SEQUENCES FOR SOLVING PUZZLES AND TOURING GRAPHS

EMILY CARLSON AND TSELIL SCHRAMM

ADVISOR: PAUL CULL
OREGON STATE UNIVERSITY

ABSTRACT. We examine Generalized Towers of Hanoi, Generalized Spin-Out, and the Combination Puzzle, and continue to describe the puzzles and their properties. We introduce Finite State Transducers (FSTs) that compute the shortest sequence of winning moves for each of these puzzles for all dimensions, and show that the solution sequence for Spin-Out is not finite-state computable when going from configuration $11\dots1$ to configuration $00\dots0$. We also examine the graphs of these puzzles, the class of graphs known as iterated complete graphs. We show that Hamiltonian paths and circuits exist but are not unique for iterated complete graphs of dimension greater than 3. We then introduce FSTs which produce Hamiltonian paths on K_d^n for every $d \in \mathbb{N}$, and prove the transducers' minimality for d odd – these FSTs define a d-ary Gray code. Finally, we create FSTs that produce a Hamiltonian circuit on K_d^n for all d.

1. INTRODUCTION

The Towers of Hanoi is a classic puzzle, and has been studied by mathematicians for over 100 years [8]. The traditional game is played with n disks on three towers, which we label 0, 1, and 2. Typically, the goal is to use the fewest number of moves to transfer all of the disks from tower 0 to tower 2, moving only one disk at a time and never placing a larger disk on top of a smaller disk. There are several well-known algorithms that solve the Towers of Hanoi, and the $(2^n - 1)$-step minimal solution sequence of moves has also been shown to be unique [2]. There is a notion of generalizing the towers of Hanoi to an arbitrarily large number of towers, and an algorithm to solve it has been proposed, but the optimality of this algorithm has remained an open problem since 1941 [5].

Another puzzle of interest is Spin-Out. This puzzle is composed of n spinners, each of which can be locked or unlocked. The spinners are arranged in a line on a rail, so that if a spinner is locked the whole line cannot slide off the rail. Furthermore, the spinners are arranged such that a spinner can only be unlocked if its neighbor to the right is locked and all other spinners to the right are unlocked (see Figure 2.2). The typical goal of the puzzle is to change the configuration from all locked spins to all unlocked spins using the fewest possible moves, which is $\lceil \frac{2}{3}(2^n - 1) \rceil$ [8].

In this paper, we work with generalized versions of both puzzles developed by Paul Cull and his students in [1], [10], and [4]. We use a version of Generalized Towers of Hanoi (hereafter abbreviated GTH) with extra restrictions placed on the allowed moves and the stipulation that the number of towers must be odd. These extra restrictions on GTH have allowed students to develop algorithms that solve GTH in $2^n - 1$ moves. The GTH puzzle is further described in Definition 2.1. We also use a modified version of Spin-Out, in which each spinner consists of a "stack" of

This work was done during the Summer 2011 REU program in Mathematics at Oregon State University.

m spinners, effectively creating 2^m possible configurations for each spinner stack. The rules of Generalized Spin-Out are described further in Definition 2.2.

A Combination Puzzle, explicitly stated in Definition 2.3, can be created from solving Generalized Towers of Hanoi and Generalized Spin-Out simultaneously, where a move is only valid it both the GTH component and Generalized Spin-Out component are legally able to make that move. In this case, there are q towers and m spinners per stack.

These generalized puzzles can each be represented as graphs, in which the vertices correspond to game configurations, and an edge exists between two vertices if and only if it is possible to move from one configuration to another using a single move. It turns out that GTH, Generalized Spin-Out, and the Combination Puzzle have graphs called iterated complete graphs [6] [10]. For each $d \in \mathbb{N}$, with $d = 2^m \cdot q$ for some odd q, the iterated complete graph K_d^n corresponds to the Combination Puzzle with q towers and m spinners per stack. When d is odd, the puzzle is actually an instance of GTH, and when $q = 1$ the puzzle is actually an instance of Generalized Spin-Out.

In this paper, we begin by setting forth some basic definitions and terminology in Section 2. In Section 3, we create Finite State Transducers (FSTs) which map a puzzle's configuration to its index in the minimal solution sequence and vice-versa; these FSTs also work as accept/reject machines, rejecting a configuration if it is not in the minimal solution sequence. We successfully define such a machine for GTH and the Combination Puzzle. For Generalized Spin-Out, we construct such an FST for the solution sequence from locked position $100\ldots0$ to unlocked position $000\ldots0$. We then show that it is not possible to compute the index of a Spin-Out configuration with finite state in the more traditional version of the game, when moving between the locked state $111\ldots1$ and the unlocked state $000\ldots0$.

We also discuss the iterated complete graphs associated with these puzzles. It has already been shown that the iterated complete graph K_d^n contains a Hamiltonian path (and for $d > 2$, a Hamiltonian circuits), and these have been shown to be unique for $d = 2,3$. In section 4 of this paper, we show that the Hamiltonian paths and circuits are not unique for $d > 3$. Then in sections 5 and 6, we give FSTs that describe a Hamiltonian path and a Hamiltonian circuit on K_d^n by mapping a vertex's label in the graph to its index along the path or circuit.

In the remainder of the paper, we discuss some tangential results and conjectures. In section 7, we describe some work involving perfect one error correcting codes. Finally, in section 8, we discuss the minimum number of disks necessary so that all the towers are used in GTH.

2. Definitions and Terminology

2.1. **Puzzle Rules.**

Definition 2.1. In *Generalized Towers of Hanoi*, we have n disks and d towers, where $d \geq 3$ and odd, and label the towers $0, 1, \ldots d-1$. Generalized Towers of Hanoi, created by Kathleen King in 2004, also requires that you only move one disk at a time and that a larger disk cannot be stacked on top of a smaller disk. However, we also have the following rules:

(1) A disk cannot be moved unless all of the disks smaller than it are on the same tower.
(2) A disk on tower b is able to move when all of the smaller disks are on a single tower a. The disk can move to tower $(2a-b) \mod d$ [4].

