Error-Correcting Codes Derived from Cellular Automata Games

Damla Zerya Aslan

July 14, 2024

Abstract

This paper presents a polynomial-time algorithm for computing Grundy values and deriving lexicodes from two-player cellular automata games, specifically focusing on acyclic digraphs. By leveraging structured subspaces, the algorithm constructs subgraphs, applies the Generalized Sprague-Grundy algorithm, and performs matrix transformations to compute lexicodes efficiently. This method is demonstrated with examples, ensuring polynomial-time complexity and a minimum Hamming distance for effective error correction. The study generalizes previous work on 1-regular games, providing a framework for computing winning moves and error-correcting codes. Additionally, it discusses challenges associated with higher-dimensional games and those involving loops.

1 Introduction

In the realm of combinatorial game theory, cellular automata games have traditionally been analyzed as zero-player games, such as Conway's Game of Life, or as solitaire games played on a grid or digraph G=(V, E). In these games, each cell or vertex of the graph can assume a finite number of possible states, typically restricted to the binary alphabet 0, 1. A position in the game is defined as an assignment of states to all vertices. The transition from one position to another follows a local rule: selecting a vertex u and "firing" it, meaning complementing its state together with the states of its neighbors in a predefined neighborhood N(u). The objective is to transform an initial configuration into a target configuration, such as changing all states to 0.

Previously these 0-player games have been extended to 2-player games, introducing new layers of complexity and strategic depth. In a two-player cellular automata game or a Celata game, the players alternate in selecting vertices to fire. Each player chooses a vertex u with state 1 and complements it along with its neighborhood N(u), thereby flipping the states of u and its neighbors. The game continues until one player achieves the target configuration (e.g., all vertices in state 0), winning the game, or until no more moves are possible, resulting in a draw.

In parallel to their emergence, strategies for Celata games have been explored (1) which then proved useful for constructing linear error-correcting codes. Error-correcting codes are mathematical constructs designed to detect and correct errors that occur during data transmission. Linear error-correcting codes, in particular, are a class of codes that use linear algebraic methods to encode and decode data. These codes are defined by generator matrices and parity-check matrices, which establish the relationships between the encoded data and the original information.

The process of encoding involves transforming a message vector into a codeword vector using the generator matrix, which adds redundancy to the message in a structured way. This redundancy enables the detection and correction of errors during transmission. The decoder uses the parity-check matrix to identify discrepancies between the received codeword and valid codewords, thereby locating and correcting errors. Linear codes such as Hamming codes and Reed-Solomon codes are widely used in various communication systems due to their efficiency and robustness.

The relationship between combinatorial games and linear error-correcting codes has been further explored through the concept of lexicographic codes, or lexicodes. Lexicodes are constructed using a greedy algorithm that builds the code incrementally. This algorithm adds codewords one by one, ensuring that each new codeword maintains a minimum distance d from all previously selected codewords. The efficiency of lexicodes arises from this greedy construction method, which simplifies the selection process and ensures that the resultant code meets specific distance requirements.

The scope of this paper is to explore the construction of lexicodes from the winning strategies and structures of Celata games. By analyzing the polynomial-time computation of Grundy values and leveraging structured subspaces, we aim to demonstrate how these game-theoretic concepts can be used to efficiently derive lexicodes with robust error-correcting properties

2 Preliminaries

We are concerned with games played on a digraph G = (V, E). Tokens are initially distributed on a subset of V. A move consists of selecting a token and moving it to a neighboring vertex, unoccupied or occupied. The player making the last move wins, and the opponent loses. If there is no last move, the outcome is a draw.

The corresponding game graph is a digraph G'= (V', E') where V' is the set of all collections of token distributions on G, and (u, v) $\in$ E' if there is a move from u to v. Note that for the case of a single token on v, we have G' = G.

A vertex u $\in$ V (game-position) is labeled N if the Next player (the player moving from u) has a winning move; and by P if the Previous player (who landed on u) can win. The set of all P-positions and all N-positions are denoted by P and N, respectively.

For any digraph G = (V, E), the set F(u) of followers of u $\in$ V is defined by

$$F(u) = \{u \in V : (u, v) \in E\}.$$

It can then be shown that

$$u \in P \text{ if and only if F(u)} \subseteq N,$$

$$u \in N \text{ if and only if F(u)} \cap P = \emptyset.$$

Further, if G is acyclic, then the sets P and N partition V: P $\cup$ N = V, P $\cap$ N = $\emptyset$.

In our study of Celata games, we will focus on acyclic games. Unlike cyclic games, acyclic games do not involve positions that can be revisited, preventing the formation of cycles in the game graph. This introduces specific dynamics and simplifies the analysis.

Definition 1: An acyclic Celata game of size n is represented by a digraph G = (V, E) with V = (z_1, ..., z_n), and E representing the allowed moves between states. Each vertex corresponds to a unique game state, and edges denote possible transitions that maintain the acyclic nature of the game. The game progresses through a series of moves, where each player flips a cell in the binary representation of the state, ensuring that no two adjacent cells are both 1, maintaining the acyclic property.

Remark 1:

(i) The indices are in the integer interval [1, n] making the notation u = $(u_1, \ldots, u_n)$ more convenient for this case. When u = 0, the game graph is empty, and the game ends immediately.

(ii) The vertex z_1 is not a leaf, meaning it has at least one outgoing edge. A token on z_1, for any $i \in \{1, \ldots, n\}$ can be flipped, maintaining the acyclic property of the game.

(iii) From any vertex z_1, every follower is accessible. Consequently, the Grundy numbers g(z_i) are monotonically increasing: $g(z_1) < g(z_2) < g(z_3) < \dots$

The analysis of Nim-Celata games involves understanding the structure of the game graph and the application of vector spaces over the binary field GF(2). Each game state can be represented as a binary vector, and the transitions between states can be modeled using linear algebra.

