\geq \Big(1-\frac{i}{q}\Big)^j \\
&\geq 1-\frac{ij}{q} \\
&\geq \frac{1}{2} \\
\Rightarrow \qquad 2^{-(q-j)}\sum_{i=0}^{l}\binom{q-j}{i} &\geq \frac{1}{2}2^j 2^{-q}\sum_{i=0}^{l}\binom{q}{i},
\end{aligned}$$

which completes the proof of (3.28). Thus by (3.27) and (3.28) we obtain

$$Pr[B(q-j,0.5)=k-i] > \frac{1}{4}2^j Pr[B(q,0.5)\leq k-i]. \tag{3.29}$$

Using (3.29) we can show

$$pp_k(j) \overset{(\maltese)}{>} \frac{1}{4}2^j > c_2 j^k \geq e_k(j). \tag{3.30}$$

For this first note that by (3.5), (3.10) and the assumption that the initial weight is equal to 2^q, we see that

$$\begin{aligned}
\sum_{i=0}^{k} x_i Pr[B(q,0.5)\leq k-i] &= \sum_{i=0}^{k} x_i \binom{q}{\leq k-i} 2^{-q} \\
&= 2^{-q} w_q(x_0,\ldots,x_k) = 1 \\
\Rightarrow \qquad \sum_{i=0}^{k} x_i Pr[B(q,0.5)\leq k-i] &= 1
\end{aligned} \tag{3.31}$$

[16] See Appendix E to see that

$$\Big(1-\frac{i}{q}\Big)^j \geq 1-\frac{ij}{q}.$$

Indeed, to see the first inequality (✠), note that by (3.29), (3.10) and then (3.31) we obtain

$$\begin{aligned} pp_k(j) &= \sum_{i=0}^{k} x_i Pr[B(q-j,0.5) = k-i] \\ &> \frac{1}{4}2^j \sum_{i=0}^{k} x_i Pr[B(q,0.5) \leq k-i] \\ &\geq \frac{1}{4}2^j. \end{aligned}$$

By the Middle Steps Stage assumption on q and j it follows that $\frac{1}{4}2^j > c_2 j^k$ and similarly from Lemma 3.2.2 we know $c_2 j^k \geq e_k(j)$. Hence (3.30) follows.

Early End Steps Stage

By the end of this stage we aim to have at most one nonpenny remaining on the board. We begin by showing that there are a bounded number of chips in each position $s < k$. This implies that there exists a bounded number of nonpennies.

We know by Lemma 3.2.2 that $e_k(j) \leq c_2 j^k$, and $e_s(j)$ is bounded too for $s < k$ by (3.14). Further, we need to show that $pp_s(j)$ is also bounded from above. We know from (3.29) that

$$pp_s(j) = \sum_{i=0}^{s} x_i Pr[B(q-j,0.5) = s-i] \tag{3.32}$$

and by (3.31)

$$1 = \sum_{i=0}^{k} Pr[B(q,0.5) \leq k-i] \tag{3.33}$$

We claim

$$Pr[B(q-j,0.5) = s-i] \leq \frac{c_3}{q} Pr[B(q-j,0.5) \leq k-i], \tag{3.34}$$

where c_3 is a constant dependant only on k.

This is equivalent to showing for arbitrary n, l, that

$$Pr[B(n,0.5) = l] \leq \frac{c_4}{n} Pr[B(n,0.5) \leq l+1] \tag{3.35}$$

To prove (3.35) we observe the following two equations

$$\begin{aligned} Pr[B(n,0.5) = l] &= 2^{-n}\binom{n}{l} \\ Pr[B(n,0.5) \leq l+1] \geq Pr[B(n,0.5) = l+1] &= \binom{n}{l+1}2^{-n} \end{aligned}$$

But

$$\begin{aligned} \binom{n}{l+1} &= \frac{n \times (n-1) \times \ldots \times (l+1)}{(l+1)!} \\ &= \binom{n}{l}\frac{n-l}{l+1} \geq \binom{n}{l}\frac{n}{2l} \end{aligned}$$

Let $c_4 = 2l$, then we obtain

$$\frac{c_4}{n}\binom{n}{l+1} \geq \binom{n}{l}.$$

It follows

$$Pr[B(n, 0.5) = l] \leq \frac{c_4}{n} Pr[B(n, 0.5) \leq l + 1].$$

Hence we obtain (3.34).

Further, it is possible to bound

$$Pr[B(q - j, 0.5) \leq k - i] \leq 2^j Pr[B(q, 0.5) \leq k - i], \tag{3.36}$$

since if $q - j$ flips of a coin give $k - i$ heads, with the probability 2^{-j}, the next j flips of the coin will all be tails. More formally, note that

$$\begin{aligned} Pr[B(q - j, 0.5) \leq k - i] &= \sum_{i=0}^{k} \binom{q-j}{k-i} 2^{-(q-j)} \\ 2^j Pr[B(q, 0.5) \leq k - i] &= \sum_{i=0}^{k} \binom{q}{k-i} 2^{-q} \times 2^j \\ &= \sum_{i=0}^{k} \binom{q}{k-i} 2^{-(q-j)}. \end{aligned}$$

But clearly

$$\binom{q}{k-i} 2^{-(q-j)} \geq \binom{q-j}{k-i} 2^{-(q-j)},$$

which implies (3.36). From (3.36), we see

$$\begin{aligned} pp_s(j) &\overset{(3.32)}{=} \sum_{i=0}^{s} x_i Pr[B(q - j, 0.5) = s - i] \\ &\overset{(3.36)}{\leq} \frac{c_3}{q} 2^j \sum_{i=0}^{k} x_i Pr[B(q, 0.5) \leq k - i] \\ &\overset{(3.33)}{<} \frac{c_3 2^j}{q}, \end{aligned}$$

which is less than one in the Early End Steps Stage.

We have shown that one can apply fictitious play. So in the original game, the position i is the same as in fictitious play. Thus altogether, using Lemma 3.2.2, this shows that with j questions remaining[17]

[17]Note that $\sum_{i=0}^{k-1} x_i$ is equivalent to the number of nonpennies. Further recall by definition

$$e_i(j) = |pp_i(j) - fic_i(j)|.$$

In addition, (3.14) bounds $e_i(j)$, such that $e_i(j) \leq M_i$.

$$\begin{aligned}\sum_{i=0}^{k-1} x_i &\leq \sum_{s<k} fic_s(j) \\ &\leq \sum_{s<k} pp_s(j) + |fic_s(j) - pp_s(j)| \\ &\leq \sum_{s<k} 1 + e_s(j) \\ &\leq M_i.\end{aligned}$$

i.e. the number of nonpennies is altogether bounded i.e. it only depends upon the value of k.

Define the nonpenniness of a state $(x_0, x_1, \ldots, x_{k-1}, x_k)$ as $\sum_{i=0}^{k-1} x_i(k-i-1)$. This represents the number of moves to the right required to make all of the nonpennies into pennies i.e. nonpennies are moved into the k^{th} position. Let M_i be the bound on the nonpenniness at the end of the Early End Steps Stage, where by above we can take M_i to be a constant solely dependent upon k. At each round, assuming that there are at least 2 nonpennies left on the board, the nonpenniness must decrease by at least one. This is due to the alternating choices between floor and ceiling of the Questioner in fictitious play, which assured that if there exists more than one nonpenny they could not all be in the set A_i nor could they all not be in A_i [18]. Within M rounds, the Questioner reaches the stage where there is a most one nonpenny[19].

Endgame Steps Stage

For the following lemma there does not exist any asymptotics i.e. j, and k can be considered as an arbitrary number. The lemma is known as the Endgame Lemma. The lemma deals with the case when there is at most one nonpenny left on the board, which occurs due to the alternating floor and ceiling choices that are made during fictitious play.

