Combinatorial constructions for error-correcting codes

(An expositionary paper in Combinatorial Game Theory & Coding Theory)

By Yuvraj Sarda

August 2021

In data storage and transmission, bits of data often get altered (or corrupted) in unpredictable fashions. This can radically change the meaning of the data. To address this issue, we add some redundancy to the data. This enables us to retrieve the actual data even if there is some corruption. The amount of redundancy varies depending upon the efficiency of the coding algorithm and the amount of corruption it accounts for. The theory of error-correcting codes studies these coding algorithms.

Since the 1980s, new constructions have been found for these error-correcting codes, many of which arise from combinatorics and combinatorial games. In this paper, I study these constructions and the connections they have to error-correcting codes. I begin with some definitions and basic results in §1. Next, in §2, I discuss the connection between the game of Nim and Hamming codes. I then present generalisations and extensions of these results from the game theory (§3) and coding theory (§4) perspectives. Finally, in §5 I explore some properties of perfect codes.

Note: Many of the results (1.1, 1.2, 3.1, 4.5, and 5.2) were independently derived and proved. However, since they resemble or take ideas from the cited sources, they likely don't count as new results. The main references for each section have been cited in the section headers. Results from other sources have been cited clearly too.

1 Some definitions from coding theory[1][2]

Definition 1.1. We call the intended message the *codeword* and the set of all codewords *codes*.

Definition 1.2. The *word length* of a code is the number of digits in each message and is denoted by n. The *base* of a code refers to the number of unique digits used to represent messages and is denoted by B. A code that corrects e errors is called an *e-error-correcting code*.

In this paper, we are largely concerned with linear codes.

Definition 1.3. A *linear* code is a code such that any linear combination $(ax + by)$ of its codewords results in another codeword.

There is a key metric used to evaluate linear codes called the Hamming distance, d:

Definition 1.4. The *Hamming distance* of two codewords is the number of positions in which they differ. The Hamming distance of a code is the minimal Hamming distance between any two of its codewords.

Theorem 1.1. *A code* corrects *upto e errors if and only if its Hamming distance $\geq 2e+1$.*

Proof. Let P and Q be any two codewords from this code. We first show the forward direction and then the backward direction.

1. Let the code have a Hamming distance $\geq 2e+1$. Then P and Q differ in at least $2e+1$ positions. Thus, any message that differs from P in at most e positions differs from Q in at least $e+1$ positions. Hence if a message can be interpreted as P accounting for upto e errors, it cannot be interpreted as Q. Likewise, if a message can be interpreted as Q accounting for upto e errors, it cannot be interpreted as P. Thus, this code can correct e errors.

2. Let the code be an e-error-correcting code. Then a message that is interpreted as P may differ from it in upto e positions. Now no message may be interpreted as 2 codewords, otherwise we would be unable to decipher its meaning. Thus, any messages interpreted as Q differ from P in at least $e+1$ positions. This message may differ from Q in e positions too. Thus, P and Q must differ in at least $(e+1)+(e) = 2e+1$ positions. □

Remark. *As in [3], sometimes authors use Theorem 1.1 to formally define the e-error-correcting code as "a set of words with minimal Hamming distance $\geq 2e+1$."*

Theorem 1.2. *A code* detects *upto e errors if its Hamming distance $\geq e+1$*

Proof. Since codewords differ in at least $e+1$ positions, any message that differs from a codeword in $< e+1$ positions is not a codeword. Thus, we know that an error has occurred. □

2 Hamming Codes and the game of Nim[2]

Let's now define a famous error correcting code. I shall omit its construction as it isn't directly connected to the discussion.

Definition 2.1. A *Hamming code* is a 1-error-correcting code. If its length $n = 2^a - 1$, then we call it a *perfect Hamming code*. If $n = 2^a$ then we call it an *extended Hamming code*. For all other lengths, we call them *shortened Hamming codes*.