Definition 2 The numerical value of a vector u $= (u_o, \dots, u_{n-1}) \in GF(2)^n$ is $|u| := \sum_{i=0}^{n-1} = u_i 2^i$. The weight of u is w(u) $= \sum_{i=0}^{n-1} u_i$, which is the number of 1-bits in u. The parity weight of u is $w'(u) = \sum_{i=0}^{n-1} u_i$.

Definition 3 The Hamming distance between two vectors u, v over GF(2) is defined by $H(u, v) = w(u \bigoplus v)$. It is the number of positions in which u and v differ. A vector u with $w(u)$ even is said to be an evil number, and one with $w(u)$ odd is an odious number.

Definition 4 The Grundy number (or nimber) $g(x)$ of a position x in a combinatorial game is defined recursively as the minimum excludant (mex) of the Grundy numbers of positions reachable from

$$g(u) = \text{mex} g(F(u)),$$

where for any set T and any function h on T, $h(T) = \{h(t) : t \in T\}$.

Definition 5 For any terminal position x (i.e., a position with no moves available), the Grundy number is zero. Therefore we have:

$$P = \{u \in V : g(u) = 0\},$$

$$N = \{u \in V : g(u) \neq 0\},$$

$$D = \{u \in v : g(u) = 1(L), 0 \notin L\},$$

where L denotes the leaves in the game tree, representing positions that lead directly to a terminal position.

We denote the sum of the integers $a, b \in Z \geq 0$ over GF(2) by $a \oplus b$, also known as XOR, or Nim-sum. The XOR of $a_1, ..., a_m$ may also be denoted by $a_1 \oplus ... \oplus a_m = \oplus_{i=1}^m a_i$, where the apostrophe denotes XOR.

Definition 6 Let $f_1, ..., f_m$ be a finite collection of disjoint acyclic games with game graphs $G_1 = (V_1, E_1), ..., G_m = (V_m, E_m)$ The sum-game, denoted by 1+ ...+ m is a two-player game where:

(a)Each game position is a tuple $(u_1, ..., u_m)$, with $u_i \in V_i$ for all i.

(b)A move consists of selecting one component game i and transitioning from u_i to v_i, provided that $(u_i,\ v_i) \in E_i$, while all other game components remain unchanged.

(c)The sum-graph $G = G_1 + ... + G_m$ is defined by:

$$V = \{(u_1, ..., u_m) : u_i \in V_i\} \text{ for all } i$$

$$E \text{ where } (u, v) \in E \text{ if there exists some } j \text{ such that}$$

$$(u_j, v_j) \in E_j \text{ and } u_i = v_i \text{ for all } i \neq j.$$

3 Dimensionality and Sparsity of V

Definition 7 Consider the set $\{z_0, z_1, ..., z_{n-1}\}$ where z_i is a unit vector in V. Specifically, z_i has a 1 in the ith position and 0s elsewhere. This set forms a basis for the vector space V, establishing that any vertex u in V can be uniquely represented as a linear combination of these basis vectors: $u = \sum_{i=0}^{n-1} u_i z_i$,
where $u_i \in \{0, 1\}$ are coefficients in GF(2). The dimension of the vector space is thus n, the number of vertices in the game graph, which also indicates the number of basis vectors.

Definition 8 The set of followers $F(u)$ for a state u in V is defined through the game's movement rules encoded by the digraph. For each active component $u_k = 1$ (indicating the presence of a specific set), the potential new sets are indicated by:

$$F(u) = F(u) = \bigcup_{u_k=1} \bigcup_{F_q(z_k) \subseteq F(z_k)} F_q^k(u)$$

where $F_q(z_k)$ is a subset of followers $F(z_k)$, representing all possible moves from z_k within the confines of q followers.

Theorem 1 In the vector space V, the sum of any two vertices $u, v \in V$ is given by their vector addition in GF(2). This operation, defined by: $u \oplus v = (u_0 \oplus v_o, u_1 \oplus v_1, ...u_{n-1} \oplus v_{n-1})$, demonstrates the additivity of the game graph under the vector space operations.

Proof 1 Consider two arbitrary vertices $u, v \in V$ represented as $u = \sum_{i=0}^{n-1} u_i z_i$ and $v = \sum_{i=0}^{n-1} v_i z_i$. Their sum in the vector space is computed as:

$$u \oplus v = \left(\bigoplus_{i=0}^{n-1} u_i z_i \right) \oplus \left(\bigoplus_{i=0}^{n-1} v_i z_i \right) = \bigoplus_{i=0}^{n-1} (u_i \oplus v_i) z_i,$$

where $\oplus$ within the summations signifies vector addition in GF(2). This also confirms that $u \oplus v$ lies in V.

Lemma 1 The possible subsequent states $F(u)$ from any vector u_i in V can be defined as:

$$F(u) \subset \bigcup_{j=1}^{h}(F(u_j) \oplus \bigoplus_{i \neq j} u_j).$$

This inclusion shows the superposition principle where the effect of individual moves from state u^j is modified by the presence of other states.

Proof 2 Consider any $v = F(u)$. By definition v results from u by applying a valid move which can be decomposed into a move from one of the u_j affected by the context of the other vectors. Hence, v can be expressed as:

$$v = u \oplus z_k \oplus \sum_{l \in F_q(z_k)} z_l$$

for some k where $u_k = 1$ and $F_q(z_k) \subset F(z_k)$ This move reflects the game's rules that allow changing the state u by 'firing' z_k and altering its q-sized follower set $F_q(z_k)$.
For any v resulting from such transformations, it must either introduce v into $F(u)$ if $u_k = 1$, aligning with the addition in GF(2) reflecting the XOR operation of vector components. In acyclic games, once a state u transitions to v, it cannot revert back, ensuring the forward progression of game states.

Corollary 2 For two specific states u_1 and u_2 in V:

$$F(u_1 \oplus u_2) \subset (u_1 \oplus F(u_2)) \cup (F(u_1) \oplus u_2).$$

This demonstrates the distributive property of the follower function over the binary vector addition, simplifying the prediction of game developments from combined states.

