The Wonderland of Combinatorial Games and Algebraic Codes

Jan van Zanten*

In the context of artificial intelligence combinatorial games play a major role, since many of these games have been analyzed to a great extent. However, they also have connections with various mathematical disciplines, and they provide the mathematical community with quite a few challenging open problems. In this paper we skim the surface of the intersection of impartial games and algebraic codes. In particular, we point out the striking intimate relationship which turns out to exist between heap games, lexicographic order and linear algebraic codes.

1 Lexicodes

First we introduce the notion of *lexicode* by a simple example. Consider the following complete list of binary words of length 5 which are ordered in standard lexicographic way.

00000 (0)	01000 (4)	10000 (5)	11000 (1)
00001 (1)	01001 (5)	10001 (4)	11001 (0)
00010 (2)	01010 (6)	10010 (7)	11010 (3)
00011 (3)	01011 (7)	10011 (6)	11011 (2)
00100 (3)	01100 (7)	10100 (6)	11100 (2)
00101 (2)	01101 (6)	10101 (7)	11101 (3)
00110 (1)	01110 (5)	10110 (4)	11110 (0)
00111 (0)	01111 (4)	10111 (5)	11111 (1)

When searching this list from top to bottom, every time selecting a word if its *Hamming distance* to all previously selected words is at least 3, we end up with the following four words: 00000, 00111, 11001, 11110.

One can immediately verify that this set is closed under addition, where addition means the bitwise exclusive-or-operation (XOR). Hence, the four words, considered as vectors over the field $GF(2)$, constitute a *linear* space (vector space) of dimension 2. Since all mutual distances between the vectors are at least 3, we may speak of a linear $[5, 2, 3]$-code. This feature of linearity does not occur by accident.

In general, if one orders the binary vectors of length n lexicographically, i.e. according to increasing values when interpreted as binary numbers, and if one

* Professor Jan van Zanten holds the named chair: *Fundamentals of Knowledge Technology from the Perspective of Theoretical Computer Science* at MICC-IKAT on behalf of the Prof. dr. R. Timman Foundation. Email: `zanten@micc.unimaas.nl`.

searches this list, starting with the zero vector, from top to bottom, selecting the next vector on the list if and only if its Hamming distance to each previously chosen vector is at least d, then one obtains a code with minimum distance d, which surprisingly turns out to be linear.

Codes constructed in this way are called lexicodes. So, the result is an $[n, k, d]$-code for some value k. The reader who likes to convince herself of this remarkable phenomenon — what has lexicographic order to do with linearity? — should try out this algorithm on the lexicographically ordered list of binary words of length 7. If she works carefully, she will find a set of 16 words (we do not distinguish between the terms word and vector and use them at will) which together form a $[7, 4, 3]$-code, the famous Hamming code of length 7, which is the first member of the family of binary Hamming codes with parameters $[2^m - 1, 2^m - 1 - m, 3]$.

The above property was detected and proved in a game-theoretic setting (impartial game theory) by Conway and Sloane in [3]. Later it was generalized in many ways by several authors. E.g., Brualdi and Pless in [2] showed that the theorem not only is valid for the "normal" lexicographic order, i.e., with respect to the standard basis in the vector space $GF(2)^n$, but also for lexicographic order with respect to an arbitrary basis. The author of this contribution generalized it further in [7] for a wide range of selection criteria, and proved everything in a purely (linear) algebraic way. Moreover, he proved that if one takes a special type of selection criteria, satisfying the so-called multiplicative property, the linearity also shows up in vector spaces and codes over fields $GF(2^{2^m}), m \geq 0$, occasionally called *Fermat fields.* Conway and Sloane already had detected this for the codes treated in [3]. Linearity of codes over this kind of fields is even more remarkable than just over $GF(2)$, since now scalars other than 0 and 1 play a role!