Lemma 3.2.3 *Let $(x_0, \ldots, x_k)$ be a position with $x_0 \leq 1$, $x_1 = \ldots = x_{k-1} = 0$ and $w_{j+1}(x_0, \ldots, x_k) = 2^{j+1}$. Then the Questioner wins in the $j+1$-move game.*

Proof By induction on j we just need to find a move ν for the Questioner s.t. $\Delta_j(P, \nu) = 0$, since $Yes(P, \nu)$ and $No(P, \nu)$ remain in this form i.e

$$0 = \Delta_j(P, \nu) = w_j(Yes(P, \nu)) - w_j(No(P, \nu)) = \sum_{i=0}^{k} (2\nu_i - x_i) \binom{j}{k-i}.$$

If $x_0 = 0$ then the condition $w_{j+1}(x_0, \ldots, x_k) = 2^{j+1}$ implies $x_k = 2^{j+1}$, which is simply the liar game without the lies, so the Questioner takes $\nu = (0, \ldots, 0, 2^j)$.

Otherwise, $x_0 = 1$. We know that

$$w_{j+1}(x_0, \ldots, x_k) = \sum_{i=0}^{k} x_i \binom{j+1}{\leq k-i}.$$

[18] We note that this is the only point in which we use the condition of the alternating choices between floor and ceiling.

[19] This certainly occurs by the end of the Early End Steps Stage.

By assumption of the lemma [20]

$$\begin{aligned} 2^{j+1} &= w_{j+1}(x_0, 0, \ldots, 0, x_k) \\ &= x_0 \binom{j+1}{\leq k} + x_k \binom{j+1}{\leq 0} \\ &= \binom{j+1}{\leq k} + x_k \\ &= \binom{j}{\leq k} + \binom{j}{\leq k-1} + x_k. \end{aligned}$$

If $j+1 \leq k$,

$$\begin{aligned} \Rightarrow \qquad 2^{j+1} &= \binom{j+1}{\leq k} + x_k \\ &= 2^{j+1} + x_k \\ \therefore \qquad x_k &= 0. \end{aligned}$$

This holds since we know, by (3.3), if $k \geq j$ then $\binom{j}{\leq k} = 2^j$. This result implies that the Questioner has already won. Note that the number of subsets of the j-element set is 2^j and $\binom{j}{\leq k}$ counts the subsets of size almost k. So

$$\binom{j}{\leq k}, \binom{j}{\leq k-1} \leq 2^j \tag{3.37}$$

Since (3.37) holds, there exists integers x and y with the condition $0 \leq y \leq x$, such that

$$\binom{j}{\leq k} + y = \binom{j}{\leq k-1} + x - y = 2^j.$$

The Questioner then plays $\nu = (1, 0, \ldots, 0, y)$ to win. □

This completes the proof of Theorem 9 since we have shown that at the start of the Endgame Steps Stage there exists only 1 nonpenny, so it is possible to apply Lemma 3.2.3 and hence the Questioner can win. □

Example 10 *Let $k = 6$ and $j = 9$. Consider the state (1,0,0,0,0,176). Then*

$$\begin{aligned} w_{10}(1, 0, 0, 0, 0, 176) &= \binom{10}{\leq 6} + 176 \\ &= (1 + 10 + 45 + 120 + 210 + 252 + 210) + 176 \\ &= 1024 \\ &= 2^{10}. \end{aligned}$$

Therefore the Questioner solves

$$\begin{aligned} 2^9 &= \binom{9}{\leq 6} + y \\ &= (1 + 9 + 36 + 84 + 126 + 126 + 84) + y \\ &= 466 + y \\ \therefore \qquad y &= 46. \end{aligned}$$

[20] Refer to Appendix D for proof of the final line.

The Questioner then selects $\nu = (1, 0, 0, 0, 0, 46)$. *If the Responder says "Yes", then the new position is* $\nu = (1, 0, 0, 0, 0, 46)$ *and if the Responder says "No", the new position is* $(0, 1, 0, 0, 0, 130)$. *For each new position, whether "Yes" or "No", we have* $w_9 = 2^9$.

3.3 Consequences of Theorem 9

Suppose that we have the original $[n, q, k]$-game with initial position $P = (n, 0, \ldots, 0)$, where q is sufficiently large. By using the result of Spencer's main theorem we wish to show that it is possible for the Questioner to win the $[n, q, k]$-game. By showing that the Questioner is able to win a much harder version of the liar game with position $P' = (n, 0, \ldots, 0, nq^k)$ using Spencer's theorem, it follows that the Questioner is able to win the original $[n, q, k]$-game.

Theorem 10 *For all* k, *there exists a constant* c_5, *such that if*

$$n \leq \frac{2^q}{\binom{q}{\leq k}} - c_5, \tag{3.38}$$

then the Questioner can win the game.

Proof Recall that a penny is defined to be a chip at the k^{th} position. The weight of a penny is clearly 1, since a penny can only be lied about once more, where it is then removed from the board. Thus by adding a penny to the game increases the weight by 1. If you were to add cq^k pennies to the game, the weight therefore increases by cq^k.

We claim that if c_5 is sufficiently large, adding cq^k pennies only increases the weight slightly. Hence we obtain the new position

$$P' = (n, 0, \ldots, 0, cq^k),$$

which still has weight $w_q(P') \leq 2^q$ and so we can apply Theorem 9.

We know the initial weight of P is

$$w_q(P) = n\binom{q}{\leq k} \tag{3.39}$$

Substituting (3.38) into (3.39) gives

$$\begin{aligned} n\binom{q}{\leq k} &\leq \Big(\frac{2^q}{\binom{q}{\leq k}} - c_5\Big)\binom{q}{\leq k} \\ &= 2^q - c_5\binom{q}{\leq k}. \end{aligned}$$

So for the claim to hold we need

$$cq^k \leq c_5\binom{q}{\leq k}, \tag{3.40}$$

since this implies that the total weight is less than or equal to 2^q after the addition of cq^k pennies to the game, and hence we will be able to apply Theorem 9. To prove (3.40),

remembering $k \ll q$, we know that

$$\binom{q}{\leq k} \geq \binom{q}{k} = \frac{q \times (q-1) \times \ldots \times (q-k+1)}{k!}$$
$$\geq \frac{(q-k)^k}{k^k} \geq \frac{\left(\frac{q}{2}\right)^k}{k^k} = \left(\frac{q}{2k}\right)^k \tag{3.41}$$

If we take $(2k)^k$ as part of a constant[21] we obtain (3.40). Hence for the position P', we can apply the main theorem and the Questioner wins the game. Adding cq^k pennies to the game only makes it harder for the Questioner to win the game. This implies that the Questioner wins the simpler game with k lies and q questions, with the initial position $P = (n, 0, \ldots, 0)$. □

3.4 The bound on the number of questions q

By using previous results about q, it is possible to give a good bound on q if n is large in comparison to the value of k.

Theorem 11 *For all k, there exists constants c' and c'', such that the following holds: Consider a liar game with search space of size at most n and with at most k lies. Let q be the smallest number of questions such that there exists a winning strategy for the Questioner. Then*

$$\log n + k \log(\log n) - c'' \leq q \leq \log n + k \log(\log n) + c' \tag{3.42}$$

Proof It is easy to see that

$$q \leq (2k+1) \log n \tag{3.43}$$

(Note that the value $\log n$ determines how many questions the Questioner needs to ask to win the liar game with $k = 0$) [22]. In the "worst" way of solving the problem, suppose the Questioner asks every question $2k$ times. If the answers are consistent then the Questioner asks the next question. However, if the answers are inconsistent the Questioner must ask the question once more and take the majority answer. Hence the Questioner must ask at most $(2k+1) \log n$ questions, which implies (3.43).

Further,

$$\binom{q}{\leq k} = \sum_{t=0}^{k} \binom{q}{t} \leq q^k \tag{3.44}$$

By (3.38) we obtain

$$n \geq \frac{2^q}{\binom{q}{\leq k}} - c \tag{3.45}$$

Rearranging (3.45) and substituting in (3.44), we obtain

$$n + c \geq \frac{2^q}{q^k k}$$
$$\Rightarrow \qquad \log(n+c) \geq q - k \log q - \log k,$$

[21] Further, we see that c_5 is dependent purely upon the value of k

[22] Remember you need to take the next greater integer if $\log n$ is not an integer.

which implies

$$\begin{aligned} q &\leq \log(n+c) + k\log q + \log k \\ &\overset{(3.43)}{\leq} \log n + \log c + k\log((2k+1)\log n) + \log k \\ &= \log n + k\log(2k+1) + \log k + k\log(\log n) + \log c. \end{aligned}$$

This gives

$$q \leq \log n + k\log(\log n) + c', \qquad (c' > 0) \tag{3.46}$$

where c' is a constant that is solely dependent on the value of k.