Example 1 Consider a 2-player game on a digraph G(2). Let $u_1 = x_1y_2$ (meaning that $w(x_1) = w(y_2) = 1$ and all other weights are 0), $u_2 = y_1y_2$. Then:

$$u_1 \oplus u_2 = x_1y_1, F(x_1y_1) = \{0, x_1x_2\}, F^{-1}(x_1y_1) = \{y_2\}, F(u_1) = \{y_1y_2, y_1\},$$

$$F(u_2) = \{x_2y_2, x_1\}, u_1 \oplus F(u_2) = \{x_1x_1, y_2\}, F(u_1) \oplus u_2 = \{0, y_2\}.$$

his example illustrates how Corollary 1 is satisfied.

Theorem 2 Let $G = (V, E)$ be the cellular automata graph of a finite acyclic digraph. Then V_f and V_0 are linear subspaces of V. Moreover, ϕ is a homomorphism from V_f onto $GF(2)^t$ for some $t \in \mathbb{Z}_{\geq 0}$ with kernel V_0

and quotient space $V_f/V_0 = \{V_i : 0 \leq i < 2^t\}$, where $\dim(V_f) = m + t$ and $m = \dim(V_0)$.

Proof 3 Let $u, v \in V_f$. Then $u \oplus v \in V_f$ by Theorem 1, and also $0 \in V_f$. Thus V_f is a subspace of V.

Let t be the smallest nonnegative integer such that $g(u) \leq 2^t - 1$ for all $u \in V_f$. Hence, if $t \geq 1$, there is some $v \in V_f$ such that $g(v) \geq 2^{t-1}$. By the properties of homomorphisms and Theorem 1, g maps V_f onto $GF(2)^t$ with kernel V_0. Thus, by linear algebra, we have the isomorphism $GF(2)^t \cong V_f/V_0$. Therefore, $\dim(V_f) = m + t$, where $m = \dim(V_0)$.

Example 2 Consider a 2-regular game played on a digraph with vertices encoded as $z_i = 2i$ for all $i \in \mathbb{Z}_{\geq 0}$. For instance, 5 means that there are tokens on vertices z_0 and z_2 only. Using the algorithm to calculate g outlined in Definition 3, we find that $g(1) = 0$, $g(4) = 0$, $g(10) = 0$. Hence, by linearity, $V_0 = \{0, 1, 4, 10, 5, 11, 14, 15\}$. We further note that $g(2) = 1$, leading to the coset $V_1 = 2 \oplus V_0 = \{2, 3, 6, 7, 8, 9, 12, 13\}$. With $m = 3$, $t = 1$, and V_f spanned by the basis vectors $\{1, 4, 10, 2\}$, we find $\dim(V_f) = 4$. This example highlights the structured nature of V_f and confirms that each g-value is assumed in V.

Lemma 2 Let V be the n-dimensional vector space over GF(2) of the cellular automata game on $G = (V, E)$ with $\dim(V_0) = m$. There exists a homomorphism ψ mapping V onto $GF(2)^{n-m}$ with kernel V_0 such that $u \in V_f$ if $\psi(u) = g(u)$, and $u \in V_1$ if $\psi(u) > g(v)$ for all $v \in V_f$.

Proof 4 From linear algebra, there exists an $(n - m - t)$-dimensional subspace W of V such that V is the direct sum of V_f and W. Thus, every $u \in V$ can be uniquely written as

$$u = w \oplus v, \quad w \in W, \quad v \in V_f.$$

Let $I : W \to GF(2)^{n-m-t}$ be any isomorphism, and define

$$\psi(u) = (I(w), g(v)),$$

where $I(w) \in GF(2)^{n-m-t}$ and $g(v) \in GF(2)^t$. This is well-defined due to the uniqueness of the representation. Then $\psi : V \to GF(2)^{n-m}$ is a homomorphism since for $u' = w' \oplus v'$ with $w' \in W$ and $v' \in V_f$,

$$u \oplus u' = (w \oplus w') \oplus (v \oplus v'),$$

and thus

$$\psi(u \oplus u') = (I(w \oplus w'), g(v \oplus v')) = (I(w), g(v)) \oplus (I(w'), g(v')) = \psi(u) \oplus \psi(u').$$

Additionally, $\psi(0u) = \psi(0) = (I(0), g(0)) = 0 = 0\psi(u)$. If $u \in V_f$, then $u = 0 \oplus u$ with $0 \in W$ and $u \in V_f$, so $\psi(u) = (0, g(u))$ with numerical value $g(u)$. For $u \in V_1$, $I(w) \neq 0$, so the numerical value of the binary vector $(I(w), g(v))$ is larger than that of $g(v)$ for all $v \in V_f$.

Theorem 3 Let $G = (V, E)$ be the cellular automata graph of the finite digraph $G = (V, E)$. Then the following hold:

(i) The set of leaves L, where a leaf is a vector u such that $F(u) = \emptyset$, forms a subspace L when taken as a linear span over GF(2).

(ii) The zero-value subspace V_0, consisting of vectors with zero g-values, can be computed as the linear span of a specific set Q, where Q is defined by:

$$Q = (Z_{2(s+1)} \cap V_0) \cup S.$$

Here, Z_i is the set of vectors of weight at most i, and S is the set of leaves with weight at most 1.

(iii) The subspace V_f, containing all vectors representing valid game states, is the linear span of Q and $Z_{f(s+1)}$, where $Z_{f(s+1)}$ is the set of vectors in Z_{s+1} that are also in V_f.

(iv) The g-values on V_f are determined by its values on $Z_{f(s+1)} \cup \{0\}$.

(v) Dimension Bound ($t \leq \lfloor \log_2(1 + g(n, s)) \rfloor$) is the maximum g-value on V_f. It is at most $2^t - 1$, where t is bounded by the logarithm of a function $g(n, s)$ which depends on the game parameters.

Proof 5

(i) By definition, leaves are vectors with no followers. Therefore, any linear combination of leaves remains a leaf, establishing L as a subspace of V.