We also must solve this puzzle in the least possible moves: $2^n - 1$.

Definition 2.2. In *Spin-Out*, we have n pieces of stacks of m spinners, where $m \in \mathbb{N}$. Each spinner in the stack can be locked and unlocked, so we represent the spinner's configuration as m-bit binary numbers (see Figure 2.1). The rules are as follows:

(1) The smallest spinner's configuration can be changed to any other configuration at any time.
(2) We can only change spinner s_i's configuration from a to b if spinner s_{i-1} is in configuration $a \oplus b$ and $s_{i-2}, \dots, s_0$ are completely unlocked (in configuration 0).

Traditionally, solving the puzzle is defined as moving from all locked position a to all unlocked position b in the fewest number of moves: $\lceil \frac{2}{3}(2^n - 1) \rceil$. Yet if we want to travel from a corner of K_r^n to another corner of K_r^n, then we can do so in a minimum of $2^n - 1$ moves.

Figure 2.2 shows a physical embodiment of this puzzle.

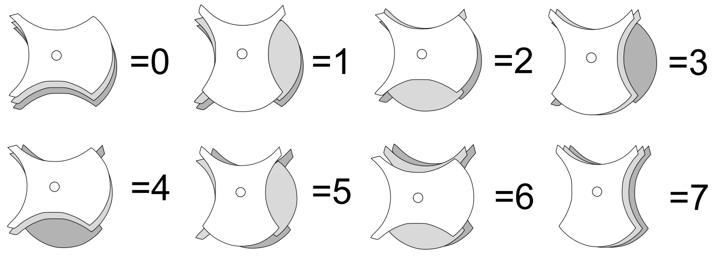


FIGURE 2.1. The configurations of pieces with $m = 3$ spinners and their labels.

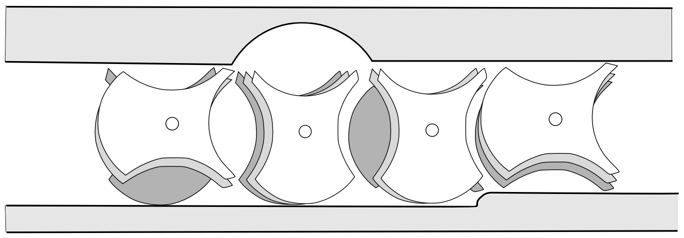


FIGURE 2.2. Example of a configuration of Spin-Out with dimension 8.

Definition 2.3. The *Combination Puzzle* for $d = 2^m \cdot q$ with q odd has q towers and n pieces, each of which is a disk with a stack of m spinners on it. It has the following rules:

(1) The smallest piece's configuration may be changed to any configuration at any time.
(2) We can change piece $x_i = (t_i, s_i)$'s configuration from $a = (t_a, s_a)$ to $b = (t_b, s_b)$ if
    (a) $s_{i-1} \oplus s_a = s_b$,
    (b) $s_{i-2}, \dots, s_0 = 0$, and
    (c) $t_{i-1}, \dots, t_0 = \frac{t_a + t_b}{2}$.

The correct solution sequence uses the fewest possible number of moves: $2^n - 1$.

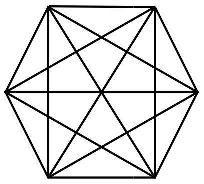
2.2. **Iterated Complete Graphs and Labeling.** First, we'll give some basic Graph Theory definitions, as in previous REU papers [8] and [1].

Definition 2.4. A *graph* $G(V,E)$ has a finite set $V(G)$ of vertices and the unordered pairs of vertices create a set $E(G)$ of edges. If two vertices $v_i, v_j \in VG$ are *adjacent*, then there exists an edge between them and $(v_i, v_j) \in E(G)$.

Definition 2.5. The *degree* of a vertex is the number of other vertices it is adjacent to.

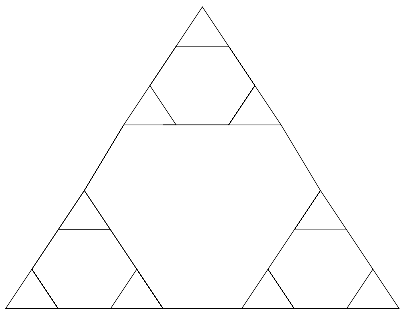
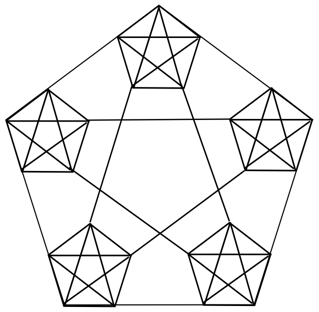
Definition 2.6. A *complete graph* on d, denoted K_d, has exactly d vertices in which every vertex is adjacent to every other vertex by exactly one edge.

Example 2.7. Below is an example of the complete graph on 6 vertices, K_6.



Definition 2.8. A *iterated complete graph* on q vertices with n iterations is represented by K_q^n. To create K_q^n recursively, we make q copies of K_q^{n-1} and connect them so that every subgraph K_q^{n-1} is connected to every other subgraph by one edge. Additionally, every subgraph K_q^{n-1} must have on vertex of degree $q-1$ while all other vertices have degree q.

Example 2.9. Below are examples of the iterated complete graphs K_3^3 and K_5^2.

Definition 2.10. A *subgraph* H of a graph G is made up of a subset of vertices $V(H) \subseteq V(G)$ along with a subset $E(H) \subseteq E(G)$ of edges associated with vertices in $V(H)$.

Definition 2.11. In a iterated complete graph K_d^n, a K_d^m*-subcopy* is a subgraph that consists of a copy of K_d^m where $1 \leq m < n$.

Definition 2.12. In a iterated complete graph on q vertices with n iterations, a *corner vertex* is a vertex with degree $d-1$ and a *non-corner vertex* is a vertex with degree d.

Remark 2.13. Throughout the paper, many of the iterated complete graphs have labeled vertices. The labels of these vertices are essentially unique, as they correspond directly to the configurations and the allowed moves between them. Instructions for labeling K_d^n geometrically and recursively can be found in [10] and in [6].