It is also easy to see that

$$q \geq \log n, \tag{3.47}$$

since $\log n$ represents the minimum number of questions required for the Questioner to win the liar game with no lies.

Now

$$\begin{aligned} \binom{q}{\leq k} &= \frac{q \times \ldots (q-k+1)}{k!} \\ &\overset{(3.41)}{\geq} \left(\frac{q}{2k}\right)^k \end{aligned} \tag{3.48}$$

We know

$$n \leq \frac{2^q}{\binom{q}{\leq k}}.$$

Substituting (3.48) into this gives

$$\begin{aligned} n &\leq \frac{2^q}{\left(\frac{q}{2k}\right)^k} \\ \Rightarrow \qquad \log n &\leq q - k\log q + k\log 2k \\ &\overset{(3.47)}{\leq} q + k\log 2k - k\log(\log n). \end{aligned}$$

This gives

$$q \geq \log n + k\log(\log n) - c'', \qquad (c'' > 0) \tag{3.49}$$

where c'' is a constant solely dependent on the value of k. By combining (3.49) and (3.46), we obtain (3.42). □

Chapter 4

Strategies for the Liar game with 1 lie

This chapter discusses a strategy with which one can solve the "Twenty Questions" game and provides three strategies which solve the liar game with 1 lie and $n = 10^6$. Using Pelc [4] we are able to show that 25 questions is sufficient to win the liar game with 1 lie and $n = 10^6$ and provide a strategy in which one can win the game using at most 25 questions. In addition, by considering the liar game in terms of chips, as in Spencer [8], and by using results of Chapter 3, we shall find an optimal strategy for the Questioner in the liar game with one lie and $n = 10^6$ and the case where $n = 2^l$, where l is a natural number.

4.1 The strategy for the "Twenty Questions" game

When considering the "Twenty Questions" game i.e. the liar game with $k = 0$ and $n = 10^6$, there are two possible strategies we could adopt to determine the integer x. The first strategy involves splitting the search space asymmetrically, whereas the second strategy adopts the approach of splitting the search space symmetrically. We wish to use the strategy that reduces the number of questions in which it takes to find the integer x.

Consider splitting the search space asymmetrically, for example, split 10^6 such that you obtain two sets $A_1 = \{1, \ldots, 250,000\}$ and $A_2 = \{250,001, \ldots, 10^6\}$, and ask a question with regard to these smaller sets, for example, "Is $x \in A_1$?". Clearly if the Responder replies "Yes", then the Questioner has to ask less questions to determine x in comparison to the Responder replying "No".

Now consider splitting the search space symmetrically i.e. split 10^6 such that you obtain two sets $A_1 = \{1, \ldots, 500,000\}$ and $A_2 = \{500,001, \ldots, 10^6\}$, and ask a question with regard to these sets, for example, "Is $x \in A_1$?". Regardless of the Responder's reply, the Questioner has to determine x from sets of equal size, so the number of questions required to determine x from either set is equal.

If we compare the two strategies after the first question is asked, it is evident that if, in the asymmetric strategy, the Responder states x is in the smaller set, then it will take less questions to identify x than it will to identify x using the symmetric strategy. However, if the Responder states that x lies in the larger set, then it will require more questions to identify x than in comparison with the symmetric strategy. Since we are

playing to always minimise the number of questions the Questioner is required to ask, we adopt the symmetric strategy. So how many questions will it take to find x in a search space of size 10^6? Remembering that we are asking questions of the form "Is $x \in A_i$?", we obtain either "Yes" or "No" answers, hence

$$\begin{aligned} & 2^q = 1000000 \\ \Leftrightarrow \qquad & q = 19.93156857. \end{aligned}$$

Thus by following the strategy of splitting the search space symmetrically, we will require at most 20 questions to identify x. The strategy to solve the "Twenty Questions" game is described as follows:

Take the search space $n = \{1, 2, 3, \ldots, 10^6\}$, and split it into two sets of equal size A_1 and A_2, where $A_1 = \{1, \ldots, 500,000\}$ and $A_2 = \{500,001, \ldots, 1,000,000\}$. The first question the Questioner asks is then "Is $x \in A_1$?". If the Responder replies "Yes", then the Questioner takes A_1 and splits this set into two sets of equal size and asks another question of the form "Is $x \in A_i$?", where A_i represents one of the smaller sets contained within A_1. If the Responder, however, replies "No" then the Questioner takes A_2 and splits this set into two sets of equal size and similarly asks another question of the form "Is $x \in A_i'$?", where A_i' represents one of the smaller sets contained within A_2. The game continues play in this fashion, splitting the sets as equally as possible, maintaining integer numbers i.e. if the set is of odd size then take one set to hold one integer more than the other set[1]. After 20 questions the Questioner will have narrowed the original search space down to one integer, namely x. Hence the Questioner wins and the game ends since x has been identified.

4.2 Strategies for the liar game with one lie

Although the liar game is similar to the above "Twenty Questions" game, our strategy has to change. Consider the liar game with at most one lie in the search space $n = \{1, \ldots, 10^6\}$. It is clear that 20 questions will not suffice to find out the integer chosen by the Responder. Described beneath are three possible strategies to follow to obtain x. One strategy requires a relatively large number of questions to identify x in comparison to the other two strategies.

4.2.1 Strategy One - 41 questions

As in the original "Twenty Questions" game the Responder picks a number x from the search space $n = \{1, 2, 3, \ldots, 10^6\}$. The Questioner now has to use a strategy which identifies x, remembering that the Responder is allowed to lie at most once. As before, split the search space symmetrically into two sets and ask the question "Is $x \in A_i$?", where A_i is a partition of the search space n. To make sure the Responder has not lied, the Questioner could ask each question twice. If the answers the Questioner receives are consistent i.e. they get two Yes's or two No's, then the Questioner knows that the Responder has told the truth, since they can only lie once. The Questioner then splits this set into two equally sized smaller sets and continues to ask questions of the form "Is $x \in A_i$?". If, however, the Questioner obtains two inconsistent answers i.e. one Yes

[1]Note that $n = \frac{n+1}{2} + \frac{n-1}{2}$.

and one No, then they must ask the question for a third time and take the majority answer. The Questioner then splits the relevant set into two equally sized smaller sets and continues to ask questions of the form "Is $x \in A_i$?", but they only need to ask the question once from here on, since the lie has been expended. This strategy requires at most 41 questions, since it could one the final question in which the Responder lies. We note that 41 questions is quite large in comparison with the number of questions used to identify x in "Twenty Questions" game.

4.2.2 Strategy Two - 26 questions

As in Strategy One, the Questioner has to identify x remembering the Responder may have lied at most once. The Questioner now asks the Responder to translate the integer x into binary. Since we know that $10^6 < 2^{20}$, we know the number we eventually obtain will have at most twenty digits. The Questioner now phrases their first question as "Is the value of x in the first position a 1?". The Responder replies "Yes" or "No". The Questioner now asks the question "Is the value in the second position a 1?" and continues to ask questions of this form until they obtain a twenty-digit number consisting of 0's and 1's. We know that the Responder has lied at most once by this point i.e. at most one digit in this twenty-digit number may be wrong, so there are 21 possible integers that x could be. It is possible to identify x by using a binary search on the number i.e the Questioner splits the digit into two sets and ask questions with regard to each set. The Questioner considers the digits from positions 1 to 10 separately from the digits in positions 11 to 20. The Questioner counts up the number of 1s in the positions 1 to 10, say there are m, and asks "Are there m 1's in the first ten digits?". We need to consider the case where the Responder replies "Yes" and the case where the Responder replies "No" separately.