(ii) Clearly, $L(Q) \subseteq V_0$. Suppose for contradiction that there exists $u \in V_0 \setminus L(Q)$ with minimal counter function $c(u)$. By the minimality and structure of Q, u must be expressed as $u = w \oplus y$ for some $w \in Q$ and $y \in Q$. Hence, $u \in L(Q)$, a contradiction.

(iii) If $u \in V_0$, then $u \in L(Q) \subseteq L(Q \cup Z_{f(s+1)})$. For $u \in V_f \setminus V_0$, there exists $v \in F(u) \cap V_0$ and $w = u \oplus v \in Z_{f(s+1)}$. Thus, $u = v \oplus w \in L(Q \cup Z_{f(s+1)})$.

(iv) Clearly, $S_r \subseteq S_l$. If $j = 0$, then $j \in S_r$. For $j \neq 0$, pick $u \in V_j$. There exists $v \in F(u) \cap V_0$ such that $w = u \oplus v \in Z_{f(s+1)}$, ensuring $j \in S_r$.

(v) Let $u \in V_f$ have the maximum g-value. By the previous points, u can be considered within $Z_{f(s+1)}$. The outdegree of u is constrained by the function $g(n, s)$, leading to the bound $2^t - 1 \leq g(n, s)$.

The dimensions of V and its subspaces V_0 and V_f are useful for the computation of lexicodes derived from the digraphs of two-player Celata games. The number of basis vectors in V_f, which corresponds to the number of linearly independent columns in the adjacency matrix of the digraph over GF(2), determines the structure and span of the vector space. This number, m, is used to calculate the lexicodes, ensuring that each lexicode is uniquely represented within the subspace V_f.

4 Computation of the Lexicode

As mentioned the set of P-positions in an acyclic cellular automata game constitutes a code. Specifically, for an acyclic game, the lexicode with distance $d = s + 2$ forms a basis for the P-positions.

Definition 8 P-positions can be represented as vectors in a vector space over GF(2), the finite field with two elements. Each P-position u can be written as: $u = (u_1, u_2, ..., u_l, m)$ where $u_1 < u_2 < ... < u_l < m$ and $g(z_{i_1}) \oplus g(z_{i_2}) \oplus ... \oplus g(z_{i_l}) \oplus m = 0$.

By the properties of the g-function, $g(z_{i_1}) < g(z_{i_2}) < ... < g(z_{i_l}) < m$. If $g(z_m) = 2^k$, u cannot be a P-position since 2^k is not the sum of distinct nonnegative integers less than 2^k.

Theorem 4 The dimension of p is determined by the number of seeds $\leq 2t - 1$, where t is the smallest integer such that $g_s(z_n) \leq 2t - 1$. Also If $g_s(z_m)$ is a seed, then there exists $u \in P_s$ with $u_m = 1$ and $w(u) = s + 2$.

Proof 6 By Definition 7, if $g(z_m) = 2^k$, u cannot be a P position since 2^k is not the sum of distinct nonnegative integeres less that 2^k. Also if $g(z_m)$ is a seed, there are powers of 2 such that $g(z_m) \oplus \sum g(z_j) \in R = 0$, where R is a suitable subset for T. Hence $v := g(z_m) \oplus \sum g(z_j) \in P$, and these v are linearly independent, forming a basis for P.

Lemma 3 The set S formed by the vectors v as described above is linearly independent, ensuring that S constitutes a basis for P.

Proof 7 Since very z_m such that $g(z_m)$ is a seed appearing in S only once, and each unique basis number, S is linearly independent.

Lemma 4 For any finite set $S \subseteq Z_{\geq 0}$, mex(S) is the smallest nonnega-

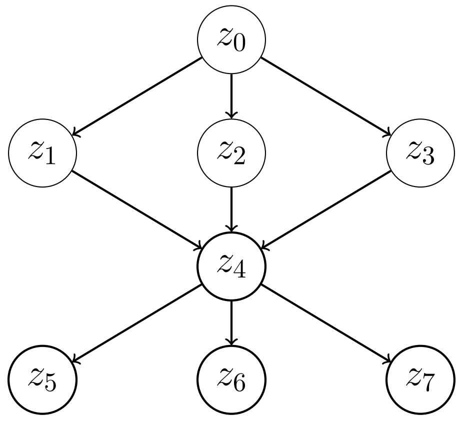

Figure 1: A simple Celata game graph

tive integer in S.

Procedure: To compute the lexicode P_s for a given acyclic Celata game, we use the following algorithm. First, we for each state z_m, we compute $g_s(z_m)$ using the mex function. Specifcally for $m \in \{1,..,n\}$, we calculate $g_s(z_m) = mex\{g(z_{i_1} \oplus g(z_{i_2}, ..., g(z_{i_j} : 1 \leq i_1 < i_2 < ... < i_j < m, j \leq s\}$

Once we have $g_s(z_m)$ values, we identify the seeds, which are the g-values that are not the sum of distinct smaller g-values. For each seed $g_s(z_m)$, we generate basis members by including the seed itself and all powers of two that appear in it. For example, if a seed is 13, the basis member induced by this seed would be 13,8,4,1.

After identifying the basis members, we employ a greedy algorithm to construct the lexicode. We begin with an ordered set of vectors $V_m = \{0, 1, ..., 2m - 1\}$ in lexicographic order. The lexicode L is initialized with the zero vector. We then iterate through each vector v in V_m, adding v to L only if it maintains a minimum Hamming distance d from all vectors already in L. This ensures that the resulting lexicode has the desired minimum distance property, making it an appropriate code for error correction.

The complexity analysis of this algorithm shows that computing $g_s(z_m)$ for each state m involves $O(n^{s+1})$, which is the dominant term. The subsequent steps of determining basis members and applying the greedy algorithm are less computationally intensive, with the overall time complexity being $O(n^{d-1})$ and the space complexity being $O(n^{d-2})$.