2.3. **Finite State Automata.**

Definition 2.14. A *Finite State Machine*, abbreviated *FSM*, is an abstract mathematical model composed of a finite number of states and transitions between these states that depend on an input value. The input value is usually a string of characters, which are read one character at a time. Because it has a finite amount of states, it has a finite internal memory. [7]

Definition 2.15. A *Finite State Transducer*, abbreviated *FST*, is a FSM that outputs a value at each transition. [7]

For an example of an FST, see Figure 2.3 below.

2.4. **Gray code.** The puzzles in this paper all share the restriction that only one piece can be moved per turn. This means that the configuration strings of consecutive moves have the *Gray property*.

Definition 2.16. A sequence of strings has the *Gray property* if the k^{th} string and the $(k+1)^{th}$ string differ by a single digit [9].

This means that any sequence of moves in the generalized puzzles corresponds to a Gray code, in that it establishes a correspondence between the natural numbers and a sequence of strings with the Gray property.

One particular form of Gray code, known as the binary reflected Gray code, is particularly tied to the Spin-Out puzzle – the Gray code strings correspond directly to the minimal sequence of configurations when moving from configuration $000\ldots0$ to $100\ldots0$ [11]. If we want to convert from a n-bit binary number $b_n\ldots b_1$ to n-bit binary reflected Gray code $g_n\ldots g_1$, we have the formulas

$$b_n = g_n \qquad \text{and} \qquad b_{n-i} = g_{n-i} + b_{n-i+1}, \tag{2.1}$$

from which we can create the FST shown in Figure 2.3.

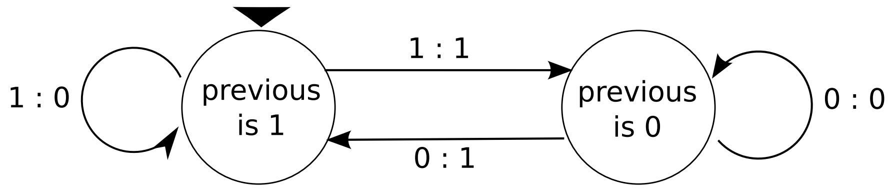


FIGURE 2.3. The FST which converts binary to binary binary reflected Gray code.

We also note that since the vertices of K_d^n correspond to configurations of the puzzle of dimension d, adjacent vertex labels differ by only one digit. Hence, any walk on K_d^n defines a Gray code. Furthermore, the labels along a Hamiltonian path on K_d^n define a bijection between n-bit d-ary numbers and some n-bit d-ary Gray code. Hence, our FST's that define Hamiltonian paths on K_d^n, described in section 6, compute a d-ary Gray code.

3. A FINITE STATE TRANSDUCER FOR COMPUTING THE INDEX OF A CONFIGURATION IN THE SOLUTION SEQUENCE.

3.1. **Solution Sequences.**

Definition 3.1. A *solution sequence* of a game is the sequence of configurations that describes the winning moves of the game in the order they are played.

Given an instance of GTH with d towers in which we wish to move the stack of disks from tower a to tower b, the solution sequence is exactly each of the moves in the minimum sequence of legal moves between configuration $aa\dots a$ to the configuration $bb\dots b$.

For an instance of Spin-Out with 2^m spinners per stack, we have two solution sequences:

(1) When trying to move between all spinners locked in position a and all spinners unlocked in position 0, the solution sequence is exactly each of the moves in the minimum sequence of legal moves between configuration $aa\dots a$ to $00\dots 0$. There are $\lceil \frac{2}{3}(2^n-1)\rceil$ configurations in this solution sequence.

(2) Alternatively, when moving from configuration $a00\dots 0$ to configuration $000\dots 0$ in Spin-Out, the solution sequence contains 2^n-1 configurations. We call this version Corner-to-Corner Spin-Out.

The latter solution sequence is simply the shortest path between two corners of $K^n_{2^m}$, and hence is in some ways the more “natural” notion of gameplay; however, the former version is more traditional.

The Combination Puzzle is similar; the solution sequence is composed only of the configurations necessary to move between a starting configuration $aaa\dots a$ and an ending configuration $bcc\dots c$.

The solution sequences for GTH, the Combination Puzzle and Spin-Out all correspond to d-ary Gray codes; this is because our games only allow the movement of one piece at a time, and so consecutive configurations differ in only one digit. For the case of Corner-to-Corner Spin-Out, the configurations in the solution sequence and the order in which they appear corresponds directly to the n-bit binary reflected Gray code [11] (this is only true if we regard *any* nonzero configuration to be locked, or “normalize” the configuration so that the only entries are 0’s and 1’s).

In this section, we describe a FST that maps a configuration from an instance of GTH to a binary number corresponding to its index in the GTH solution sequence, and prove its minimality. We then describe an FST that maps a configuration to a binary number corresponding to its index in the Spin-Out solution sequence when moving from configuration $00\dots 0$ to $a00\dots 0$. Then, we define an FST that maps a configuration from an instance of the Combined Puzzle to its index in the solution sequence in which the disks move from tower a to tower b and the spinners change from configuration $c00\dots 0$ to $00\dots 0$. Finally, we show that there can be no FST that maps configuration to solution sequence index for *any* instance of Spin-Out when moving from configuration $aa\dots a$ to $00\dots 0$.

3.2. **Configuration-to-Index FST for Generalized Towers of Hanoi.**

In order to describe and prove the correctness of this FST, we first find a simple way to describe all legal moves in an instance of GTH.

Lemma 3.2. *In a game of GTH with d towers in which the disks are moved from tower a to tower b, $a,b \in \{0,\dots,d-1\}$, a move from tower i to tower j is only valid if*

$$i-j \equiv 2^c(b-a) \bmod d$$

for some integer c.