Case 1: The Responder answers "Yes"
Consider first that the Responder replies "Yes". We know the Responder has not lied about these ten digits. Since if they had lied at this point it means that there were not m 1's in the first ten digits. However, this implies that the Responder had lied about one of the first ten digits, which in turn implies that they have lied twice, which is not possible. Hence the Questioner should write these digits down and consider the second ten-digit number. Similarly, as above, the Questioner must now count up the number of 1's that appear in this ten-digit number and ask the question with regard to the number of 1's that appear in this number. If the Responder replies "Yes" to the question, then these ten digits were also correct and you can conclude that the Responder did not lie. If, however, the Responder answers "No" the remaining analysis is similar to Case 2.

Case 2: The Responder answers "No"
Consider that the Responder replies "No" to the initial question[2]. We do not know if this answer is a lie, but we know that they have used up the one lie that they were allowed at this point. This is simple to see as, if they answered this question truthfully it means that the Responder had previously lied. Alternatively, if the Responder had lied when answering this question, it means that these ten digits were correct and the one lie they were allowed to use has been expended. In either situation it means that the questions the Questioner continues with must all be answered truthfully. So if the

[2]i.e. referring to the question regarding the first ten-digit number.

Responder replied "No", the Questioner must then split up this ten-digit number into two five-digit numbers. For each of these five-digit numbers the Questioner counts the number of 1's and asks question with regard to the number of 1's in each digit, as before, in the knowledge that all answers from now on are truthful. This means that the Questioner should continue separating the numbers[3] into smaller numbers and asking questions with regard to the number of 1's in each number until they identify the digit that has been lied about. At this point the Questioner changes this digit from a 0 to a 1 or vice versa and thus has obtained x.

Using this strategy to determine x only requires at most 26 questions. This occurs, since when splitting the original binary number into smaller numbers we assume that the Responder answers "No" to the larger of the smaller numbers at each division[4], which results in the Questioner asking an additional 6 questions to the 20 questions they had already asked. We note that 26 questions is significantly less than the 41 required in Strategy One.

4.2.3 Strategy Three - 25 questions

Pelc [3] proves that the minimum number of questions that the Questioner has to ask to identify x in the liar game with one lie and $n = 10^6$ is in fact 25 questions[5]. Here we give a strategy, which is simpler than that of Pelc, which also achieves this. This strategy, which requires 25 questions, is very similar to the strategy used for Strategy Two. The Responder chooses an integer x between 1 and 10^6 and the Questioner asks them to translate it into binary. The Questioner phrases their questions "Is the value at the first position a 1?" and so forth. The answers the Responder gives form a twenty-digit number consisting of 0's and 1's. At this point we have used twenty questions and we have five left to use. Now we split the twenty-digit number into two numbers. How many digits should these two numbers contain? We know from Strategy Two that splitting the search space evenly into two sets requires 26 questions to identify the integer x. So assume that we split the twenty digit-number into two numbers, one containing more digits than the other. We now ask a question with regard to the larger number, so we use one question and hence have four remaining questions left to ask. Suppose the larger number contains 14 digits and the smaller number contains 6 digits. In the worst possible case we have to perform a binary search on 14 digits. So we split this number into two numbers each containing 7 digits. As explained in Section 4.2.2, we know that the Responder has expended their lie by this point, hence the answers to all questions from this point will be truthful. The Questioner continues asking questions regarding the number of 1's in the numbers and at worst will use four questions from the point where they asked about the number of 1's in the fourteen-digit number and thus they only require 25 questions to identify x. Notice that if the fourteen digit-number did

[3]When separating the numbers into smaller numbers keep them as even as possible. In the case of a number of odd length, n say, split it such that one number contains $\frac{n+1}{2}$ digits and the other number contains $\frac{n-1}{2}$ digits.

[4]i.e. the 20 digit-number splits into two numbers of length 10. The 10 digit-number splits into two numbers of length 5. The 5 digit-number splits into two numbers, one of length 3 and the other of length 2. Taking the number of greater length i.e the 3 digit-number, this splits into two numbers, one of length 2 and the other length 1. Again taking the number of greater length i.e. the 2 digit-number, this splits into two numbers of length 1.

[5]See Section 4.4 for the proof of this.

contain the correct number of 1's then the Questioner need only ask a maximum of 3 questions in regard to the six-digit number and hence only uses 24 questions at the most to identify x. Hence we have found a strategy that uses 25 questions or less to determine x in the liar game with one lie and $n = 10^6$.

4.2.4 The Algorithm for Strategy Three

The following algorithm describes precisely the strategy the Questioner should follow to identify the Responder's chosen x within 25 questions.

The Setup: Ask the Responder to choose an integer x between $\{1, \ldots, 10^6\}$ and convert this number into binary, so that they have a twenty-digit number.

Step One:
For $m = 1$ to 20.
Ask the Responder: Is the value of x in position m a 1?

IF The Responder replies YES write 1 in position m.

ELSE Write 0 in position m.

Step Two:
Split the twenty-digit number into two smaller numbers. The first number of length 14 should contain the digits from positions 1 to 14. Label this number y. The second number of length 6 should contain the digits from positions 15 to 20. Label this number z. Do not alter the order of the digits. Count the total number of 1's in y. Write this number down, say y'. Count the total number of 1's in z. Write this number down, say z'.
Ask the Responder: Are there y' 1's in y?

IF YES the digits of y are correct. Write these digits down.
Ask the Responder: Are there z' 1's in z?

- **IF** YES the digits of z are correct. Write these digits down. You have now obtained the Responder's original number, which is identical to the one you originally obtained in Step One i.e. the Responder did not use their lie.
- **ELSE** Let $a = z$. Go to Step Three.

ELSE The digits of z are correct. Write these digits down. Let $a = y$. Go to Step Three.

Step Three:
Let n denote the length of a. Split a into two smaller numbers both of length $\frac{n}{2}$. Label one of these numbers b and label the other c. Count the total number of 1's in b, say there are b'. Count the total number of 1's in c, say there are c'.
Ask the Responder: Are there b' 1's in b?

IF YES the digits of b are correct. Write these digits down. Let $a = c$.

IF $|a| \neq 1$

IF n is even repeat Step Three.

ELSE Go to Step Four.

ELSE

IF a is the number 1, change it to 0 and write it down. You have now obtained the Responder's original number. The Responder lied in the a^{th} position.

ELSE Change a to a 1 and write it down. You have now obtained the Responder's original number. The Responder lied in the a^{th} position.

ELSE The digits of of c are correct. Write these digits down. Let $a = b$.

IF $|a| \neq 1$

IF n is even repeat Step Three.

ELSE Go to Step Four.

ELSE

IF a is the number 1, change it to 0 and write it down. You have now obtained the Responder's original number. The Responder lied in the a^{th} position.

ELSE Change a to a 1 and write it down. You have now obtained the Responder's original number. The Responder lied in the a^{th} position.

Step Four:
Let n denote the length of a. Split a into two smaller numbers one of length $\frac{n+1}{2}$ and one of length $\frac{n-1}{2}$. For example, if a is equivalent to the seven-digit number representing the positions 1 to 7, it splits into the two numbers, one containing the numbers of positions 1 to 4 and the other containing the numbers of positions 5 to 7. Label the number of length $\frac{n+1}{2}$ by b. Count the total number of 1's in b, say there are b'. Label the number of length $\frac{n-1}{2}$ by c. Count the total number of 1's in c, say there are c'.
Ask the Responder: Are there b' 1's in b?

IF YES the digits of b are correct. Write these digits down. Let $a = c$.

IF $|a| \neq 1$ repeat Step Four.

ELSE

IF a is the number 1, change it to 0 and write

it down. You have now obtained the Responder's original number. The Responder lied in the a^{th} position.

ELSE Change a to a 1 and write it down. You have now obtained the Responder's original number. The Responder lied in the a^{th} position.

ELSE The digits of c are correct. Write these digits down. Let $a = b$. Repeat Step Three.