2 The Game of Nim

This game is known to a wide range of people for being played in pubs and bars, and even to a more sophisticated public for its featuring in an avant-garde film somewhere in the mid-sixties of the previous century, called "l Année dernière in Marienbad" (Nouvelle Vague). It is played with several heaps of matches, or beans or whatever small objects available. A move is to select a heap, and to remove any number of matches from it, possibly the whole heap. Any *position* in Nim is therefore the sum of several one-heap Nim-games. Two players take turns to make a move and, as usual, the player who is not able to make a move — which is the case when all matches are removed — loses. Nim is a special member of the family of heap games, which itself is a subset of the class of *impartial games.* The term impartial implies that the set of options for both players is the same, regardless whose turn it is to move. The solution of the game of Nim is rather easy. Let $n_1, n_2, \ldots, n_r$ be the sizes of the heaps. Write these numbers in binary and take the XOR-sum, which for the sake of this application is also called *Nim-sum.* If you are in a situation where you can leave your opponent a position with a corresponding Nim-sum equal to zero, you are in a winning position. If you play carefully, you can always win, i.e. there is a winning strategy for you. Your opponent can only produce a position with a non-zero sum, whatever move he makes. Next you can restore a zero-sum position by removing an appropriate number of matches. The proof is almost trivial when one realizes that the binary representation of the Nim-sum can be altered into any different binary representation of the same length by an appropriate legal move. This goes on until you leave your opponent the ultimate zero-sum position when there is not a single match left. If you yourself have to make a move while the Nim-sum is zero, you are in a non-winning position, and you only can pray that your opponent is going to make a mistake.

To formalize this procedure, we introduce the *g-value* of a position P, by assigning to P the integer $g(P)$ the binary representation of which is equal to the above Nim-sum corresponding to P. So, if $g(P) = 0$ you are in a winning position and if $g(P) \neq 0$ you are not. Furthermore, it is obvious that $g(P)$ is the Nim-sum

$$g(P) = g(P_1) \oplus g(P_2) \oplus \cdots \oplus g(P_r), \tag{1}$$

where $P_1, P_2, \ldots, P_r$ are the single-heaps (sometimes called *atomic positions*) of P.

This may seem an unnecessarily complicated way to describe a rather simple rule, but by doing so, we imbed the game of Nim into the general theory of heap games due to Sprague and Grundy [1, 3, 4]. A second tool in this context is the *mex*-operation (minimal excluded) defined on the family of sets of natural numbers in the following way: *mex* X is the least non-negative integer not in X.

If one defines for single-heap games (positions) P_i of size i that $g(i) \equiv g(P_i) := i$ for all $i \in \mathbb{N}$, we have the following theorem for all positions in the game of Nim.

Theorem 1. *If P is some position other than the empty heap, then the g-value of P is recursively determined by $g(0) = 0$ and the relation $g(P) = mex\{g(Q), g(R), \ldots\}$, where $Q, R, \ldots$ are the positions which originate from P by one legal move.*

One can even omit the initial condition $g(0) = 0$, when one realizes that $mex(\emptyset) = 0$, as a consequence of the definition of the *mex*-operation. For a formal proof of the above theorem, we refer to [3, 4].

3 Nim and Minimum-Distance-3 Codes

Next we shall discuss the relationship which exists between the game of Nim and other heap games on the one hand and lexicodes on the other. First we derive a slightly different rule to solve Nim. Let the position P contain m_1 heaps of size 1, m_2 heaps of size 2, …, m_n heaps of size n. Then we have from the previous section that

$$g(P) = (g(1) \oplus \cdots \oplus g(1)) \oplus (g(2) \oplus \cdots \oplus g(2)) \oplus \cdots \oplus (g(n) \oplus \cdots \oplus g(n)), \tag{2}$$

where the Nim-summation is over m_1 terms $g(1)$, m_2 terms $g(2)$, ... and m_n terms $g(n)$. So, it only matters whether the integers m_i are even or odd. Therefore, we introduce coefficients $c_i \in GF(2)$, such that $c_i = 0$ if m_i is even and $c_i = 1$ if m_i is odd, for $1 \leq i \leq n$. We now write 2 as

$$g(P) = c_1 g(1) \oplus c_2 g(2) \oplus \cdots \oplus c_n g(n), \tag{3}$$

or, equivalently, as vector equality

$$\mathbf{g}(P) = c_1 \mathbf{g}(1) + c_2 \mathbf{g}(2) + \cdots + c_n \mathbf{g}(n), \tag{4}$$