Proof. Consider an instance of GTH with n disks and d towers. Suppose that a move of the kth smallest disk from tower i to tower j is valid. Then at one point the k smallest disks were all stacked on tower i, and the stack of the k smallest disks will eventually be re-stacked on tower j. Because the smallest disk is one of the k smallest disks, it must be transferred from tower i to tower j; by the equal increment theorem, this will involve some number of moves of the smallest disk by $\frac{b-a}{2^{n-1}}$ towers at a time, so

$$i-j \equiv m \cdot \frac{b-a}{2^{n-1}} \bmod d$$

for some integer m.

Now, realize that the movement of the tower of size k will take 2^k-1 moves in total, 2^{k-1} of which are movements of the smallest disk. Hence, $m=2^{k-1}$. Therefore,

$$i-j \equiv 2^{k-1} \cdot \frac{b-a}{2^{n-1}} \bmod d \equiv 2^{k-n}(b-a) \bmod d.$$

Then we set $c=k-n$, and conclude that a move from tower i to j is valid only if $i-j \equiv 2^c(b-a) \bmod d$ for some integer c, as desired. □

We can use these principles and the recursive algorithm described in [10] to construct a Finite State Transducer that takes as input a d-ary string representation of the game configuration and returns the position number in binary. The states will represent legal moves in the game, and the lemma above allows us to construct the exact necessary amount of states.

Definition 3.3. Let $|2|_d$ denote the multiplicative order of 2 in the ring $\mathbb{Z}/d\mathbb{Z}$ where d is odd.

Claim 3.4. *In the FST for d towers and n disks, there will be at most $d \cdot |2|_d$ non-trash states.*

Proof. Since we show in Proposition 8.2 that for $n > \log_2(d-1)$ all d towers are occupied, there may be starting moves from each of the d towers. Since 2 has finite order in $\mathbb{Z}_d$, there are $|2|_d$ towers that can be reached from each starting tower. Hence, we expect at most $d \cdot |2|_d$ states. □

Construction 3.5. We construct a Finite State Transducer $CtoI$ that takes as input a configuration string in d-ary and returns the configuration's binary index number in the sequence of winning moves when moving the stack of towers from tower a to tower b as follows:

(1) Let the input be a position string $x_{n-1}x_{n-2}\dots x_0$, where x_i corresponds to the ith largest disk's tower; thus $x_i \in \{0,1,\dots,d-1\}$.
(2) Create the set of states
$$S = \left\{ \big(i, i+2^m(b-a)\big) \ : \ i \in \{0,\dots,d-1\}\ ,\ m \in \{1,\dots,|2|_d\} \right\} \cup \big\{(Error)\big\}.$$
(3) Set the start state $s_0=(a,b)$, representing the starting and ending towers at the start of the game.
(4) Let $CtoI(x_{n-1}x_{n-2}\dots x_0;\ (a,b))$ be the output of the machine when $x_{n-1}\dots x_0$ is the input string and the machine is in state (a,b).
(5) Let $i,j \in \{0,\dots,d-1\}$, and define the transitions of $CtoI$ as follows:

$$CtoI\left(x_{n-1}x_{n-2}\dots x_0;\ (i,j)\right) = \begin{cases} 0\ CtoI\left(x_{n-2}\dots x_0;\ \left(i, \frac{i+j}{2}\right)\right) & : x_{n-1}=i, \\ 1\ CtoI\left(x_{n-2}\dots x_0;\ \left(\frac{i+j}{2}, j\right)\right) & : x_{n-1}=j, \\ CtoI\left(x_{n-2}\dots x_0;\ (Error)\right) & : \text{otherwise.} \end{cases}$$

$$CtoI\left(x_{n-1}x_{n-2}\dots x_0;\ (Error)\right) = CtoI\left(x_{n-2}\dots x_0;\ (Error)\right)$$

Figure 3.1 is an example of the FST for $d = 3$, or the traditional Towers of Hanoi game. An demonstration of $\mathcal{C}to\,I$'s operation for $d = 5$ is given in Example 3.7.

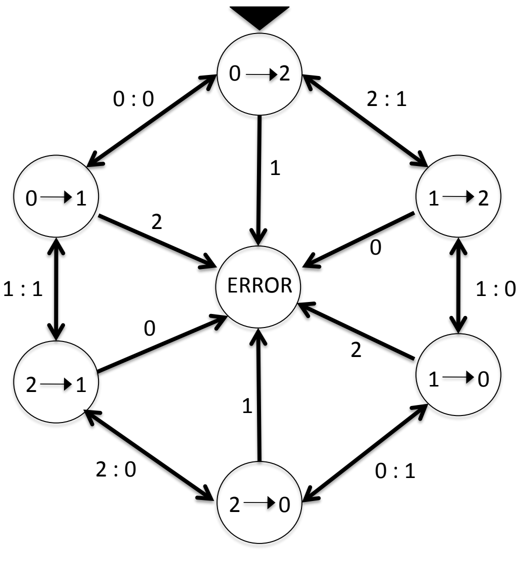


FIGURE 3.1. The FST from Construction 3.5 with $d = 3$.

Remark 3.6. When constructing the FST from Construction 3.5 for prime d, there are exactly $d(d-1)$ states. This is because $|2|_d = d$ for prime d, and so there are d products of the form$(b-a)2^m$. However, if we consider the case of $d = 9$, we note that $|2|_9 = 6$. Hence, if moving the disks from tower 0 to tower 8, for each $i \in \{0,\dots,8\}$ we only need construct the states

$$\begin{aligned} S &= \big\{(i,i+2^0\cdot 8),(i,i+2^1\cdot 8),(i,i+2^2\cdot 8),(i,i+2^3\cdot 8),(i,i+2^4\cdot 8),(i,i+2^5\cdot 8)\big\} \\ &= \big\{(i,i+8),(i,i+7),(i,i+5),(i,i+1),(i,i+2),(i,i+4)\big\}. \end{aligned}$$

Note that the states $(i,i+3)$ and $(i,i+6)$, need not be included, so there is some savings in the number of states because $|2|_9 < 9$.

Regardless, the number of states in $\mathcal{C}to\,I$ grows at a rate between d and d^2; hence, the machines for $d > 3$ are quite tedious to construct. However, we can demonstrate how such a machine works using the convenient notation described in Construction 3.5.