Example 3 Consider the Celata game displayed in Fig 2 with n = 8 and s = 3. G-value of every state in the graph is computed using the mex function:

$$g(z_0) = 0$$
$$g(z_1) = 1$$
$$g(z_2) = 2$$
$$g(z_3) = 4$$
$$g(z_4) = 8$$
$$g(z_5) = 15$$
$$g(z_6) = 16$$
$$g(z_7) = 32$$

We construct and initial matrix W for these values:

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

From this initial matrix, we select s = 3 rightmost linearly independent columns, achieving:

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

These columns represent the basis of vector states for our game. To compute the lexicode using the greedy algorithm we proceed by constructing the 2^s elements of V^s in lexigrophic order: $V^s = \{0, 1, ..., 2^s - 1\}$. For s = 3, this set is $V^3 = \{0, 1, 2, 3, 4, 5, 6, 7\}$. These elements are mapped to binary vectors: $V^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

We then construct the set V^n by applying the mapping $A_k = W_{independent} \cdot K$, where K is the column vector of the binary value of k. Using the greedy algorithm, we select codewords maintaining the minimum Hamming distance d = 5.

The resulting lexicode is $P = \{(00000000), (00001100), (00110011), (11110000)\}$.

5 The Special Case $q = 1$

In the context of Celata games, $q = 1$ refers to the restriction that each move affects exactly one token. This simplifies the analysis and allows for polynomial-time computation of g-values and lexicodes.

Lemma 5 Given a basis $\{z_0, z_1, \ldots, z_{n-1}\}$ of the vector space V, any additional vector $v \in V \setminus \text{span}\{z_0, z_1, \ldots, z_{n-1}\}$ can extend this basis to $\{z_0, z_1, \ldots, z_{n-1}, v\}$.

Proof 7 Since $v \notin \text{span}\{z_0, z_1, \ldots, z_{n-1}\}$, v is linearly independent of the basis vectors. Adding v to the basis extends the dimension by 1, forming a new basis for the extended space.

Lemma 6 The subspace V_f, containing all vectors representing valid game states, is the linear span of a set Q and $Z_f(s+1)$, where $Z_f(s+1)$ is the set of vectors in Z_{s+1} that are also in V_f. The g-values on V_f are determined by its values on $Z_f(s+1) \cup \{0\}$. The lexicode derived from V_f can be computed efficiently using polynomial-time algorithms, leveraging the structure and dimensionality of V_f.

To compute the g-function for an acyclic Celata game with $q = 1$, we use an approach that builds upon the structured subspaces V_0 and V_f. The algorithm proceeds as follows:

We start by constructing the subgraphs $G[2]$ and $G[4]$. The subgraph $G[2] = (V[2], E[2])$ is induced by the set $Y_2 \cup S$, where Y_2 contains vectors of weight 2, and S is the set of leaves. Similarly, $G[4] = (V[4], E[4])$ is induced by $Y_4 \cup Y_2 \cup S$, with Y_4 containing vectors of weight 4. These subgraphs help in isolating and examining specific portions of the original graph G relevant to the computation of the g-values.

Next, we apply the first iteration of Algorithm GSG (Generalized Sprague-Grundy) to $G[4]$. This step involves identifying a set $Q = \{q_1, \ldots, q_p\}$ of

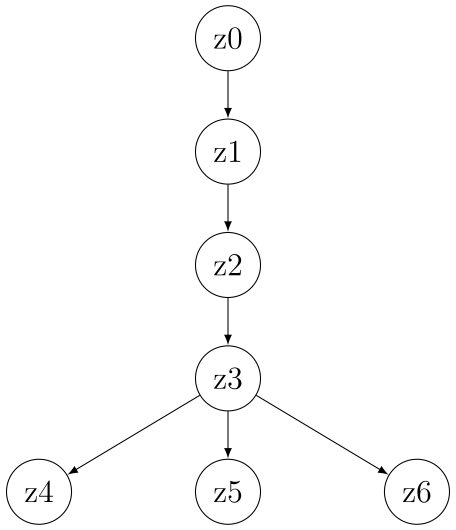

Figure 2: Illustrating Example 4

vectors in V_0, along with their counter values c. These vectors and values are stored for subsequent steps. Similarly, we apply Algorithm GSG to $G[2]$, where the largest g-value is determined to be $2^t - 1$ for some integer t. The vectors $v_1, \ldots, v_t$ with specific g-values are stored alongside their monotonic counter values c.

Following this, we construct a matrix A comprising vectors $q_1, \ldots, q_p$, $v_1, \ldots, v_t, z_0, \ldots, z_{n-1}$, where z_i are unit vectors. This matrix is then transformed into a row-echelon matrix E using elementary row operations. The indices of the unit vectors in E help determine the bases for the subspaces V_0 and V_f.

Specifically, let $1 \leq i_1 < \ldots < i_n \leq p + t + n$ be the indices of the unit vectors in E. The matrix B, consisting of columns $A_{i_1}, \ldots, A_{i_n}$ of A, provides a basis for V. The inverse matrix B^{-1} enables the computation of the homomorphism $\psi(z_i) = (\psi_{m_i}, \ldots, \psi_{n-1,i})$ for $0 \leq i < n$.

The homomorphism ψ enables us to represent all vectors as linear combinations of the basis vectors, facilitating the polynomial time computation of g-values across the vector space V. The following detailed example illustrates the application of this algorithm:

Example 4 Consider the game in Fig 2 game with $n = 7$. The adjacency matrix for this game's G is:

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

First, we construct the subgraphs $G[2]$ and $G[4]$. The subgraph $G[2]$ includes vertices with weights 2 or less. For simplicity, assume $G[2]$ contains z_0, z_1, z_2. The subgraph $G[4]$ includes vertices with weights 4 or less. Assume it contains z_0, z_1, z_2, z_3, z_4.

Applying the first iteration of algorithm to $G[4]$, we identify the set $Q = \{q_1, q_2\}$ with vectors $q_1 = z_1 + z_2$ and $q_2 = z_2 + z_3$. Then it becomes apparent that we find the largest g-value to be $2^1 - 1 = 1$, with vector $v_1 = z_0 + z_1$.