where the g-values are written in binary and are considered as vectors in $GF(2)^m$ with $m := [^2 log\ n]$. Vice versa, one can ask for those vectors $(c_1, c_2, \ldots, c_n) \in GF(2)^n$, which correspond to winning positions. To answer that question one has to solve the set of linear equations

$$H\mathbf{c} = \mathbf{0} \tag{5}$$

where H is an $m \times n$ matrix over $GF(2)$ such that its n columns are the binary representations of the integers $1, 2, \ldots, n$ We remind the reader that to each solution $\mathbf{c}$ of (5), there corresponds an infinite set of winning positions P, since the c_i-value only stands for the parity of the number of heaps of size i.

Now, suppose some $\mathbf{c}$ represents a winning position P. Then we have $g(P) = 0$. A next move will change $\mathbf{c}$ into some $\mathbf{c}'$ which differs from $\mathbf{c}$ in either one position, if a complete heap was removed, or in two positions, if not all matches of some heap were taken. As a consequence, the Hamming distance between $\mathbf{c}$ and $\mathbf{c}'$ is equal to 1 or 2.

We now search the lexicographically ordered list of binary vectors (words) of length n, starting with $\mathbf{0}$, and every time selecting the next vector at Hamming distance at least 3 from all previously selected vectors. Suppose we find in this way the set of vectors $C := \{\mathbf{c}_1 = \mathbf{0}, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$. Each of the remaining vectors in the list has a Hamming distance 1 or 2 to at least one of the vectors of C. We shall prove now that C is the *complete set* of solutions of (5) by showing that all vectors of C have a g-value 0, while all vectors not in C have a g-value larger than 0. We consider some position P represented by vector $\mathbf{c}$. Since the g-values do not depend on the absolute values of the m_i, but only on their parities, we may assume that the integers m_i are equal to 1 or to 0. Because of this assumption, any legal move changes $\mathbf{c}$ into a vector $\mathbf{c}'$ which is less than $\mathbf{c}$ and hence occurs higher in the list. Moreover, we have $d(\mathbf{c}, \mathbf{c}')$ is equal to 1 or 2. Now, the zero vector $\mathbf{c}_1$ certainly represents a winning position and therefore $g(\mathbf{c}_1) = 0$. The next vectors in the list which are not selected as elements of C since they are at distance less than 3 from $\mathbf{c}_1$, all have the property that $\mathbf{c}_1$ originates from them by one legal move. So, according to (1), all these vectors have a g-value larger than 0. The vector $\mathbf{c}_2$ is at distance 3 from $\mathbf{c}_1$, and hence $\mathbf{c}_1$ does not stem from $\mathbf{c}_2$ by one legal move. Again by (1), it follows that $g(\mathbf{c}_2) = 0$. Continuing in this way, or by applying induction, we arrive at the result that $g(\mathbf{c}) = 0$ if and only if $\mathbf{c} \in C$. So, C contains all vectors satisfying (5). Since the set of all solutions of (5) forms a vector space (the *kernel* or *nullspace* of H), the resulting code C is linear, and so we have an $[n, k, 3]$-code for some value k. We mark the vectors of C by 0, their common g-value. Similarly, we can mark the other vectors of the list by their respective g-values. The set of vectors marked by i is precisely the coset $C + \mathbf{v}_i$, where $\mathbf{v}_i$ is some vector with g-value i, and constitutes the complete solution of the equation $H\mathbf{c} = \mathbf{i}$.

Implicit in the above arguments is that a winning position (represented by a vector) $\mathbf{c}$, with $g(\mathbf{c}) = 0$, *is always changed* into a non-winning position $\mathbf{c}' < \mathbf{c}$, with $g(\mathbf{c}') \neq 0$, by any legal move, and that $\mathbf{c}'$ can always be changed into a winning position $\mathbf{c}'' < \mathbf{c}'$, with $g(\mathbf{c}'') = 0$. This is due to the search algorithm which rejected $\mathbf{c}'$ as element of C, because there was at least one vector, i.e. $\mathbf{c}'' \in C$, at distance less than 3 from $\mathbf{c}'$.