Example 3.7. We demonstrate the operation of $\mathcal{C}to\,I$ for $d = 5$ and $n = 5$, moving from tower 0 to tower 4.

Consider the configuration 42333. Here, the largest disk is on its destination tower, the second-largest disk is on the intermediate tower where it was stacked so that the largest disk could move, and all other disks are on the tower that will allow the second-largest disk to jump onto the largest disk. Hence, we expect 2^4 moves to have transpired to move the largest disk, 0 additional moves for the movement of the second largest disk, and $2^2+2^1+2^0$ moves for the movement of the smaller 3 disks.

Now, we use the configuration as input to $\mathcal{C}to\mathcal{I}$:

$$\begin{aligned}
\mathcal{C}to\mathcal{I}(42111;\ (0,4)) &= 1\ \mathcal{C}to\mathcal{I}(2111;\ (2,4)) && \text{since 4 is the value of the finish tower,}\\
&= 10\ \mathcal{C}to\mathcal{I}(333;\ (2,3)) && \text{since 2 is the value of the start tower ,}\\
&= 101\ \mathcal{C}to\mathcal{I}(33;\ (0,3))\\
&= 1011\ \mathcal{C}to\mathcal{I}(3;\ (4,3))\\
&= 10111.
\end{aligned}$$

Hence, the output is the binary number $10111 = 2^4 + 2^2 + 2^1 + 2^0$, as expected.

Now, we note that if we choose an input configuration not within the sequence, the output is an error. Consider the configuration 00333. Here, we have the three smaller disks stacked on tower 3, where by the game rules they must be stacked on tower 2 for the largest disk to be moved onto tower 4. Since getting to this configuration diverges from the shortest sequence of configurations, we expect to get an error.

When we use the configuration as input to $\mathcal{C}to\mathcal{I}$,

$$\begin{aligned}
\mathcal{C}to\mathcal{I}(00333;\ (0,4)) &= 0\ \mathcal{C}to\mathcal{I}(0333;\ (0,2))\\
&= 00\ \mathcal{C}to\mathcal{I}(333;\ (0,1))\\
&= 00\ \ \mathcal{C}to\mathcal{I}(33;\ (Error)) && \text{since 3 is neither the value of}\\
& && \text{the start nor finish tower.}
\end{aligned}$$

Hence, we get an error, as we hoped.

Now, after gaining some intuition as to how $\mathcal{C}to\mathcal{I}$ works, we prove its correctness.

Proposition 3.8. *The FST $\mathcal{C}to\mathcal{I}$ described in Construction 3.5 takes the input configuration $x_{n-1}x_{n-2}\dots x_0$ to its index in the solution sequence in a game of GTH with d towers when moving from tower a to tower b.*

Proof. We proceed by induction on k, the length of the input string.

Base Case. Let $k = 1$. Then our input string is $x_0 \in \{0,\dots,d-1\}$. We then see that as defined in Construction 3.5,

$$\mathcal{C}to\mathcal{I}(x_0;\ (a \to b)) = \begin{cases} 0 & : x_0 = a,\\ 1 & : x_0 = b,\\ \varepsilon & : \text{otherwise.} \end{cases}$$

Then we conclude that $\mathcal{C}to\mathcal{I}$ gives the correct output when $k = 1$, since the solution sequence for a single disk is simply a, b, and $\mathcal{C}to\mathcal{I}$ assigns a and b to their proper positions in the solution sequence and all other values of x_0 to the empty string, as desired.

Inductive Step. Suppose that the result holds for k and consider $k+1$. Then we have input string $x_k x_{k-1}\dots x_0$, and say we wish to move from tower a to tower b for $a, b \in \{0,\dots,d-1\}$. Now, as defined in Construction 3.5,

$$\mathcal{C}to\mathcal{I}\left(x_k x_{k-1}\dots x_0;\ (a,b)\right) = \begin{cases} 0\ \mathcal{C}to\mathcal{I}\left(x_{k-1}\dots x_0;\ \left(a, \frac{a+b}{2}\right)\right) & : x_k = a,\\ 1\ \mathcal{C}to\mathcal{I}\left(x_{k-1}\dots x_0;\ \left(\frac{a+b}{2}, b\right)\right) & : x_k = b,\\ \quad \mathcal{C}to\mathcal{I}\left(x_{k-1}\dots x_0;\ (Error)\right) & : \text{otherwise.} \end{cases}$$

First, we realize that if $x_k = a$, then the largest disk must still be on the first tower, and so the configuration must be in the first half of the solution sequence, hence the output of 0 is correct. Furthermore, by the recursive algorithm for solving GTH [8], we realize that in this case the k

smaller disks are in the process of being stacked on tower $\frac{a+b}{2}$, hence by the inductive hypothesis $\mathcal{C}to\mathcal{I}\left(x_{k-1}\ldots x_0;\ \left(a,\frac{a+b}{2}\right)\right)$ will give the correct output.

Similarly, if $x_k = b$, then the largest disk must have been moved to the last tower, and so the configuration must be in the second half of the solution sequence, hence the output of 1 is correct. Again by the recursive algorithm for GTH [8], we realize that in this case the k smaller disks are in the process of being moved from tower $\frac{a+b}{2}$ to tower b, hence by the inductive hypothesis $\mathcal{C}to\mathcal{I}\left(x_{k-1}\ldots x_0;\ \left(\frac{a+b}{2},b\right)\right)$ will give the correct output.