Next, we construct the matrix A including vectors $q_1, q_2, v_1, z_0, z_1, z_2, z_3, z_4, z_5, z_6$:

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

Transforming A into row-echelon matrix E:

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

The basis vectors are determined from E, and the homomorphism $\psi(z_i)$ is computed.

This example demonstrates how the algorithm applies to an acyclic digraph, transforming it into an equivalent row-echelon form and calculating the necessary homomorphisms to determine the g-values and structure of the lexicodes.

6 Forcing a Win in Celata Games

We can look into the strategy for forcing a win in acyclic cellular automata games by building on the concept of the g-function, computed using our polynomial-time algorithm, to determine P-, N-, and D-positions for any state in the game.

Definition 9 Let $R \subseteq V$. A representation u_e of $u \in V$ over R is a subset $u_e = \{u_1, \ldots, u_h\} \subseteq R$ of distinct elements u_i. If R is either indicated by the context or irrelevant, we may simply say that u_e is a representation of $u = \sum_{j=1}^{h} u_j$. The empty representation is denoted by $\emptyset$.

Definition 10 For $u_e = \{u_1, \ldots, u_h\} \subseteq R$ and $v_j \in F(u_j)$, a follower function for representations is given by

$$F_e(u_e, u_j, v_j) = \begin{cases} u_e \cup \{v_j\} \setminus \{u_j\} & \text{if } v_j \notin u_e \\ u_e \setminus \{u_j\} & \text{if } v_j = u_i \text{ for some } i \neq j. \end{cases}$$

Definition 11 For $u_e = \{u_1, \ldots, u_h\} \subseteq R$, the set of all representation followers of u_e is defined by

$$F_e(u_e) = \bigcup_{j=1}^{h} \bigcup_{v_j \in F(u_j)} F_e(u_e, u_j, v_j).$$

Lemma 7 For any $u_e = \{u_1, \ldots, u_h\} \subseteq R$:

(a) $F(\xi(u_e)) \subseteq \xi(F_e(u_e))$.

(b) $\xi(F_e(u_e)) \subseteq F(\xi(u_e)) \cup F^{-1}(\xi(u_e))$.

(c) Let $v_e = F_e(u_e, u_j, v_j)$, where $v_j \in F(u_j)$. Then $\xi(u_e) \in F(\xi(v_e))$ if and only if $v_j = \xi(F_e(u_e))$.

Theorem 5 For any integer $0 \le p < 2^t$, a function $\gamma : (R_j, V) \to R_j$ can be computed in polynomial time, such that if $u_e = \{u_1, \ldots, u_h\} \subseteq R_j$ is a representation of $\xi(u_e) \in V_p$ and $\xi(v_e) \in F(\xi(u_e))$ with $g(\xi(v_e)) > p$, then $\gamma(u_e, \xi(v_e)) = w_e \subseteq R_j$, where $\xi(w_e) \in F(\xi(v_e)) \cap V_p$ and $c(w_e) < c(u_e)$.

Proof 8 Let $v_0 \in F(\xi(u_e))$ with $g(v_0) > g(\xi(u_e))$. By (a), $v_0 \in \xi(F_e(u_e))$, say $v_0 = \xi(F_e(u_e, u_j, v_j))$. Since $w(u_j) \le 4$, the computation of j, k such that $v_j = F_k(u_j) = v_0 + \sum_{i \ne j} u_i$ takes $O(n^2)$ steps. For simplicity of notation, assume that $j = 1$. Since $g(v_0) > g(\xi(u_e))$ and $u_e = \{u_1, \ldots, u_h\}$ is a representation over R_j, i.e., its elements have g-values 0 or distinct powers of 2, it follows that $g(v_1) > g(u_1)$ and $v_1 \ne u_i$ (for $1 \le i \le h$), so $v_e^0 = \{v_1, u_2, \ldots, u_h\}$ is a representation of $v_0 = v_1 + \sum_{i=2}^{h} u_i$ over V. Also, $v_1 \in V_4$, hence v_1 has only $O(n)$ followers, so we can compute $w_1 \in F(v_1) \cap V_4$ with $g(w_1) = g(u_1)$ and $c(w_1) < c(u_1)$ in $O(n)$ steps. Hence $c(w_e^0) < c(u_e)$, where $w_e^0 = \{w_1, u_2, \ldots, u_h\}$ if $w_1 \ne u_i$ (for $2 \le i \le h$), or $w_e^0 = \{u_2, \ldots, u_{i-1}, u_{i+1}, \ldots, u_h\}$ otherwise, and in any case $w_e^0 \subseteq F_e(v_e^0) \cap R_j$, so $\xi(w_e^0) \in \xi(F_e(v_e^0))$. Hence, by (b), $\xi(w_e^0) \in F(\xi(v_e^0)) \cup F^{-1}(\xi(v_e^0))$.

If $\xi(w_e^0) \in F(\xi(v_e^0))$, we let $\gamma(u_e, \xi(v_e^0)) = w_e^0$, which satisfies the desired requirements. If $\xi(v_e^0) \in F(\xi(w_e^0))$, we replace the ancestor $\xi(u_e)$ of $\xi(v_e^0)$ by its ancestor $\xi(w_e^0)$ with representation w_e^0. A representation v_e^1 of $\xi(v_e^0) = \xi(v_e^1)$ can be obtained from w_e^0 based on (a), as at the beginning of this proof. As before we get $w_e^1 \subseteq F_e(v_e^1) \cap R_j$ with $c(w_e^1) < c(w_e^0)$ and $\xi(w_e^1) \in F(\xi(v_e^1)) \cup F^{-1}(\xi(v_e^1))$. This process thus leads to the formation of two sequences $v_e^0, v_e^1, \ldots; w_e^0, w_e^1, \ldots$, where $\xi(v_e^0) = \xi(v_e^i)$ (for $i = 1, 2, \ldots$), $w_e^i \subseteq R_j$, $\xi(w_e^i) \in F(\xi(v_e^i)) \cup F^{-1}(\xi(v_e^i))$ (for $i = 1, 2, \ldots$). Since $c(w_e^0) > c(w_e^1) > \ldots$, these sequences must be finite. In fact, each sequence has at most $O(n^5)$ terms. Since this process keeps producing a new sequence term if $\xi(w_e^i) \in F^{-1}(\xi(v_e^i))$, there exists $j = O(n^5)$ such that $\xi(w_e^j) \in F(\xi(v_e^j))$. We then define $\gamma(u_e, \xi(v_e)) = w_e^j$, which satisfies the desired requirements. Finally, it can be decided in $O(n)$ steps whether $\xi(w_e^i) \in F(\xi(v_e^i))$ or $\xi(v_e^i) \in F(\xi(w_e^i))$ by using (c).