To illustrate the linear algebraic aspects of the game of Nim, we take $n = 7$, i.e. the maximal size of the heaps is 7. The set of equations (5) now has the form

$$\begin{pmatrix} 1\,1\,1\,1\,0\,0\,0 \\ 1\,1\,0\,0\,1\,1\,0 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix} \begin{pmatrix} c_7 \\ c_6 \\ \vdots \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

The solution of this set constitutes the binary $[7, 4, 3]$-Hamming code. Apart from the zero vector and the all-one vector, this code contains seven vectors of weight 3 and seven vectors of weight 4. All these vectors correspond to winning positions. E.g., the vector $(c_7, c_6, \ldots, c_1) = (0, 1, 0, 1, 0, 1, 0)$ stands for winning positions with an odd number of 2-heaps, an odd number of 4-heaps and an odd number of 6-heaps, while the numbers of heaps of size 1, 3, 5 and 7 are all even. Leaving out the first two columns of the matrix provides us with the parity check matrix which defines the $[5, 2, 3]$-code of our first example. In general, for $n = 2^m - 1$, the constructed lexicodes are the binary Hamming codes with parameters $[2^m - 1, 2^m - 1 - m, 3]$. For other values of n, one obtains shortened Hamming codes.

4 Lexicographic Order and Linearity

In order to deal with other heap games and their role in deriving lexicodes, we present a more general version of our search algorithm. Let P denote some property or criterion such that all vectors of the lexicographically ordered list of words of length n can be tested as to whether they satisfy P or not. We formulate the following recursive rule.

> *Choose the next vector* $\mathbf{x}$ *of the list whose distance vector* $\mathbf{x} + \mathbf{y}$ *with respect to each previously chosen vector* $\mathbf{y}$ *satisfies* P.

The resulting set of vectors is called $C(\mathrm{P})$. In [7] we proved the following.

Theorem 2. *The lexicode* $C(\mathrm{P})$ *is linear for any selection criterion* P.

Completely similar to the case of Nim, one can assign g-values (Nim-values) to the vectors of the list which is searched by the algorithm with P as selection criterion. Again due to the nature of the search, one has that a vector $\mathbf{c}$ ($\neq \mathbf{0}$) with $g(\mathbf{c}) = 0$, is always changed by a legal move into a vector $\mathbf{c}' < \mathbf{c}$ such that $g(\mathbf{c}') \neq 0$, and that $\mathbf{c}'$ can always be changed into a vector $\mathbf{c}'' < \mathbf{c}'$, with $g(\mathbf{c}'') = 0$. So, in a corresponding heap game the winning positions are characterized by $g(\mathbf{c}) = 0$, just like in Nim.

Theorem 2 generalizes a similar statement in [3]. Actually, in [7] the theorem is even proved to be true for lists which are ordered in lexicographic sense with respect to an *arbitrary* basis, thus generalizing also a result in [2]. However, we will not exploit this here, and stick to the standard bases in the relevant vector spaces. The only requirement about P is that it is fixed, and hence that it should not be changed dynamically during the course of the algorithm. We shall use the notation P[$\mathbf{v}$] which stands for "$\mathbf{v}$ satisfies P". So, in the above algorithm the criterion is "P[$\mathbf{x}+\mathbf{y}$] for all $\mathbf{y}$ already chosen". In this terminology, we can formulate the criterion of the previous section as $\|\mathbf{x}+\mathbf{y}\| \geq 3$, where the left hand side denotes the *Hamming weight* of $\mathbf{x}+\mathbf{y}$. It will be obvious that one could impose any other requirement to the *weight spectrum* of $\mathbf{x}+\mathbf{y}$. To this end we introduce the notion of *turning set* (cf. [3, 7]), being a set of index values. As selection criterion P[$\mathbf{v}$] we take the requirement that the set of indices of the non-vanishing components of $\mathbf{v}$ does not belong to a prescribed family T of turning sets. Somewhat sloppy, we write $\mathbf{v} \notin T$. In the case of Nim, the family T_N consists of all sets with less than three indices. In the next section we shall study in more detail another particular family of turning sets. However, as we shall see below, one can also require criteria P which cannot be expressed in terms of turning sets.