Finally, if $a \neq x_k \neq b$, then the largest disk is on a tower that it should never be on in the solution sequence, since it is redundant; $\mathcal{C}to\mathcal{I}$ correctly enters the error state, as desired. □

Proposition 3.9. *In the FST described Construction 3.5, there are no redundant states.*

Proof. We show that no two states are equivalent. Consider state (i,j). Let ε stand for an empty output. Then we note that for state (k,ℓ) where $k \neq i$ and $j \neq \ell$, we can differentiate between (i,j) and (k,ℓ) with the input ℓ:

$$\mathcal{C}to\mathcal{I}\big(\ell,(i,j)\big)\big) = \varepsilon \neq 1 = \mathcal{C}to\mathcal{I}\big(\ell,(k,\ell)\big).$$

To differentiate between state (i,j) and state (i,ℓ) with $\ell \neq j$, note that it is impossible that $\ell = i$, hence

$$\mathcal{C}to\mathcal{I}\big(j,(i,j)\big) = 1 \neq \varepsilon = \mathcal{C}to\mathcal{I}\big(j,(i,\ell)\big).$$

Similarly, to differentiate between state (i,j) and state (k,j) with $i \neq k$, realize that

$$\mathcal{C}to\mathcal{I}\big(i,(i,j)\big) = 0 \neq \varepsilon = \mathcal{C}to\mathcal{I}\big(i,(k,j)\big).$$

Therefore, for each pair of non-trash states there exists an input that will produce different outputs. The $(Error)$ state case is trivial. We conclude that this Finite State Transducer is minimal. □

Corollary 3.10. *The FST for d towers and n disks will have exactly $d \cdot |2|_d + 1$ states.*

Proof. This follows directly from Propositions 3.8 and 3.9. □

3.3. **Corner-to-Corner Configuration-to-Index FST for Generalized Spin-Out.** The traditional Spin-Out puzzle requires the player to move from configuration $aaa\ldots a$ to configuration $000\ldots 0$, with $a \in \{1,\ldots,2^m-1\}$. However, we can reconsider the object of the game to be movement from configuration $a00\ldots 0$ to configuration $000\ldots 0$ and vice versa. This corresponds to movement between corners on the iterated complete graph $K_{2^m}^n$, and so it is natural to ask how one moves between these two configurations.

As noted in [11], the solution sequence for this puzzle is exactly analogous to binary reflected Gray code, and so we can modify the FST that converts Gray to Binary very slightly in order to obtain the FST we want.

We describe a FST that maps a configuration to the index in the solution sequence in the corner-to-corner case.

Construction 3.11. We construct an FST $\mathcal{C}to\mathcal{I}_s$ that takes as input a configuration of Generalized Spin-Out with m spinners per stack and returns the index of that configuration in the solution sequence from $000\ldots 0$ to $a00\ldots 0$ or $a00\ldots 0$ to $000\ldots 0$ for some fixed $a \in \{1,\ldots,2^m-1\}$. The construction is as follows:

(1) Let the input string be a configuration $x_{n-1}x_{n-2}\ldots x_0$ where each $x_i \in \{0,\ldots,2^m-1\}$.

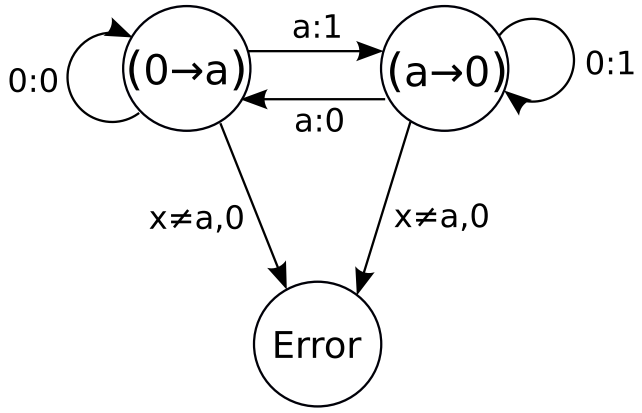


FIGURE 3.2. This is the FST $\mathcal{C}to\,I_s$ described in Construction 3.11. It takes a configuration from Spin-Out to its index in the solution sequence going from $a00\ldots0$ to $000\ldots0$ if started in state $(a,0)$ and to the index in the solution sequence going from $000\ldots0$ to $a00\ldots0$ if started in state $(0,a)$

.

(2) Create the set of states

$$S = \{(a \to 0), (0 \to a), (Error)\}.$$

(3) Set the start state to $(0 \to a)$ if moving from configuration $000\ldots0$ to configuration $a00\ldots0$; set the start state to $(a \to 0)$ if moving from configuration $a00\ldots0$ to configuration $000\ldots0$.
(4) Let $\mathcal{C}to\,I_s(x_{n-1}x_{n-2}\ldots x_0;\ s)$ be the output of the machine when $x_{n-1}x_{n-2}\ldots x_0$ is the input string and $s \in S$ is the machine's current state.
(5) Let $i, j \in \{a,0\}$ and $i \neq j$. We define the transitions of $\mathcal{C}to\,I_s$ as follows:

$$\begin{aligned}
\mathcal{C}to\,I_s\,(x_{n-1}x_{n-2}\ldots x_0;\ (i \to j)) &= \begin{cases} \frac{1}{a}i\ \mathcal{C}to\,I_s(x_{n-2}\ldots x_0;\ (i \to j)) & : x_{n-1} = 0, \\ \frac{1}{a}j\ \mathcal{C}to\,I_s(x_{n-2}\ldots x_0;\ (j \to i)) & : x_{n-1} = a, \\ \mathcal{C}to\,I_s\,(x_{n-2}\ldots x_0;\ (Error)) & : \text{otherwise.} \end{cases} \\
\mathcal{C}to\,I_s\,(x_{n-1}x_{n-2}\ldots x_0;\ (Error)) &= \mathcal{C}to\,I_s\,(x_{n-2}\ldots x_0;\ (Error)).
\end{aligned}$$

The FST $\mathcal{C}to\,I_s$ is shown in Figure 3.2.

Remark 3.12. We do not offer proof of the correctness of the machine, as the process is directly analogous to binary reflected Gray code [11], and as we know the Binary to Gray machine given by Figure 2.3 and Equation 2.1 is correct and a self-inverse, $\mathcal{C}to\,I_s$ must also be correct.