Theorem 6 Given an N-position in a sum of r games containing a two-player acyclic cellular automata game played on a finite digraph $G = (V, E)$

with $|V| = n$. The subset M of moves on G leading to a win has size $O(n^5)$, and its computation needs $O(n^6)$ steps.

Proof 9 Apply Algorithm CEL to G $O(n^6)$ steps. Given an N-position of the sum, we may assume that a winning move is of type (ii). So we have to move from a vertex u in G, which corresponds to $u \in V_\ell$ or to $g(u) = 1(K)$, $p \in K$ in the cellular automata game-graph, to $v \in F(u) \cap V_p$.

Assume first $u \in V_\ell$. Compute $B^{-1}u$ to get a representation $u_e = \{u_1, \ldots, u_h\} \subseteq R_s$ with $\xi(u_e) = \sum_{i=1}^{h} u_i$, where $s = \lceil \log_2(\ell + 1) \rceil$ $O(n^2)$ steps. For a move of type (ii), let $v \in F(\xi(u_e)) \cap V_p$. By (a), $v \in \xi(F_e(u_e))$, say $v = \xi(F_e(u_e, u_1, v_1))$. As we saw at the beginning of the proof of Theorem 5, the computation of $v_1 = F_k(u_1) = v + \sum_{i \neq j} u_i$ takes $O(n^2)$ steps. It can always be arranged so that $g(v_1) < g(u_1)$. Also $w(v_1) \leq 4$. Thus $v_e^1 = \{v_1\}$ is a representation. Replacing u_1 by v_1 in u_e and deleting v_1 if $v_1 = u_i$ for some i, we get a representation v_e of v over R_j, where $j = \lceil \log_2(p+1) \rceil$ $O(n)$ steps. Since $p < \ell$ and c is monotonic we have $c(v_e) < c(u_e)$.

Secondly, assume $g(u) = 1(K)$. Scan the $O(n^2)$ followers of u, to locate one, say v, which is in V_p. Compute $B^{-1}v$ to yield a representation $v_e = \{v_1, \ldots, v_h\} \subseteq R_j$ with $v = \xi(v_e) = \sum_{i=1}^{h} v_i$, where $j = \lceil \log_2(p + 1) \rceil$ $O(n^4)$ steps. By definition, $c(v_e) < c(u_e)$.

In any subsequent move of type (ii) we compute the new representation from the previous one as was done above for the case $u \in V_\ell$, where v_e was computed from u_e in $O(n)$ steps.

For a move of type (i), assume that player II moves from $u_i = \xi(u_e^i) \in V_p$ with $u_e^i \subseteq R_j$ where $j = \lceil \log_2(p + 1) \rceil$ to $v_i \in F(u_i)$ with $g(v_i) > g(u_i)$. Then player I computes $u_e^{i+1} = \gamma(u_e^i, v_i)$ $O(n)$ steps and moves to $u_{i+1} = \xi(u_e^{i+1}) \in F(v_i) \cap V_p$ such that $c(u_e^{i+1}) < c(u_e^i)$. This can be done as we saw in Theorem 5.

Thus, c decreases strictly for both a move of type (i) and of type (ii). Since $c(u_e) = O(n^5)$, player I can win in $O(n^5)$ moves made in the acyclic cellular automata game, for whatever sequence of moves of type (i) and (ii) is taken. This is in addition to any other moves in the other sum components. Since each computation of one move of type (ii) and of γ requires $O(n)$ steps, the entire computation time for player I in the acyclic cellular automata game is $O(n^6)$.

6.1 Lexicodes From Winning Positions

We now explore how strategies and equations for forcing a win in acyclic Celata games can be leveraged to create lexicodes with polynomial-time complexity. This process utilizes previously developed polynomial-time algorithms through the following steps:

1. Construct subgraphs $G[2]$ and $G[4]$ to isolate specific portions of the original game graph G.

2. Apply the Generalized Sprague-Grundy (GSG) algorithm to compute g-values and identify critical states. Using the follower function F_e, we calculate g-values in polynomial time, ensuring transitions preserve the structure required for error-correcting codes:

$$\gamma(u_e, \xi(v_e)) = w_e \subseteq R_j, \xi(w_e) \in F(\xi(v_e)) \cap V_p \text{ and } c(w_e) < c(u_e).$$

3. Represent these states to form basis vectors for the lexicode. Encode each game state u_e and its transitions $\xi(u_e)$ as binary strings. Transitions are determined using the follower function F_e and the polynomial-time computable function γ. Form the lexicode by concatenating these binary representations.

4. Ensure the lexicode maintains a lexicographic order to guarantee a minimum Hamming distance.

Example 5 Consider a game on a digraph $G = (V, E)$ with vertices $\{z_0, z_1, z_2, z_3, z_4\}$. The adjacency matrix A is given by:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Following the algorithm to construct the lexicode:

1. **Identify Subgraphs**: Assume $G[2]$ contains vertices $\{z_0, z_1, z_2\}$ and $G[4]$ contains vertices $\{z_0, z_1, z_2, z_3, z_4\}$.

2. **GSG Algorithm Application**: Apply the GSG algorithm to $G[4]$ to find the set $Q = \{q_1, q_2\}$ with vectors $q_1 = z_1 + z_2$ and $q_2 = z_2 + z_3$.