First we emphasize that, whatever selection criterion we apply, the zero vector will always be selected. This is because the set of previously chosen vectors $\mathbf{y}$ is empty in the case $\mathbf{x} = \mathbf{0}$. As a consequence, all vectors of C satisfy property P, except possibly the zero vector. As an example we consider again the list of words of length 5 presented in Section 1. If we take for P[$\mathbf{v}$] the requirement that the weight of $\mathbf{v}$ is at least 3, then all vectors in C satisfy that property, except $\mathbf{0}$ which has weight 0.

We now present a few examples where P cannot be phrased in terms of turning sets or weight spectra. First, P[$\mathbf{v}$] will stand for the property that $\mathbf{v}$ contains the pattern 11. Applying the algorithm again to the list of Section 1, provides us with the set

$$C = \{00000, 00011, 01100, 01111\}.$$

This set indeed is linear and all words contain 11, except the zero word.

Another possibility is that P[$\mathbf{v}$] stands for the requirement ($v_1 + v_2 + v_5 = 0 \wedge v_3 + v_4 = 0$). Now we find the set

$$C = \{00000, 00011, 01100, 01111, 10001, 10010, 11101, 11110\}.$$

All vectors, including the zero vector now satisfy P. We remark that the linearity in this case is no surprise at all, since P only says that $\mathbf{v}$ has to satisfy a homogeneous set of linear equations. Whatever order the list of binary words of length 5 may have, we would always wind up with the complete solution of such a set, which — as almost everybody knows — is linear.

A bigger surprise is obtained by a search with $\mathrm{P}[v] \equiv (v_1 + v_2 + v_5 = 0 \vee v_3 + v_4 = 0)$, which results in the *linear* code

$$\begin{aligned} \mathrm{C} = \{&00000, 00001, 00010, 00011, 01100, 01101, 01110, 01111,\\ &10000, 10001, 10010, 10011, 11100, 11101, 11110, 11111\}. \end{aligned}$$

Remember that the union of two non-empty linear spaces is not linear again!

Finally, we mention the special example of $\mathrm{P}[\mathbf{v}] \equiv (\|\mathbf{v}\|$ is even), providing us with the full linear *even weight* code C, consisting of all words of even weight. Since $\|\mathbf{x}+\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + 2(\mathbf{x},\mathbf{y}) + \|\mathbf{y}\|^2$, where $(\cdot\,,\cdot)$ denotes the inner (dot-) product of two vectors, it follows that all vectors are orthogonal to each other and to themselves. In other words, C is a *self-orthogonal*, though trivial, $[n, n/2, 2]$-code. In fact, we don't need our algorithm to find its words. Less trivial self-orthogonal (or even *self-dual*) codes are obtained by taking $P[\mathbf{v}] \equiv ((\|\mathbf{v}\| \text{ is even}) \wedge (\|\mathbf{v}\| \geq d))$. For concrete examples, we refer to the literature mentioned in [7].

5 Grundy's Game

In this section we consider the following family of turning sets.

$$T_G := \{\{1,2,3\},\{1,3,4\},\ldots,\{2,n-2,n\},\{1,n-1,n\}\} \tag{6}$$

which contains all subsets of three indices such that the largest integer is the sum of the other two. Remember that we index codewords by $1, 2, \ldots, n$ from right to left, because of our lexicographic order. The search criterion is now that $\mathbf{x} + \mathbf{y} \notin T_G$. For $n = 4$ we obtain the following list.

0000 (0)	1000 (0)
0001 (0)	1001 (0)
0010 (0)	1010 (0)
0011 (0)	1011 (0)
0100 (1)	1100 (1)
0101 (1)	1101 (1)
0110 (1)	1110 (1)
0111 (1)	1111 (1)