Proposition 3.13. *The FST $\mathcal{C}to\,I_s$ described in 3.11 is minimal.*

Proof. Note that the outputs have to be as they are, since there is only one solution sequence. Hence, We consider the input a to the states $(a \to 0)$, $(0 \to a)$, and $(Error)$ and note that

$$\mathcal{C}to\,I_s(a,(a \to 0)) = 0 \neq 1 = \mathcal{C}to\,I_s(a,(0 \to a)),$$

$$\mathcal{C}to\,I_s(a,(a \to 0)) = 0 \neq \varepsilon = \mathcal{C}to\,I_s(a,(Error)) = \varepsilon \neq 1 = \mathcal{C}to\,I_s(a,(0 \to a)).$$

Hence, since the outputs are different, the FST is minimal. □

3.4. **FST for Combined Puzzle.** Now in possession of FST's for both Corner-to-Corner Spin-Out and for GTH, we can define a FST for the Combined puzzle if traveling from corner to corner in K_d^n for $d = 2^m \cdot q$, where $m \in \mathbb{N}$ and q odd.

Construction 3.14. We construct a Finite State Transducer $\mathcal{C}to\mathcal{I}$ that takes as input a configuration string in d-ary and returns the configuration's binary index number in the sequence of winning moves in the Combined puzzle when moving from the configuration $aa\ldots a$ to the configuration $bcc\ldots c$ where $c = b - \ell = 2^m \cdot z$ for some $\ell \in \{0,\ldots,2^m-1\}$, $z \in \{0,\ldots,q-1\}$. Note that ℓ is the ending spinner configuration for the highest-order spinner.

(1) Let the input be a position string $x_{n-1}x_{n-2}\ldots x_0$, where x_i corresponds to the configuration of the i^{th} largest piece; thus $x_i = 2^m \cdot t + r$ for some $t \in \{0,1,\ldots,q-1\}$ and $r \in \{0,\ldots,2^m - 1\}$.

(2) Create the set of states

$$S = \left\{ \big(i, i+2^k(c-a), dir\big) \;:\; i \in \{0,\ldots,d-1\}\,,\; k \in \{1,\ldots,|2|_d\}, dir \in \{orig, rev\} \right\} \cup \big\{(Error)\big\}.$$

(3) Set the input state $s_0 = (a, c, orig)$, where a is the starting tower and c is the ending tower.

(4) Let $\mathcal{C}to\mathcal{I}(x_{n-1}x_{n-2}\ldots x_0;\ (a,c,dir))$ be the output of the machine when $x_{n-1}\ldots x_0$ is the input string and the machine starts in state (a,c,dir).

(5) Let $i, j \in \{0,\ldots,d-1\}$, and let $x_{n-1} = 2^m \cdot t + r$. Define the transitions of $\mathcal{C}to\mathcal{I}$ as follows:

$$\mathcal{C}to\mathcal{I}\,(x_{n-1}x_{n-2}\ldots x_0;\ (i,j,orig)) = \begin{cases} 0\ \mathcal{C}to\mathcal{I}\left(x_{n-2}\ldots x_0;\ \left(i, \frac{i+j}{2}, orig\right)\right) & : t = i \text{ and } r = 0 \\ 1\ \mathcal{C}to\mathcal{I}\left(x_{n-2}\ldots x_0;\ \left(\frac{i+j}{2}, j, rev\right)\right) & : t = j \text{ and } r = \ell \\ \quad \mathcal{C}to\mathcal{I}\,(x_{n-2}\ldots x_0;\ (Error)) & : \text{otherwise.} \end{cases}$$

$$\mathcal{C}to\mathcal{I}\,(x_{n-1}x_{n-2}\ldots x_0;\ (i,j,rev)) = \begin{cases} 0\ \mathcal{C}to\mathcal{I}\left(x_{n-2}\ldots x_0;\ \left(i, \frac{i+j}{2}, orig\right)\right) & : t = i \text{ and } r = \ell \\ 1\ \mathcal{C}to\mathcal{I}\left(x_{n-2}\ldots x_0;\ \left(\frac{i+j}{2}, j, rev\right)\right) & : t = j \text{ and } r = 0 \\ \quad \mathcal{C}to\mathcal{I}\,(x_{n-2}\ldots x_0;\ (Error)) & : \text{otherwise.} \end{cases}$$

$$\mathcal{C}to\mathcal{I}\,(x_{n-1}x_{n-2}\ldots x_0;\ (s_{err}, s_{err})) = \mathcal{C}to\mathcal{I}\,(x_{n-2}\ldots x_0;\ (Error))$$

Proposition 3.15. *The FST $\mathcal{C}to\mathcal{I}$ defined in Construction 3.14 correctly maps a configuration of the Combination Puzzle to its index in the solution sequence moving from corner $aaa\ldots a$ to corner $bcc\ldots c$ where $c = b - k$ for some $k \in \{0,\ldots,2^m-1\}$.*

Proof. The proof is by induction on the length of the input string, and follows a very similar format to the proof of Proposition 3.8, relying on the recursive algorithms set forth in [8]. □

We now show some examples of $\mathcal{C}to\mathcal{I}$'s correct operations on inputs from the Combination Puzzle with $d = 5$.

Example 3.16. Let $d = 10$, $n = 4$, and consider the movement from configuration 2222 to configuration 7666. Then the disks start all stacked on tower 1 and are transferred to tower 3, while the spinners all start unlocked in position 0 and only the last disk is transferred to locked position 1.

Then consider the configuration 7544. We expect this configuration will be number 1000, since the spinners are in the configuration directly after the highest-order spin was changed, and the largest disk is on the destination tower with all other towers on tower $\frac{1+3}{2}$. We begin the machine in state $(0,3,orig)$ since we wish to move from tower 0 to tower 3. Then the conversion is as follows:

$$\begin{aligned}
CtoI(7544;(1,3,orig)) &= 1\ CtoI(544;(2,3,rev)) && \text{since } 7 = 2^1\cdot 3+1,\\
&= 10\ CtoI(44;(2,0,orig)) && \text{since } 5 = 2^1\cdot 2+1,\\
&= 100\ CtoI(4;(2,1,orig)) && \text{since } 4 = 2^1\cdot 2+0,\\
&= 1000 && \text{since } 4 = 2^1\cdot 2+0.
\end{aligned}$$

Hence, the outcome is as we expected.