Remark. *Although Hamming codes are typically used in base 2, that is not necessary. We shall see the generalised definition in Theorem 4.4.*

Here is an example of a Hamming code of length 7:

0000000,	0101010,	1001011,	1100001,
0000111,	0101101,	1001100,	1100110,
0011001,	0110011,	1010010,	1111000,
0011110,	0110100,	1010101,	1111111

This code has a very nice connection to the famous and widely-studied game of Nim.

Definition 2.2. *Nim* is an impartial game where two players alternate moves. In any Nim position, there are several piles of stones. On their turn, the players must remove some stones from any one pile. The first player without any moves loses. Let $*n$ denote a Nim pile of size n. A Nim game is therefore represented as $*a_1 + *a_2 + ... + *a_n$ where $\{a_k\} \in \mathbb{N}$.

Theorem 2.1. *Any Nim game can be represented as a single string of 0s and 1s.*

Proof. It is well known that if there are 2 Nim piles of the same size in any game G, they may be eliminated without changing the outcome of the game. This is because whenever a player moves from $*n + *n$ to $*k + *n$ where $k < n$, the other player can move to $*k + *k$.

This carries on until this pile is emptied, with the player who moves first in the equal piles having to move first again in some other pile.

This means that any Nim game can be represented as a sum of distinctly sized piles. Thus, the sequence $...a_4a_3a_2a_1$ represents this game, where $a_k \in \{0, 1\}$ represents the presence of a Nim pile of size k. □

In this representation, we have obtain 2 types of Nim moves.

1. Removing an entire pile: This is equivalent to flipping a 1 to a 0 in the binary string.
2. Reducing the size of a single pile: This is equivalent to removing the entire pile, and then adding a smaller pile of reduced size. So, this move is just flipping the digits corresponding to the initial and final pile sizes.

Example. The Nim game $*5 + *7 + *2 + *1 + *2 + *3$ is represented as 1010101 in our notation. If we move in $*7$ to $*3$, then the corresponding move in our notation is to move to position 0010001, where we may ignore the leading 0s to get 10001.

Interpreting Nim in this binary fashion allows us to obtain the following profound result:

Theorem 2.2. *The codewords in Hamming code are the $\mathcal{P}$ positions in the binary representation of Nim*

Proof. Let $\mathscr{P}$ denote the set of all codewords, interpreted as Nim games. Let $\mathscr{N}$ denote the set containing all the other messages, again interpreted as Nim games.

1. Consider the $\mathscr{N}$ positions. Since this is not a codeword, it differs from some codeword in < 3 positions. We can thus move to this position by changing 1 or 2 of the positions. This is permissible by the types of Nim moves obtained in binary representation. Thus, from any $\mathscr{N}$ position, we can move to a $\mathscr{P}$ position.
2. Now, consider the $\mathscr{P}$ positions. Since this is a codeword, it differs from another codeword in at least 3 positions. Changing ≥ 3 positions is not possible using just the types of Nim moves obtained in binary representation. Thus, the only moves we can make from a $\mathscr{P}$ position, is to an $\mathscr{N}$ position.

It now follows from the famous partition theorem that $\mathscr{P}$ positions and $\mathscr{N}$ positions are actually just the $\mathcal{P}$ and $\mathcal{N}$ positions of the game. □

Remark. *This theorem reveals the deep connection between heap games like Nim and linear codes. We shall look at this more generally, first in theorem 3.1 and then in theorem 4.1.*

3 Turning Turtles: the generalisation from Game Theory

The game of Turning Turtles is a generalisation of Nim. Analysing it similarly, we find that it has connections to codes that can detect/correct multiple errors.

Definition 3.1. *Turning Turtles* is an impartial game where two players alternate turns. There are n turtles placed in a row. These turtles may be upright, or upside-down. On their turn, a player must flip between 1 and k turtles, with the added condition that the leftmost turtle must be flipped from upside-down to upright. The first player without any moves left loses.

Example. The case k=2 is just Nim in the binary form.

It is easy to see how a Turning Turtles position may be represented as a binary string. We simply use 1 to denote that a turtle is upside-down and 0 to denote that it is upright. The location of a digit in the binary string corresponds to the location of the corresponding turtle in the row.