There really exists a well-known heap game corresponding to the family of turning sets T_G, known as *Grundy's game*. It is almost as simple as the game of Nim, and therefore very appropriate to be played in bars. Its only rule is: split some heap into two *unequal* heaps. It follows immediately that $g(1) = g(2) = 0$, since one heap with one or two matches cannot be split adequately. A position with one heap containing three matches is a non-winning position, since that heap can be split into two heaps of size 1 and 2. From Theorem 1 it follows that $g(3) = 1$. We emphasize that the rules for computing g-values are the same as those for Nim or for any other heap-game. The only difference which plays a role when applying Theorem 1, is that "the positions reached by one legal move" depend on the particular game. Applying the rules of Grundy's game provides us respectively with $g(0) = 0$, $g(1) = 0$, $g(2) = 0$, $g(2,1) = 0 \oplus 0 = 0$, $g(3) = mex\{g(2,1)\} = mex\{0\} = 1$, $g(3,1) = 1 \oplus 0 = 1$, $g(3,2) = 1 \oplus 0 = 1$, $g(4) = mex\{g(3,1)\} = mex\{1\} = 0$, $g(4,1) = 0 \oplus 0 = 0$, $g(5) = mex\{g(4,1), g(3,2)\} = mex\{0,1\} = 2$, $g(4,2) = 0 \oplus 0 = 0$, $g(5,1) = 0 \oplus 0 = 0$, $g(6) = mex\{g(4,2), g(5,1)\} = mex\{0,0\} = 1, \ldots$

By continuing in this way one can derive the values for all one-heap positions (cf. [3]):

$$\begin{array}{rcccccccccccccccccc} i\colon & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \ldots \\ g(i)\colon & 0 & 0 & 0 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 3 & 2 & 1 & \ldots \end{array}$$

Because of the values of $g(1)$, $g(2)$, $g(3)$ and $g(4)$ the codewords of the aforementioned code all satisfy the set of equations:

$$\begin{pmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,1\,0\,0 \end{pmatrix} \begin{pmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

6 Other Impartial Games

The game of Nim sometimes shows up in disguised form. Consider a row of coins of length n. Number the coins by $1, 2, \ldots, n$ from right to left. A legal move is to turn a head to a tail, with the additional option of turning at most one other coin to the right of it, either from head to tail or from tail to head. The game has ended as soon as all coins show tails, i.e. the player who is unable to move loses. Identifying head with one and tail with zero, it will be clear that this game is Nim in disguise. or better, it is equivalent to a reduced form of Nim where there is at most one heap of size i, for all relevant values of i. When it is played in heap version, one should remove pairs of heaps of the same size every time such a pair is produced (cf. also the derivation of the Nim-values in Section 3). Surprisingly, this coin game is called *Turning turtles*, since originally it was supposed to be played with turtles instead of coins.

A small variation is obtained when we allow a move to consist of turning up to three coins, provided the leftmost one goes from head to tail. This last condition of course guarantees again that a move produces "smaller" vectors and eventually leads to an end position. The corresponding search of the lexicographically ordered list is now to select every time the smallest word at Hamming distance at least four. The corresponding game is called *Mock turtles*, another "Alice-in-wonderland type name". The resulting lexicode is the linear *extended Hamming* code with parameters $[8, 4, 4]$, related to the binary Hamming code of length 7 by an additional *parity check*.

Some other well-known codes which turn out to be lexicodes, related to games with exotic names as well, are (cf. [1, 3, 6]):

(i) the $[17, 9, 5]$ quadratic residue code and the $[18, 9, 6]$ extended quadratic residue code with corresponding game known as *Moebius*;
(ii) the $[23, 12, 7]$ Golay code and the $[24, 12, 8]$ extended Golay code, with corresponding game called *Mogul*;
(iii) the $[27, 9, 10]$ and $[31, 12, 10]$-codes with corresponding game *Moidores*.

7 Lexicodes over Other Finite Fields

Conway and Sloane in [3] already detected and proved that a lexicode of length n over the field $GF(2^l), l > 0$ also has the property that it is closed with respect to addition, and moreover, that if l is a 2-power itself, it is closed with respect to multiplication by the scalars of the field as well. The only requirement for the first phenomenon is that the vector space $GF(2^l)^n$ is spanned by an ordered basis $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$, where the n components of each vector $\mathbf{b}_i$ are all equal to 1 or 0, a so-called *binary basis* and that the elements of $GF(2^l)$ are ordered as binary vectors of length l, in lexicographic sense. Applying our search algorithm, provides us with a lexicode which we denote by $C(\mathrm{P}; GF(2^l))$. For $l = 1$, and taking the standard basis for B, we obtain the lexicodes $C(\mathrm{P})$ of the previous sections. In [7] the following generalization is proved.