4.3 Identifying x using the concept of Spencer's chips

Consider Step One of the algorithm described in Section 4.2.4. The Questioner asks the Responder to translate their chosen integer x into binary and the Questioner proceeds to ask questions of the form "Is the value of x at position m a 1?", until they obtain a twenty-digit number [6]. Considering that the Responder may have lied at most once when responding to these questions, the Questioner obtains 21 possible binary numbers which represent x, 20 of which are pennies[7] and one that is a nonpenny. Call this state P, where $P = (1, 20)$ and has weight[8]

$$w_5(1, 20) = \binom{5}{\leq 1} + 20 = 26.$$

Clearly 26 is not a power of 2, so we add pennies to our state P until the weight is equal to a power of 2 [9]. In this case we add 6 pennies to P, such that $P = (1, 26)$ and the weight of this P is $w_5(1, 26) = 32 = 2^5$. The Questioner now needs to ask a question about a set A, such that, regardless of the Responder's reply, the weight of the new state, say P', is exactly half the weight of P i.e. the weight of P' is equal to 2^4. Suppose that A contains the one nonpenny still on the board and y pennies. Using the Endgame Lemma[10], we solve

$$\begin{aligned} & \binom{p-1}{\leq 1} + y = \binom{p-1}{\leq 0} + r - y \\ \Leftrightarrow \quad & r - y + 1 = p + y \\ \Leftrightarrow \quad & y = \frac{1}{2}(r + 1 - p), \end{aligned}$$

where r represents the total number of pennies on the board, it is possible to determine the value of pennies that appear in A. With 5 questions remaining and 26 pennies, we determine that $y = 11$ i.e. 11 pennies should be contained in A along with the one nonpenny. The Questioner asks "Is $x \in A$?". If the Responder replies "No", this implies we obtain the state $P' = (0, 16)$, with weight $w_4(0, 16) = 2^4$. The Questioner has 4

[6] We note at this point that we have used 20 questions and have 5 remaining in which to identify x.
[7] Recall Definition 3.2.1.
[8] See equation (3.5).
[9] Recall from the proof of Theorem 10 that the addition of pennies only makes the game harder for the Questioner to win.
[10] Lemma 3.2.3.

questions left to ask, by performing a binary search on these chips we obtain x using these four questions. If, however, the Responder replies "Yes", the Questioner obtains the new state $P' = (1, 11)$ with weight $w_4(1, 11) = 2^4$. The Questioner now needs to ask a question about a new set A, where A has the same properties as before[11]. Redefining r, such that $r = y = 11$, and taking $p = 4$, we obtain $y = 4$. So A now contains the one nonpenny and four pennies. Similarly as before the Questioner asks "Is $x \in A$?" and as before if the Responder replies "No", we perform a binary search on the pennies in the new state to find x, or if the Responder replies "Yes", we calculate the number of pennies in the new set A along with the one nonpenny and ask the question of the form "Is $x \in A$?" [12], repeating the process until we identify x. We note that assuming that the Responder replies "Yes" to x being contained in A, when $p = 3$, we end up with the state with one nonpenny and two nonpennies in which we can easily determine x within two questions.

So altogether, this gives us another optimal strategy which identifies x in 25 questions when $n = 10^6$ and one lie is permitted (actually it is quite similar to Strategy 3 - one can see that the use of the chip terminology makes the analysis and description a lot simpler).

In Section 4.5 we will deduce the same bound from Theorem 13, which uses a similar argument to give an optimal strategy for the case of the liar game with at most one lie and $n = 2^l$, where l is a natural number.

4.4 Pelc's lower bound

Pelc [3] gives three lemmas which use general results to find an exact solution to Ulam's problem i.e. the minimum number of questions required to solve the liar game with one lie and $n = 10^6$. The focus of this section will be looking at the first lemma Pelc describes, which discusses a lower bound on the number of questions required so that the Questioner can win the liar game with one lie. The lemma splits into two cases, one for even n and one for odd n, where n is the size of the search space.

If the idea of a game is to identify an element x from search space $\{1, \ldots, n\}$ within q questions, the game is known as an $[n, q]$ game. The Questioner wins the $[n, q]$ game if they are able to identify x using at most q questions. The Responder wins the $[n, q]$ game if they use an adversary strategy[13], and the Questioner is unable to identify x within q questions. Hence, either the Responder wins or the Questioner wins the $[n, q]$ game.

At each stage of the liar game, when the Questioner asks a question, we can define *state* of the game as (x_0, x_1), where x_0 and x_1 are natural numbers. x_0 represents the size of the "truth" set i.e. the set of elements of $\{1, \ldots, n\}$ that satisfy the answers given to all previous questions. x_1 represents the size of the "lie" set i.e. the set of the elements of $\{1, \ldots, n\}$ that satisfy the answers given to all previous questions except one, so the lie has been expended. This is a simple version of the chips game discussed by Spencer in [8].

[11] i.e. A contains the one nonpenny and y pennies.

[12] When $p = 3$ and $r = 4$, then $y = 1$.

[13] The adversary strategy is a strategy in which the Responder does not need to actually pick a number x from the search space, but answer questions consistently so that there exists an element of the search space that satisfies all but one answer.

The weight of the state (x_0, x_1) with j questions remaining is defined

$$w_j(x_0, x_1) = x_0(j+1) + x_1.$$

This can be interpreted by considering that each element of x_0 gives $j+1$ possibilities of lying to each of the remaining j questions or not lying at all. In x_1, the Responder must tell the truth to each answer, since you are working on the assumption the lie has been expended, so each element in x_1 gives just one possibility.

After the Questioner poses any question the state (x_0, x_1) gives rise to two states $(\widehat{x_0}, \widehat{x_1})$ and $(\widetilde{x_0}, \widetilde{x_1})$, corresponding to answers of "Yes" and "No" respectively. This implies the following equation

$$w_q(x_0, x_1) = w_{q-1}(\widehat{x_0}, \widehat{x_1}) + w_{q-1}(\widetilde{x_0}, \widetilde{x_1}) \tag{4.1}$$

Claim The result of equation (4.1) holds.

Proof To show that (4.1) holds, we need to be clear what $\widehat{x_0}$, $\widehat{x_1}$, $\widetilde{x_0}$, $\widetilde{x_1}$ represent. Consider the state (x_0, x_1) as shown in Figure 4.1. When the Responder answers a question suppose that the black circles are the chips which refer to the "Yes" reply and the white circles are the chips which refer to the "No" reply. Label the black chips in x_0 by x_0'. Label the black chips in x_1 by x_1'. This is represented in Figure 4.2. When the Responder replies to a question the state (x_0, x_1) splits into the following two states $(\widehat{x_0}, \widehat{x_1})$ and $(\widetilde{x_0}, \widetilde{x_1})$, which are shown in Figure 4.3.

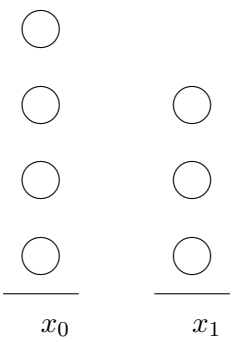


Figure 4.1: An example of a state

Hence, from the diagrams we obtain

$$\widehat{x_0} = x_0'$$
$$\widehat{x_1} = x_1' + (x_0 - x_0') = x_1' + x_0 - \widehat{x_0}$$

and

$$\widetilde{x_0} = x_0 - x_0' = x_0 - \widehat{x_0}$$
$$\widetilde{x_1} = x_0' + x_1 - x_1' = \widehat{x_0} + x_1 - x_1'.$$

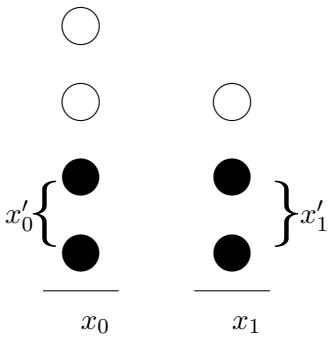


Figure 4.2: Highlighting the chips which represent the "Yes" and "No" answers.