Theorem 3.1. *The set of $\mathcal{P}$ positions of Turning Turtles with $k = 2e$ permissible flips forms a linear e-error-correcting code.*

Proof. It is well known that there is no move from any $\mathcal{P}$ position to another $\mathcal{P}$ position, since otherwise the former won't be a $\mathcal{P}$ position. This means that these P positions differ in at least $2e + 1$ positions from each other i.e. their hamming distance is $\geq 2e + 1$. It follows from Theorem 1.1 that a code consisting of these P positions can correct e errors. It also follows from Theorem 1.2 that this code can detect $2e$ errors. Furthermore, since the value of $\mathcal{P}$ positions $= 0$, the nim-sum of any two P positions is also 0, another $\mathcal{P}$ position. Thus, as per definition 1.3, this error correcting code is also a linear code. □

Example. The $\mathcal{P}$ positions of Turning Turtles with $k = 6$ form the famous perfect binary Golay code of length 23, which corrects for 3 errors and detects 6 errors. The game of Mogul is Turning Turtles with $k = 7$. It thereby forms the extended binary Golay code which corrects 3 errors and detects 7 errors.

Remark. *NASA is known to have used the Golay codes to encode colored photos for its Voyager missions. This was because the data transmission channels were quite noisy and repeating any transmissions was unfeasible due to memory, bandwidth and time constraints.*

Example. The game of Möbius is Turning Turtles with $k = 5$. The $\mathcal{P}$ positions of this game has some surprising connections to the famous Möbius transformation from complex analysis. Interestingly, the set of its $\mathcal{P}$ positions also form the binary extended quadratic residue code of length 18.

4 Lexicodes: the generalisation from Coding Theory[4]

The Hamming code and the Golay code are just examples from a much larger class of codes called lexicographic codes (or lexicodes). These codes are constructed similarly and share much of the connections with combinatorial games.

Definition 4.1. A *lexicographic code* is one that can be constructed using the following generic greedy algorithm:

1. We work through all words in alphabetical (or lexicographic) order.
2. Each time we encounter a codeword that is not prohibitively near earlier codewords (by some metric), we select it.

4.1 Generalising Theorems 2.2 & 3.1 to all heap games

Definition 4.2. A *heap game* is an impartial game consisting of several heaps of stone. These rules of these games can be defined using *turning sets*, which are of the general form $\{a_0, a_1, a_2, ...a_n\}$ where $\{a_k\}$ is a decreasing sequence of non-negative integers. Using the appropriate turning set (if one exists), on their turn, a player may replace a pile of size a_0 with the series of smaller piles of $a_1, a_2, \cdots, a_n$. As usual, the player without any moves left loses.

Example. As seen below, Nim is defined by the family of turning sets $\{a_0, a_1\}$ such that $a_0 > a_1 \geq 0$.

$$
\begin{aligned}
&\{1,0\}\\
&\{2,0\},\{2,1\}\\
&\{3,0\},\{3,1\},\{3,2\},\\
&\{4,0\},\{4,1\},\{4,2\},\{4,3\}\\
&\{5,0\},\{5,1\},\{5,2\},\{5,3\},\{5,4\}\\
&\quad\vdots\\
&\{a_0,0\},\{a_0,1\},\{a_0,2\},\cdots,\{a_0,(a_0-1)\}
\end{aligned}
$$

Theorem 4.1. *For any turning set and any base, the $\mathcal{P}$ positions of the game correspond to the codewords in the lexicode.*

Proof. The proof is by induction on the position G. We must check for 2 things:

1. If G is not in the lexicode, this must be because there is a smaller number G' in the lexicode for which $\{G, G'\}$ is a turning set. Therefore, by the induction hypothesis, the move from G to G' is a winning move, and G is not a winning position.

2. If G is in the lexicode, and G to G' is any legal move, then $G' < G$ and $\{G, G'\}$ is a turning set. Since we accept G we must have rejected each such G', and so the move from G to G' cannot be a winning move. Therefore G is a winning position. □

4.2 Lexicodes with Nim-like properties

Definition 4.3. Let $dnsum(k)$ denote the nim-sum of each of the digits (the digital nim-sum) of k.