Theorem 3. *The lexicode $C(\mathrm{P}; GF(2^l))$ is closed under addition, for any binary basis of $GF(2^l)^n$ and for any selection criterion* P.

In general such a code is not linear for an arbitrary value of l, because it is not closed under scalar multiplication. Only if $l = 2^m, m \geq 0$, it turns out to be linear. In case that P stands for a property expressed in terms of turning sets, this feature was already noticed and also proved in [3] in the context of the theory of heap games, using so-called *Nim-multiplication.* The proof given in [7], which is of purely algebraic nature, holds for a larger class of properties P satisfying the requirement of *multiplicativity*

$$\mathrm{P}[\mathbf{v}] \to \mathrm{P}[\alpha\mathbf{v}], \tag{7}$$

for all non-zero elements $\alpha \in GF(2^{2^m})$. To accomplish such a proof, one first has to define a kind of *canonical order* for the elements of this field (cf. [7]). The resulting theorem can be formulated as follows.

Theorem 4. *The lexicode $C(\mathrm{P}; GF(2^{2^m}))$ is linear for any binary basis of $GF(2^{2^m})$ and for any multiplicative property* P.

In [3] it is remarked that by taking for P the criterion that the weight of $\mathbf{v}$ should be at least 3 (cf. Section 4), one obtains Hamming codes again. More precisely, if $n = B^r - 1/B - 1$, with $r = 2^{2^m}$, then the lexicodes are normal Hamming codes, while those of other lengths are shortened Hamming codes. For more examples in terms of weight spectra or turning sets, we refer to [3, 5, 6]. In order to illustrate that Theorems 3 and 4 go beyond these concepts, we present a few small examples which can be verified by hand. Let us consider the vector space $GF(4)^3$ with its vectors arranged in lexicographic order:

$$000, 001, 002, 003, 010, \ldots, 333. \tag{8}$$

The integers 0, 1, 2, 3 stand for the elements 0, 1, ω, ω^2 of $GF(2^2)$, where ω is defined by

$$\omega^2 + \omega + 1 = 0. \tag{9}$$

Since 0, 1, 2, and 3 are elements of a field which is an extension of $GF(2)$, their addition rules coincide with Nim-addition, while they multiply among each other according to the field multiplication imposed by (9). So, one has $2 * 2 = 3$, $2 * 3 = 3 * 2 = 1$ and $3 * 3 = 2$.

Next, we take as selection criterion P[**v**] the equation $v_1 + v_2 + v_3 = 0$. The resulting lexicode is the set

$$\begin{aligned} C = \{&000, 011, 022, 033, 101, 110, 123, 132 \\ &202, 213, 220, 231303, 312, 321, 330\}. \end{aligned}$$

As one can verify, this code is invariant under addition and also under multiplication. Notice that its dimension is 2 and that its minimum distance also equals 2, and so C is a linear $[3, 2, 2]_4$-code. This example illustrates Theorem 4, since P is multiplicative. If one would take for P[**v**] the requirement $v_1 + v_2 + v_3 = 1$, the resulting lexicode would be the trivial code $C = \{000, 001\}$. This code is indeed closed with respect to addition (Theorem 3), but not with respect to multiplication, since now P is not multiplicative.

Next, we take $\mathrm{P}[\mathbf{v}] \equiv (v_1 v_2 v_3 = 0)$, and apply the algorithm again to list 8 which yields $C = \{000, 001, \ldots, 033\}$ containing the first 16 words of 8. The reader should observe that a word like 100 is "blocked" e.g. by 011. Since P is clearly multiplicative, the code is linear according to Theorem 4. Applying $\mathrm{P}[\mathbf{v}] \equiv (v_1 v_2 v_3 = 1)$, gives the code $C = \{000, 111, 222, 333\}$, which is also linear, because P satisfies property 7. This time the multiplicative property is due to the equality $\alpha^3 = 1$, for all $\alpha \neq 0$ in $GF(4)$. Hence, the latter example can be generalized for all vector spaces $GF(4)^n$ with $n \equiv 0 \pmod 3$. For $GF(4)^2$ and $\mathrm{P}[\mathbf{v}] \equiv (v_1 v_2 = 1)$, the lexicode appears to be $C = \{00, 11, 23, 32\}$, which is closed again under addition, but obviously not under multiplication.