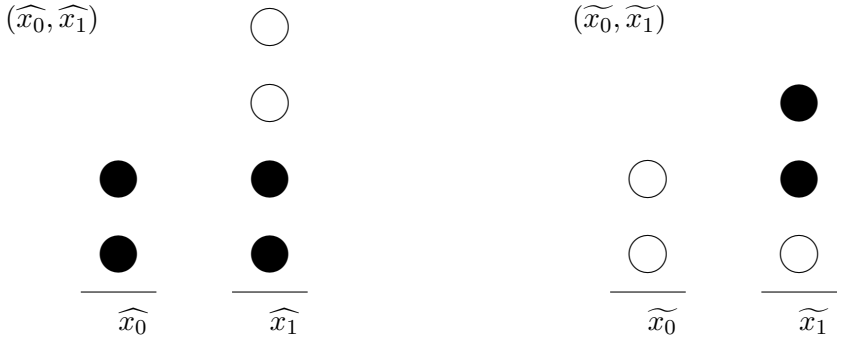


Figure 4.3: How the state (x_0, x_1) changes after a question is answered.

Now evaluate the right hand side of (4.1)

$$\begin{aligned} w_{q-1}(\widehat{x_0}, \widehat{x_1}) + w_{q-1}(\widetilde{x_0}, \widetilde{x_1}) &= (\widehat{x_0}q + \widehat{x_1}) + (\widetilde{x_0}q + \widetilde{x_1}) \\ &= \widehat{x_0}q + ((x_0 - \widehat{x_0}) + x_1') + \widetilde{x_0}q + (\widehat{x_0} + x_1 - x_1') \\ &= \widehat{x_0}q + \widetilde{x_0} + \widetilde{x_0}q + \widehat{x_0} + x_1 \\ &= x_0(q+1) + x_1 \\ &= w_q(x_0, x_1). \end{aligned}$$

□

The following theorem provides a condition for which the Responder will always possess a winning strategy for the liar game for a search space of size n and q questions and one lie.

Theorem 12 *(a) For even n the Responder wins the $[n, q]$ game if $n(q+1) > 2^q$.*
(b) For odd n the Responder wins the $[n, q]$ game if $n(q+1) + (q-1) > 2^q$.

Proof (a) Let the Responder play the adversary strategy, where the Responder always chooses the state of larger weight after each question has been asked. The weight of the initial state is given by

$$w_q(n, 0) = n(q+1) > 2^q.$$

After at most q questions the weight of the resulting state (x_0^*, x_1^*) will be at least 2, since after q questions

$$w_0(x_0^*, x_1^*) > 2^0 = 1.$$

Since we are only concerned with natural numbers, this implies that after at most q questions the weight of (x_0^*, x_1^*) is at least 2. We need to show is that such a state with weight at least 2 cannot be equal to the state $(1, 0)$. Clearly the state $(1, 0)$ would only occur if the state prior to it was $(1, c)$, where c is a natural number of size less than n, with j questions remaining. The Questioner asks a question with regard to the single element in x_0 of $(1, c)$, hence, we obtain the states $(1, 0)$ and $(0, c+1)$. Assume that the weight of $(0, c+1)$ is at most that of $(1, 0)$ i.e.

$$w_{j-1}(1, 0) \geq w_{j-1}(0, c+1),$$

which in turn implies

$$j \geq c + 1 \tag{4.2}$$

Now since the Responder is playing the adversary strategy they pick the new state of greater weight, so the Responder chooses the state $(1, 0)$.

The state $(1, c)$ was reached after $q - j$ questions using the adversary strategy. The weight of this state is

$$w_j(1, c) = j + 1 + c > 2^q \times 2^{-(q-j)} = 2^j \tag{4.3}$$

i.e. the weight of total number of questions to begin with the weight of the number of questions asked by this point removed. Substituting (4.2) into (4.3) implies

$$\begin{aligned} & 2j > 2^j \\ \Rightarrow \qquad & j > 2^{j-1}. \end{aligned}$$

However, $j \not> 2^{j-1}$ for all j, where j is a natural number. Hence, $(1,0)$ was not the state of larger weight and it therefore contradicts the Responder's adversary strategy i.e. by the adversary strategy the Responder would not have chosen the state $(1,0)$ in order to win the game.

(b) For odd n, any question asked at the beginning of the $[n,q]$ game gives the states $(\widehat{x_0},\widehat{x_1})$ and $(\widetilde{x_0},\widetilde{x_1})$. We note that

$$\frac{n+1}{2}+\frac{n-1}{2}=n.$$

It is easy to see that

$$q\frac{n+1}{2}+\frac{n-1}{2} \geq q\frac{n-1}{2}+\frac{n+1}{2}.$$

Hence,

$$\max(w_{q-1}(\widehat{x_0},\widehat{x_1}), w_{q-1}(\widetilde{x_0},\widetilde{x_1})) \geq q\frac{n+1}{2}+\frac{n-1}{2}.$$

Assuming the Responder adopts the adversary strategy, the first question yields at state of weight at least $q\frac{n+1}{2}+\frac{n-1}{2}$. By assumption

$$n(q+1)+(q-1) > 2^q.$$

Rearranging this gives

$$\begin{aligned} & q(n+1)+(n-1) > 2^q \\ \Rightarrow \qquad & q\frac{n+1}{2}+\frac{n-1}{2} > 2^{q-1}. \end{aligned}$$

So after at most $q-1$ questions, the adversary strategy gives a state of weight if at least 2 . Following the proof of (a), it can be shown that $(1,0)$ is not the state the Responder chooses and hence, by playing the adversary strategy the Responder wins. □

We note that part a) of the theorem is equivalent to the case $k=1$ of Theorem 8 in Chapter 3. To see this consider the weight of the game with search size space n, q questions and at most one lie i.e. $k \geq 1$. Recall (3.1)

$$p_i^* = \sum_{i=0}^{k} \binom{q}{k-i} \times 2^{-q},$$

which defines the weight of an individual chip on position i and the weight of a state is defined in (3.2) as

$$w^* = \sum_{i=0}^{k} x_i p_i^*.$$

Now at the start of play all of the chips lie on position x_0 i.e. there are n chips on x_0 and no chips on x_1, which implies the weight of the state is

$$n \times \left(\binom{q}{1} + \binom{q}{0} \right) \times 2^{-q} = n(q+1) \times 2^{-q}.$$

Theorem 8 states that the weight of a state is greater than 1 i.e.

$$\begin{aligned} & n(q+1) \times 2^{-q} > 1 \\ \Rightarrow \qquad & n(q+1) > 2^q. \end{aligned}$$

In the case of odd, the additional term $(q-1)$ means part b) of Theorem 12 is a little stronger than Theorem 8.

Using Theorem 12 we can show that even with an optimal strategy the Questioner requires at least 25 questions to win the Ulam's searching game with one lie. Note that if we suppose $q = 24$, by substituting $q = 24$ and $n = 10^6$ into Theorem 12 part a) gives

$$10^6 \times 25 > 2^{24}.$$

Thus the Responder wins when $q = 24$ i.e. there does not exist an optimal strategy for $q = 24$ with which the Questioner can win.

4.5 The general strategy for the liar game with one lie of search space size $n = 2^l$

Consider the liar game with one lie and a search space of size n, such that $n = 2^l$, where l is a natural number. The Responder chooses an integer x from $n = 2^l$ and within q questions the Questioner must identify x, where q is sufficiently large. The initial state of the game is $P = (n, 0)$, which has initial weight[14]

$$w_q(x_0, x_1) = x_0(q+1) + x_1 = n(q+1).$$

Theorem 13 *If $n \leq \frac{2^q}{q+1}$ and $n = 2^l$, where l is a natural number, then the Questioner has a strategy which identifies x in q questions.*

Proof Firstly we note that the weight of the initial state is

$$w_q(n, 0) = n(q+1) \leq 2^q.$$

In addition we note that[15]

$$l \leq q - \log(q+1),$$

and since l is a natural number, this implies

$$l \leq q - \lceil \log(q+1) \rceil \tag{4.4}$$

[14] See Chapter 3 Equation (3.5).