3. **Compute g-values and Representations**:

$$g(z_0) = 0, g(z_1) = 1, g(z_2) = 2, g(z_3) = 3, g(z_4) = 4$$

$$u_e(z_0) = \emptyset, u_e(z_1) = \{z_0\}, u_e(z_2) = \{z_1\}, u_e(z_3) = \{z_2\}, u_e(z_4) = \{z_3\}$$

4. **Form the Lexicode**: Select vectors in lexicographic order:

$$\{(0,0,0,0,0),(1,0,0,0,0),(1,1,0,0,0),(1,1,1,0,0),(1,1,1,1,0),(1,1,1,1,1)\}$$

This lexicode has a minimum Hamming distance of 2, ensuring it can correct single-bit errors.

Example 6 Consider another simple example with game states $V = \{u, v, w\}$ and their representations u_e, v_e, w_e:

$$u_e = \{u_1, u_2\}, v_e = \{v_1, v_2\}, w_e = \{w_1, w_2\}$$

Using the polynomial-time algorithm, classify u and v as P-positions and w as an N-position. Compute the set of moves M leading from w to a P-position:

$$u_e \to 00, v_e \to 01, w_e \to 10$$

Form the lexicode by concatenating these binary representations:

$$\{00, 01, 10\}$$

To ensure the lexicodes preserve error-correcting properties, we use the structure of game state transitions and the follower functions. Each transition is computed as follows:

1. Compute the follower function F_e for a given representation u_e.
2. Use the function γ to find the next state w_e such that $c(w_e) < c(u_e)$.
3. Encode the state w_e as a binary string, maintaining polynomial-time complexity for each transition.

7 Epilogue

In this paper, we have extended the framework of two-player cellular automata games to derive error-correcting codes, specifically focusing on the construction of lexicodes. Our approach leverages the polynomial-time computability of Grundy values and critical states, facilitated by the follower function F_e and the polynomial-time function γ. This ensures that the derived lexicodes maintain their error-correcting properties efficiently.

Moreover, we have shown that the derived lexicodes from these games are not only theoretically sound but also practically implementable. The polynomial strategy used for acyclic games guarantees the feasibility of computing winning moves and subsequently the error-correcting codes.

A collection of two-player cellular automata games with only a minimum of the underlying theory can be found in various references. An obvious remaining question is whether Celata games have a polynomial strategy for every $q > 1$. We have, in fact, provided a polynomial infrastructure for the general case, in the sense that everything up to the end of Section 6 is consistent with a polynomial strategy for all $q \geq 1$. However, $V[q+1]$ and $V[2(q+1)]$ are not restrictions of V when $q > 1$, so we cannot apply Lemma 3. Therefore, we cannot prove polynomiality for $q > 1$ in the same way we used for 1-regular games.

The special case of 1-regular games has been analyzed in previous works. Our current paper is a generalization, simplifying and clarifying many aspects of these earlier studies. Misère play of 1-regular games, where the player making the last move loses, has been investigated as well. 1-regular games are barely polynomial, with complexity $O(|V|^6)$, and any perturbation typically results in PSPACE-hard games.

A strategy in the broad sense, defined by Kalmár and Smith, depends on the present position and all its antecedents from the beginning of the play. While both authors concluded that it suffices to consider strategies in the narrow sense, depending only on the present position, our exploration revealed that a broad sense strategy was necessary for computing a winning move in polynomial time due to the constraints of maintaining polynomiality within a small subgraph of the game-graph.

Cellular automata games can potentially lead to linear error-correcting codes with Hamming distances greater than 4. The current work was motivated by the desire to create such games, which naturally induce codes with superior error-correcting properties. In practice, the computation of V_0, which is all that's needed for the codes, can often be done by inspection. The best codes may be derived from a simplified digraph, where V_0 can also be computed easily.

We aimed to further explore polynomial games with token interactions and to create two-player cellular automata games. Three key ideas were used:(I) additivity of γ^{**}, (II) computation by restriction, namely computing V_0, V_f, and γ with a restricted linear span of sparse vectors of polynomial size, (III)computing a winning move efficiently.

While (I) and (II) are polynomial for all two-player cellular automata games, this has been shown for (III) only for the special case $q = 1$ (1-regular games). Thus, the main open question is the complexity status of (III) for $q > 1$. Another question that hasn't been settled in this paper is what happens when loops are permitted in the ground graph. Previously this has been investigated by Fraenkel with his method of quantifying cyclic cellular automata games.

References

[1] Fraenkel, A., “Combinatorial games with an annihilation rule,” in *The Influence of Computing on Mathematical Research and Education, Symposium in Applied Mathematics*, J. LaSalle, Ed., vol. 20. American Mathematical Society, 1974, pp. 87–91.

[2] Fraenkel, A., “Complexity, appeal and challenges of combinatorial games,” expanded version of a keynote address at Dagstuhl Seminar “Algorithmic Combinatorial Game Theory”, Feb. 17–22, 2002. To appear in *Theoretical Computer Science*, special issue on Algorithmic Combinatorial Game Theory. Preprint available: http://www.wisdom.weizmann.ac.il/~fraenkel.

[3] Fraenkel, A., “Error-correcting codes derived from combinatorial games,” in *Games of No Chance*, R. Nowakowski, Ed., vol. 29. Cambridge University Press, 1996, pp. 417–431.

[4] Fraenkel, A., “Two-player games on cellular automata,” in *More Games of No Chance*, R. Nowakowski, Ed., vol. 42. Cambridge University Press, 2002, pp. 279–306.

[5] Ferguson, T., “Misère annihilation games,” *Journal of Combinatorial Theory, Series A*, vol. 37, pp. 205–230, 1984.

[6] Trachtenberg, A., and Vardy, A., “Lexicographic codes: Constructions, bounds, and Trellis complexity,” in *31st Annual Conference on Information Sciences and Systems*, 1997. [Online]. Available: http://citeseer.nj.nec.com/448945.html

[7] Yesha, Y., *Theory of Annihilation Games*, PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1978.