Example. $dnsum(324215) = 3 \oplus 2 \oplus 4 \oplus 2 \oplus 1 \oplus 5 = 3$.

Theorem 4.2. *If B is of the form 2^a, then the lexicode defined by any family of turning sets is closed under component-wise nim-addition.*

I outline the approach and present the intuition for the proof of this theorem below. For further details and the rigorous proof, the reader should see [4].

1. Verify that if the base $B = 2$ in theorem 4.1, then lexicode produced is actually linear. For example, take the codewords 11001 and 00111. The nim-sum of these codewords is $11001 \oplus 00111 = (5 \oplus 0) \oplus (4 \oplus 0) \oplus (0 \oplus 3) \oplus (0 \oplus 2) \oplus (1 \oplus 1)$. Using the associativity of the nim-sum operation, we rearrange to obtain $(5 \oplus 4 \oplus 1) \oplus (3 \oplus 2 \oplus 1)$. This is just the nim-sum of the digital nim-sums of the codewords. However, since the codewords are $\mathcal{P}$ positions, their digital nim-sums are 0. Thus, $11001 \oplus 00111 = 0$ i.e. the nim-sum of these two codewords is also a codeword.

2. Now, observe that any word in base $B = 2^a$ can be converted to a word in base $B = 2$ without changing many of its properties. For instance, in $(68)_8 = (01101000)_2$. Using this fact, we may expand our code in base $B = 2^a$ to a code in base $B = 2$, from which we can repeatedly apply the result from step 1 to obtain the theorem.

Theorem 4.3. *If B is of the form 2^{2^a}, then the lexicode defined by any family of turning sets is closed under component-wise nim-multiplication by numbers α in the range $0 \leq \alpha < B$. In other words, the lexicode is a linear code over the field $GF(2^{2^a})$.*

I redirect the reader to [4] for the proof of this theorem. Unsurprisingly, this result builds on Theorem 4.2.

4.3 Interpreting Hamming codes as lexicodes

Using these two key results, we can precisely understand the Hamming codes in terms of the more general lexicodes.

Theorem 4.4. *The Hamming codes are the lexicodes with base* $B = 2^{2^a}$*, Hamming distance* $d = 3$ *and length*

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

The lexicodes with other lengths are the shortened and the extended Hamming codes.

The proof of this theorem requires a deeper understanding of the Hamming codes and their constructions. It is thereby outside the scope of this paper.

Remark. *The Hamming code example we looked at in Section 1 was in fact, a binary lexicode of length 7 and Hamming distance 3.*

4.4 The zero-sum codes

So far, we've represented Nim in a binary fashion to obtain the Hamming code. However, we can represent it in other ways too. In particular, we may represent the size of each Nim pile in the game as a digit in a string, the base of which is the size of largest Nim pile. Thus, the game $*5 + *7 + *2 + *1 + *2 + *3$ can just be written as 572123, where the ordering of these digits doesn't really matter. Since the value of a Nim game is the nim-sum of all its pile sizes, this game is a $\mathcal{P}$ position if and only if the $dnsum(572123) = 0$. This gives rise to the zero-sum code:

Definition 4.4. The *zero-sum code* of length n is the set of all strings of length n whose digital nim-sum $= 0$.

Theorem 4.5. *For any base* B*, length* n *and hamming distance* $d = 2$*, the lexicode beginning with the zero word* $000\ldots0$ *is the zero-sum code.*

Proof. We shall prove this using induction on the codewords. Clearly, this holds for the base case: $dnsum(000\ldots0) = 0$. Now assume that the following induction hypothesis holds: the lexicode contains a number $m < k$ if and only if $dnsum(m) = 0$. Then the 2 claims below follow:

> **Claim 1.** *If* $dnsum(k) = 0$*, then it will be selected.*
>
> *Proof.* Let s be an arbitrary word that has already been selected. Obviously, $k \neq s$. Since $dnsum(k) = dnsum(s) = 0$, $dnsum(k - s) = 0$. $(k - s)$ therefore has been selected too. Since $k \neq s$ obviously, $(k - s) \neq 000\ldots0$. Thus, the Hamming distance $d = 2$ condition yields that $k - s$ has at least 2 non-zero digits. Thus, k and s differ in at least 2 positions. It follows that the lexicode will select k. ■
>
> **Claim 2.** *If* $dnsum(k) \neq 0$*, then it will not be selected.*
>
> *Proof.* Let k' denote the word where the largest digit x of k has been replaced with $x \oplus dnsum(k)$. By construction, $dnsum(k') = 0$. Also by construction, the hamming distance of k and k' (a selected word) $= 1$. Thus, k will not be selected. ■

Therefore, by mathematical induction, this lexicode contains all numbers with digital nim-sum $= 0$, and only those numbers. Thus, this lexicode is the same as the zero-sum code. □

Remark. *This theorem could be proven by interpreting zero-sum codes as Nim games too. However, this threatens a loss of generality, due to which I have avoided Nim-game route.*

5 Perfect codes[3][5]

So far, we have focused extensively on the error-correcting properties of codes. However, the length of the codes are significant too: they determine how effectively a code utilizes and interprets the set of all possible words. Hence, I explore these in this section.

Definition 5.1. An e-error-correcting code of length n is called a *perfect code* if, for every string of length n, there is exactly one codeword such that the string and the codeword differ in at most e positions.

Due to their significance in maximising the number of codewords, perfect codes, their constructions and their properties have been studied extensively. I direct the reader to the numerous papers by Faina Solov'eva for more detail.

Below are some particularly interesting properties of perfect codes. In this section, the shorthand # shall denote 'the number of'.

Theorem 5.1. (Solov'eva, [6]) *For every codeword x in a perfect code, $x \oplus 111...1$ is also a codeword.*

Theorem 5.2. (Simplified version of the Sphere-packing bound [1]) *For an e-error-correcting code of length n and base B,*

$$\#codewords \leq \frac{B^n}{\sum_{i=0}^{e} \binom{n}{i}}$$

with equality only when $\#unassignedMessages = 0$ i.e. when the code is a perfect code.

Proof. We first make a simple observation:

$$\#messages = (\#codewords) \cdot (\#messagesPerCodeword) + (\#unassignedMessages)$$

This can be rearranged to obtain

$$\#codewords = \frac{\#messages - \#unassignedMessages}{\#messagesPerCodeword}$$

The $\#messages$ of length n is B^n since there are B choices for each of the n digit positions. The $\#messagesPerCodeword = \sum_{i=0}^{e} \binom{n}{i}$ since any message of this type is just the codeword altered in between 0 and e positions. We may eliminate the $\#unassignedMessages$ term by replacing the $=$ with a $\leq$ sign. The theorem follows from here by making the relevant substitutions. □

Corollary. *Since $\#codewords \in \mathbb{N}$, a code is a perfect code if and only if $\sum_{i=0}^{e} \binom{n}{i}$ divides B^n.*

This corollary turns out to be particularly useful in finding the parameters B and n for a lexicode that would result in a perfect code.

Example. The Hamming Code of length 7 (and base 2) introduced earlier is a perfect code. This is because $\binom{7}{0} + \binom{7}{1} = 1 + 7 = 8 = 2^3$ which divides 2^7. We proved a much more general case of this in theorem 4.4

Example. The binary Golay Code of length 23 is also a perfect code. This is because $\binom{23}{0} + \binom{23}{1} + \binom{23}{3} + \binom{23}{4} = 2^{11}$ which divides 2^{23}.

Theorem 5.3. *(Pless, [7]) Every perfect 2-error-correcting code of word length 11 over GF(3) is equivalent to the ternary Golay code. Every perfect 3-error-correcting code of word length 23 over GF(2) is equivalent to the binary Golay code.*

I redirect the reader to [7] for the proof. The author here made the assumption that these codes are linear. However, as noted in Comment 9.1 of [3], subsequent work has proven this result true even without this assumption.

. . . . . .

Thus far, it may appear that combinatorial constructions are all about games. However, this is not true. Clever constructions like the one below are also very potent and sometimes yield profound results.

Definition 5.2. The term *perfect 1-code* denotes a perfect binary code that corrects 1 error.