[15] Remember that we are working with logarithms to the base 2.

Consider each integer of $n = 2^l$ in its binary form i.e. we have 2^l strings of length l. The Questioner performs a binary search on these numbers by asking questions of the form "Is the value of x in position m a 1?". The binary search on the search space n uses exactly l questions and as a result we obtain $l+1$ possible binary numbers for x. There is exactly one number which satisfies all the answers, hence we consider this number as a nonpenny. There are also l numbers which satisfy all but one answer, so we consider these numbers as pennies[16]. Therefore, after the binary search has been performed we obtain the position $P^*(1, l)$, shown in Figure 4.4.

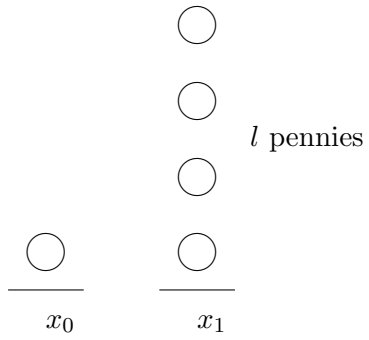


Figure 4.4: The Questioner's position after l questions i.e. $P^* = (1, l)$

Moreover,

$$\begin{aligned} w_{q-l}(x_0, x_1) &= 1 \times (q - l + 1) + l \times 1 \\ &= q + 1 \\ &= \frac{w_q(n, 0)}{n} = \frac{w_q(n, 0)}{2^l}, \end{aligned}$$

where $x_0 = 1$ and $x_1 = l$. This shows that the Questioner has played optimally so far, as after each question has been asked, the weight been exactly has halved.

Let $p = q - l$. By (4.4), it now suffices to identify x within $p = \lceil \log(q+1) \rceil$ questions. Note that the weight of the state P^* is $2^{p-1} < w(1, l) \leq 2^p$ [17]. Suppose that $q+1$ is not a power of 2. Similarly, as in the proof of Theorem 10, it is easy to see that we can add pennies to the state until the total weight is equal to 2^p, as the addition of pennies will only make the game harder for the Questioner. Suppose that we now have r pennies in total, so we obtain the new state $\widehat{P^*} = (1, r)$, with $r \geq l$, where the weight of $\widehat{P^*}$ equals 2^p. So we obtain

$$p + 1 + r = w_p(1, r) = 2^p \tag{4.5}$$

We now have two cases to consider:

Case One: If $r < p + 1$, then

$$\begin{aligned} & 2(p+1) > 2^p \\ \Rightarrow \quad & (p+1) > 2^{p-1}, \end{aligned}$$

[16] See Chapter 3 Definition 3.2.1.

[17] The lower bound occurs since $q+1$ may not be equal to a power of 2 and if it was less than or equal to 2^{p-1}, we would require less than p questions to identify x.

which holds if and only if $p \leq 2$. This means that we have one nonpenny and at most two pennies. Therefore, the Questioner can easily identify x in two questions.

Case Two: If $r \geq p+1$, then

$$\begin{aligned} & 2^p \geq 2(p+1) \\ \Rightarrow \qquad & 2^{p-1} \geq p+1, \end{aligned}$$

which holds if and only if $p \geq 3$.

We know that the total weight of this state is even, so we wish to find a set, say A, such that when a question is asked about it, regardless of the Responder's reply, the weight is exactly halved, which implies that we are still playing optimally. This is equivalent to applying the Endgame Lemma[18]. Since this strategy is simpler than that of the Endgame Lemma, we give a self contained argument here. Assume that A contains the nonpenny and y pennies and that the weight of A is equal to 2^{p-1}. Suppose that x is contained in A, then the weight of the resulting state is

$$\binom{p-1}{\leq 1} + y = p + y.$$

If, however, x is not contained in A, the resulting state has the weight

$$\binom{p-1}{\leq 0} + r - y = r - y + 1.$$

Thus we wish to solve

$$\begin{aligned} & r - y + 1 = p + y \\ \Leftrightarrow \qquad & y = \frac{1}{2}(r + 1 - p) \end{aligned} \tag{4.6}$$

We note that (4.5) implies $(r+1-p)$ is even and so y is an integer. Moreover, the condition $r \geq p$, implies that y is an natural number i.e. $y \geq 0$.

Suppose that the Questioner asks "Is $x \in A$?". Here we have two possible outcomes to consider.

Outcome One: The Responder replies "Yes".
If the Responder replies "Yes", we obtain a position P', which consists of one nonpenny and y pennies i.e. $P' = (1, y)$, which has weight 2^{p-1}. If $p - 1 \leq 2$, then by Case One, the Questioner can easily identify x. If $p - 1 > 2$, we redefine r, such that $r = y$ and then calculate the new value of y by (4.6), to obtain a new set A. The Questioner now asks "Is $x \in A$?" for the new A. If the Responder replies "Yes", we repeat this step again, however, if the Responder replies "No", we refer to result of Outcome Two.

Outcome Two: The Responder replies "No".
If the Responder replies "No" then we obtain a position P', which consists only of pennies i.e. $P' = (0, r - y + 1)$, which has weight 2^{p-1}. Since we have $p - 1$ questions remaining we perform a binary search on the $r - y - 1$ pennies remaining and after $p - 1$ questions we will have identified x.

[18] Lemma 3.2.3.

Since after each question has been asked we are playing optimally i.e. we reduce the weight of the state exactly by half, within p questions, from the point of adding pennies, the Questioner is able to identify x. Hence within q questions, as required, the Questioner can identify x. $\square$

Chapter 5

Conclusion

One of the main aims in this project was to identify an optimal strategy in which it is possible for the Questioner, using at most 25 questions, to win the liar game with one lie and $n = 10^6$. Chapter 4 Section 4.2.4 provided an algorithm (called Strategy 3) for which it was possible for the Questioner to identify x within 25 questions and as a consequence a winning strategy for the liar game with one lie and $n = 10^6$. Furthermore, using the Endgame Lemma result of Theorem 9 in Chapter 3, we were able to provide an alternative optimal strategy, described in terms of chips, which was a simpler version of Strategy 3. It was shown in Chapter 2, that there exists a 1 error-correcting code that is able to optimally solve the liar game with one lie and $n = 10^6$. However, we are able to conclude that for search spaces of different sizes, that the code used to solve our original liar game is not an appropriate strategy to solve liar games with one lie and $n \neq 10^6$. Additionally, in Chapter 4 we gave an optimal strategy for the liar game with one lie and $n = 2^l$, where l is a natural number, by using a simpler form of the Endgame Lemma described in Chapter 3. Besides these exact results, we presented an approximate result due to Spencer [8], which asymptotically solves the liar game for a large search space and a fixed number of lies.

It would be desirable to extend the strategies discussed in Chapter 4 to consider the liar game with more than one lie. In particular we could extend this project by discussing Ulam's liar game with two lies and $n = 10^6$. If we were to apply a similar strategy as to that in Strategy 2 or Strategy 3, described in Chapter 4, we would obtain a twenty-digit binary number. However, instead of having only 21 possible binary numbers representing x, as with one lie, we have 211 possible binary numbers representing x, since

$$\binom{20}{2} + \binom{20}{1} + \binom{20}{0} = 211$$

(Indeed if the Responder had lied exactly i times, there are $\binom{20}{i}$ candidates for the value of x). This clearly would require a more complex strategy to solve optimally than that required in Strategy 3. In [1] Czyzowicz, Mundici and Pelc showed that the minimum number of questions required by the Questioner for the liar game with two lies and $n = 10^6$ was 29.

Another extension of this project could be to investigate the consequences of Rényi's [5] description of the liar game, where we consider that the Responder lies a given percentage of the total number of questions. Further extensions to the liar game are discussed in Pelc [4].