8 Grundy Functions

Each heap game (coin-turning game) has its own Grundy function $g : \mathbb{N} \rightarrow \mathbb{N}$, where $g(i)$ is defined as the Nim-value of the single-heap game (position) of size i in that particular game. In principle one can compute the value $g(i)$ for any $i \in \mathbb{N}$, by applying Theorem 1 a number of times. One could say that the integer function $g : \mathbb{N} \rightarrow \mathbb{N}$ is the restriction to singletons of the function $g : \mathcal{P}(\mathbb{N}) \rightarrow \mathbb{N}$ which is defined on the family of subsets of $\mathbb{N}$. In Section 2 we saw that for the game of Nim itself, the g-function is completely trivial: $g(i) = i, \forall\, i \in \mathbb{N}$.

The g-function for Mock turtle can also be described in a simple way: $g(i) = 2(i-1)$ if the binary representation of $i-1$ contains an odd number of ones, whereas $g(i) = 2i-1$ otherwise. However, in case of Grundy's game the behavior of the Grundy function is still not completely understood. Though the first 10^7 values have been calculated (by Mike Guy, cf. [3, 5]), it is not even known if it is a periodic function or not. Among these known values, the integers

$$0, 1, 6, 7, 10, 11, 12, 13, 18, 19, 20, 21, 24, \ldots$$

occur quite rarely, and are called for that reason *rare values* (cf. [5]). Their binary representation contains an even number of ones when ignoring the last bit. Because of the latter condition, they occur in pairs of consecutive integers. It also follows immediately that they form a linear subspace of dimension $n-1$ of $GF(2)^n$ when restricting ourselves to binary representations of length n. This subspace sometimes is called *sparse space*, and its only coset has the name *common coset*. Somewhat more precisely, in Grundy's game, only 1273 rare values occurred among the calculated first ten million, while $108 (= (1101100)_2)$ turned out to be the only one in the range $36184 \leq i < 10^7$. If the rare values really do not occur anymore as Nim-values, then the g-function for Grundy's game will ultimately be periodic, be it that its period will be very large (cf. [5]).

As for lexicodes over $GF(2^{2^m})$, there is a complete theory for $m = 1$ and minimum distance $d = 3$ (Hamming codes over $GF(4)$). The structure of the next family of lexicodes over $GF(4)$ with minimum distance 4 has been developed by Wilson (cf. [3] and the references mentioned there), who also computed the first few corresponding g-values, called *Wilson numbers* in [3]. In general however, one can say that the theory of Grundy functions is poorly developed and seems to be the home of quite a lot of challenging problems, like so many other areas in the field of combinatorial games.

References

1. E.R. Berlekamp, J.H. Conway and R.K.Guy, Winning ways (two volumes), Academic Press, London, 1982.
2. R.A. Brualdi and V.S. Pless, Greedy codes, *J. Combin. Theory Ser. A*, **64** (1993) pp. 10 - 30.
3. J.H. Conway and N.J.A. Sloane, Lexicographic codes: Error-correcting codes from game theory, *IEEE Trans. Inform. Theory*, **32** (1986) pp. 337 - 348.
4. P.M. Grundy, Mathematics and Games, *Eureka* **2**, pp. 6-8; reprinted in *Eureka* **27** (1964), pp.9-11.
5. R.K. Guy, Impartial games, in: Combinatorial Games, *Proc. Sympos. Appl. Math.* **43**, (R.K. Guy, ed.), Amer. Math. Soc., Providence, R.I, 1991, pp.35 -55.
6. Vera Pless, Games and codes, in: Combinatorial Games, *Proc. Sympos. Appl. Math.* **43**, (R.K. Guy, ed.), Amer. Math. Soc., Providence, R.I, 1991, pp. 101 - 110.
7. A.J. van Zanten, Lexicographic order and linearity, *Designs, Codes and Cryptography*, **10** (1997) pp. 85 - 97.