Given a perfect 1-code of length n we construct a perfect 1-code of length $2n+1$ containing the given code as a 'subcode'. The construction involves first relating the extended versions of these codes and then eliminating a coordinate to yield the perfect 1-codes.

Let $\{C_i\} = \{C_0, C_1, ..., C_n\}$ be the set of perfect 1-codes of length $n = 2^m - 1$ and hamming distance 3 that completely partitions the set of all possible messages. Define $\{B_i\} = \{B_0, B_1, ..., B_n\}$ similarly. An example of such a partitioning is when C_{i+1} is just the translation of C_i by a unit vector.

Definition 5.3. For any code C_i, let C_i^* denote its extended code constructed by adding an overall parity bit.

Let E^* be an extended code of length $2n + 2$ constructed such that a pair of codewords $(x, y) \in E^*$ if and only if $x \in C_i^*$ and $y \in B_j^*$ where the sequence $\{j\}$ is some permutation of the sequence $\{i\} = \{0, 1, 2, \ldots, n\}$.

Theorem 5.4. *(Phelps, [8]) E^* is an extended perfect 1-code of length $2n+2$. Consequently, E is a perfect 1-code of length $2n + 1$.*

I omit the long and complex proof of this theorem. Instead, the reader would find it in [8].

6 Conclusion

The exploration in this paper began with the specific case of Hamming codes, and the game of Nim. Then, in successive stages of generalisation, we found lexicodes to be deeply connected to heap games. Finally, we explored the properties of these lexicodes and their perfect versions, only to once again encounter combinatorial constructions. I hope this paper has enabled the reader to understand and appreciate the underlying combinatorics in the theory of error-correcting codes.

7 Beyond

The paper began by explaining how error-correcting codes had important applications in data transmission and storage. However, the tools developed here are far more powerful than just that. These codes and their combinatorial interpretations are key to solving many connected problems – problems which resemble the structure and properties of the various codes. I leave the reader to ponder over two partially-open problems taken from [3]:

1. **The football cup predictions:** There is a betting competition where we must predict the outcomes of n football matches (win, lose or draw). You are aiming to get one of the top 2 prizes, which are awarded for correctly predicting the outcomes of all n or $n - 1$ games respectfully. What is the most efficient way of making a number of forecasts such that you are guaranteed to win one of the prizes?

2. **Counterfeit coin puzzle:** We are given a set of n coins, out of which at most 1 is a counterfeit coin. Counterfeit coins have a weight different to normal coins. To help decipher which coin is counterfeit, we have a weighing scale that compares any two weights and tell us whether they are equal, or which is greater (if either is). What weighing program accurately identifies the counterfeit coin (if any), with the least number of comparisons?

   So far, this is standard. There is, however, an additional condition: the set of comparisons (the weighing program) must be completely predetermined i.e. the outcomes of the comparisons may not affect future comparisons.

References

[1] F. J. Macwilliams and N. Sloane. *The theory of error correcting codes*, volume 16. North-Holland Pub. C, 3rd edition, 1977.

[2] Simon Rubinstein-Salzedo. Combinatorial game theory. Chapters 3 and 4 from this book.

[3] J. H. Van Lint. A survey of perfect codes. *The Rocky Mountain Journal of Mathematics*, 5(2):199–224, 1975.

[4] J. Conway and N. Sloane. Lexicographic codes: Error-correcting codes from game theory. *IEEE Transactions on Information Theory*, 32(3):337–348, 1986.

[5] H. S. Shapiro and D. L. Slotnick. On the mathematical theory of error-correcting codes. *IBM Journal of Research and Development*, 3(1):25–34, 1959.

[6] F.I. Solov'eva. Perfect binary codes: Bounds and properties. *Discrete Mathematics*, 213(1-3):283–290, 2000.

[7] Vera Pless. On the uniqueness of the golay codes. *Journal of Combinatorial Theory*, 5(3):215–228, 1968.

[8] K. T. Phelps. A combinatorial construction of perfect codes. *Algebraic Discrete Methods*, 4(3):398–403, 1983.