Appendix A

Proof of Theorem 2.2

Let $q = 1 - p$. Hence (2.2) becomes

$$C(p) = 1 + p \log p + q \log q.$$

Suppose that source emits 0 with probability α and 1 with probability $\beta = 1 - \alpha$. Then the output ζ has the distribution [1]

$$\begin{aligned} &0 \textit{ with probability } \alpha q + \beta p \\ &1 \textit{ with probability } \beta q + \alpha p \end{aligned}$$

Using the definition of information, we have

$$\begin{aligned} I(\varphi|\zeta) &= H(\varphi) - H(\varphi|\zeta) \\ &= p \log p + q \log q - (\alpha q + \beta p) \log(\alpha q + \beta p) - (\alpha p + \beta q) \log(\alpha p + \beta q) \\ &= p \log p + q \log q - (\alpha(q-p) + p) \log(\alpha(q-p) + p) \\ &\quad - (\alpha(p-q) + q) \log(\alpha(p-q) + q). \end{aligned}$$

To find α, differentiate $I(\varphi|\zeta)$ with respect to α.

$$\begin{aligned} \frac{dI(\varphi|\zeta)}{d\alpha} &= -\big((q-p) \log(\alpha(q-p) + p) + (p-q) \log(\alpha(p-q) + q) \\ &= -(q-p) \log \Big(\frac{\alpha(q-p) + p}{\alpha(p-q) + q} \Big) \\ &= 0 \end{aligned}$$

$$\begin{aligned} \Rightarrow \qquad \frac{\alpha(q-p) + p}{\alpha(p-q) + q} &= 1 \\ \alpha(q-p) &= (1-\alpha)(q-p) \\ \textit{i.e.} \qquad \alpha &= \frac{1}{2}. \end{aligned}$$

Take the second derivative of $I(\varphi|\zeta)$ to show that α is maximum.

$$\frac{d^2 I(\varphi|\zeta)}{d\alpha^2} = \frac{-(q-p)^2}{\alpha(q-p) + p} - \frac{(p-q)^2}{\alpha(p-q) + q}.$$

[1] The probability 0 is transmitted correctly + the probability 0 is not transmitted correctly. Similarly this can be shown for the probability for the number 1.

Now since p, q, $\alpha > 0$

$$\Rightarrow \qquad \frac{d^2 I(\varphi|\zeta)}{d\alpha^2} < 0$$

$$\Rightarrow \qquad \alpha = \frac{1}{2} \qquad is\ maximum$$

$$\Rightarrow \qquad (2.1). \qquad \square$$

Appendix B

The operations on G

We have

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Now swap the 3^{rd} and 6^{th} columns of G and swap 1^{st} and 4^{th} columns of G to obtain

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Add the 4^{th} row to the first row 1^{st} to get

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Now add the 1^{st} row to the 2^{nd} row and we end up with

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Appendix C

Perfect information game

Consider a two player game, in which each player aims to win the game. Assume that the game is finite and that all the information is given to each player before play of the game begins. This is known as a **zero-sum perfect information game**. A consequence of the game being of perfect information is that it can be strictly determined i.e. there exists a winning strategy for one of the players. The following result is shown in [14].

Claim Every zero-sum two-player finite game of perfect information is strictly determined.

Proof Suppose that we have two players, Player 1 and Player 2 and assume that the outcome of the game is not determined before play begins i.e. neither player holds a winning strategy before play begins. Further, assume that there exist a finite number of moves in which the players have to win the game. The first move made by Player 1 cannot be part of Player 1's winning strategy, since this would imply that the outcome of the game is determined before play began. Similarly, this first move cannot form part of Player 2's winning strategy. Therefore, the first move of the game is non-deterministic. This implies that all the following moves cannot form part of a winning strategy, since you could start the game in the position obtained as a result of making the first move and reduce play of the game by one move. Hence the game is infinite, since no move can be deterministic. However, this contradicts the assumption that the game is finite. Therefore, the game is strictly determined i.e. there exists a winning strategy for one of the players.

Note that although a winning strategy exists for one of the players there is no guarantee they will find it during play and make use of it.

Appendix D

Fact $\binom{j}{l} + \binom{j}{l-1} = \binom{j+1}{l}$

We can show that

$$\binom{j}{\leq k-i} + \binom{j}{\leq k-i+1} = \binom{j+1}{\leq k-i}$$

by using the property

$$\binom{j}{l} + \binom{j}{l-1} = \binom{j+1}{l}.$$

Claim

$$\binom{j}{l} + \binom{j}{l-1} = \binom{j+1}{l}.$$

Proof We know that $\binom{j+1}{l}$ is equal to the number of choices for l balls in a box with $j+1$ balls. Now consider a box with j red balls and one blue ball. The number of ways of choosing l red balls is $\binom{j}{l}$. The number of ways of choosing the one blue ball and $l-1$ red balls out of j is $\binom{j}{l-1}$.

Hence

$$\binom{j}{l} + \binom{j}{l-1} = \binom{j+1}{l}.$$

□

It follows from this that

$$\binom{j}{\leq l} + \binom{j}{\leq l-1} = \binom{j+1}{\leq l}.$$

Let $l = k - i$, which implies

$$\binom{j}{\leq k-i} + \binom{j}{\leq k-i-1} = \binom{j+1}{\leq k-i}.$$

Appendix E

Fact $(1-x)^k \geq 1-xk$

Show that

$$(1-x)^k \geq 1-xk \tag{E.1}$$

for all $a \geq 0$.

Inductive statement:

$$P_k = (1-x)^k \geq 1-xk.$$

Induction start: Let $k = 0$. Then

$$(1-x)^0 = 1 = 1 - x \times 0$$

Therefore, P_k holds for $k = 0$.

Inductive step: Suppose that the inductive statement P_k holds for $k = a$ i.e.

$$(1-x)^a \geq 1-xa.$$

Show that the inductive statement holds for $k = a+1$.

$$\begin{aligned} (1-x)^{a+1} &= (1-x)^a \times (1-x) \geq (1-xa) \times (1-x) \\ &= 1 - xa - x + x^2 a \\ &= 1 - x(a+1) + x^2 a \geq 1 - x(a+1) \\ \Rightarrow \quad (1-x)^{a+1} &\geq (1-x(a+1)). \end{aligned}$$

Therefore, P_k holds for $k = a+1$ and so we have proven (E.1).

Bibliography

[1] J. Czyzowicz, D. Mundici, A. Pelc, *Solution of Ulam's problem on binary search with two lies*, J. Combin. Theory Ser. A 49 (1988), 384-388

[2] Andrzej Pelc, *Coding with bounded error fraction*, Ars Combin. 24 (1987), 17-22

[3] Andrzej Pelc, *Solution of Ulam's Problem on searching with a lie*, Journal of Combinatorial Theory, Series A 44 (1987), 129-140

[4] Andrzej Pelc, *Fundamental Study: Searching games with errors - fifty years of coping with liars*, Theoretical Computer Science 270 (2002), 71-109

[5] A. Rényi, *On a problem of information theory*, MTA Mat. Kut. Int. Kozl. 6B (1961) 505-516

[6] M. Sereno, *Binary search with errors and variable cost queries,* Inform. Process. Lett 68 (1998), 261-270

[7] J. Spencer, *Guess a Number - with Lying*, Math. Mag. 57, No. 2, (1984), 105-108

[8] Joel Spencer, *Ulam's searching game with a fixed number of lies*, Theoretical Computer Science 95 (1992), 307-321

[9] J. Spencer, P. Winkler, *Three thresholds for a liar*, Combin. Probab. Comput. 1 (1992), 81-93

[10] S.M. Ulam, *Adventures of a Mathematician*, Scribner, New York (1976), 281

[11] Dominic Welsh, *Codes and Cryptography*, Oxford Science Publications (2000)

[12] *Bounds for unrestricted binary codes*

`www.s2.chalmers.se/~agrell/bounds/unr.html`

[13] *Zero-Sum Two Person Games of Perfect Information*

`http://library.thinkquest.org/26408/math/2person_zs_pi.shtml`

[14] *Why Two Person Zero-Sum Games of Perfect Information are Strictly Determined*

`http://library.thinkquest.org/26408/math/strict.html`