Now, we note that for input 7444, we will get an error:

$$\begin{aligned}
CtoI(7444;(1,3,orig)) &= 1\ CtoI(444;(2,3,rev)) && \text{since } 7 = 2^1\cdot 3+1,\\
&= 1\ CtoI(44;(Error)) && \text{since } 4 = 2^1\cdot 2+0,
\end{aligned}$$

and so the transition is into the error state. This is comforting, since this configuration indicates that the spinners are in the desired final positions while the disks lag behind.

Proposition 3.17. *The FST $CtoI$ defined in Construction 3.14 is minimal.*

Proof. Consider the combined puzzle for $d = 2^m\cdot q$ which transitions from $a\cdots a$ to $b(b-\ell)\cdots(b-\ell)$. By Lemma 3.2, we realize that at most the states $(i, i+2^k(b-\ell-a), dir)$ can exist for each $k\in\{0,\ldots,|2|_q\}$, $i\in\{0,\ldots,q\}$. These are exactly the states of $CtoI$. We note first that the solution sequence is unique, hence there is no flexibility in the output of the states. By Now, we note that for $i, j\in\{0,\ldots,q\}$ with $i\neq j$, if we have input $x = 2^m\cdot i+0$ we have

$$CtoI\left(2^m\cdot i+0;\ (i,j,orig)\right) = 0 \neq \varepsilon = CtoI\left(2^m\cdot i+0;\ (j,i,orig)\right),$$

$$CtoI\left(2^m\cdot i+0;\ (i,j,orig)\right) = 0 \neq \varepsilon = CI\left(2^m\cdot i+0;\ (i,j,rev)\right),$$

$$CtoI\left(2^m\cdot i+0;\ (i,j,orig)\right) = 0 \neq 1 = CI\left(2^m\cdot i+0;\ (j,i,rev)\right),$$

so the state $(i,j,orig)$ is distinct from all other states.

Now, note that for input $y = 2^m\cdot i+\ell$,

$$CtoI\left(2^m\cdot i+\ell;\ (j,i,orig)\right) = 1 \neq 0 = CI\left(2^m\cdot i+\ell;\ (i,j,rev)\right),$$

$$CtoI\left(2^m\cdot i+\ell;\ (j,i,orig)\right) = 1 \neq \varepsilon = CI\left(2^m\cdot i+\ell;\ (j,i,rev)\right),$$

so the state $(j,i,orig)$ is distinct from all other states.

Finally, note that for input y,

$$CI\left(2^m\cdot i+\ell;\ (i,j,rev)\right) = 0 \neq \varepsilon = CI\left(2^m\cdot i+\ell;\ (j,i,rev)\right),$$

and so the states (i,j,rev) and (j,i,rev) are distinct from all other states.

Hence, we conclude that $CtoI$ is minimal. □

3.5. **Lack of Existence of Configuration to Index FST for Generalized Spin-Out.** We would wish to define a similar Configuration-to-Index FST for the traditional Spin-Out, that is Spin-Out where one begins in state $aa\ldots a$ and ends in state $00\ldots 0$. However, in this subsection we prove that it is impossible to define such an FST.

Definition 3.18. Let $GtoB$ be the function that takes a string in binary reflected Gray code to the binary number corresponding to its index in the sequence of Gray strings.

Definition 3.19. Let $BtoG$ be the inverse of $GtoB$.

Lemma 3.20. *Suppose f is a function that takes a Generalized Spin-Out configuration c to its binary index b in the sequence of winning moves. Then if c is actually in the solution sequence from configuration $aa\dots a$ to configuration $00\dots 0$, then $f(c) = \lceil \frac{2}{3}(2^n-1)\rceil - GtoB\left(\frac{c}{a}\right)$.*

Proof. Note that the configurations in the solution sequence of Spin-Out are composed of strings that only have bits in the set $\{a, 0\}$; otherwise the configuration is not in the shortest winning sequence. Furthermore, if these strings are divided by a and viewed as binary strings, they correspond directly to the binary reflected Gray code [11], with $000\dots 0$ being the 0th number in the grey code and $111\dots 1$ being the $\lceil \frac{2}{3}(2^n-1)\rceil$th number.

In Generalized Spin-Out, the object of the game is to move from the configuration $aaa\dots a$ to $000\dots 0$, so to find a configuration c's index in the sequence of winning Spin-Out moves it suffices to subtract $\lceil \frac{2}{3}(2^n-1)\rceil - GtoB\left(\frac{c}{a}\right)$. □

Theorem 3.21. *Computing a function f that takes a configuration c in the Generalized Spin-Out solution sequence going from configuration $aa\dots a$ to $00\dots 0$ to its binary index b in the solution sequence requires linear space.*

Proof. Suppose by way of contradiction that there exists an FST $\mathcal{F}$ that takes as input c, a Generalized Spin-Out configuration in the solution sequence from configuration $aa\dots a$ to $00\dots 0$, and returns b, the binary number corresponding to the index of c in the sequence of winning moves. Note that $\mathcal{F}$ cannot read input from the lower-order end, since the strings $aa0\dots 0$ and $000\dots 0$ must have distinct outputs:

$$\mathcal{F}(aa0\dots 0) = \begin{cases} 00(10)^{\frac{n-2}{2}} & : n \equiv 0 \bmod 2 \\ 0(01)^{\frac{n-1}{2}} & : n \equiv 1 \bmod 2 \end{cases}, \qquad \mathcal{F}(000\dots 0) = 11\cdots 1.$$

(For example, In Generalized Spin-Out with $d = 2^2$, going from 3333 to 0000 we have $\mathcal{F}(3300) = 0010$ while $\mathcal{F}(0000) = 1111$).

Since the first $n-2$ bits of the input strings are identical, but the two lower-order output bits of the two strings are always different, $\mathcal{F}$ cannot distinguish between the two input strings with a finite number of states when reading from the lower-order end. Hence, $\mathcal{F}$ must read inputs from the higher-order end.

Now, suppose we wish to subtract some binary number d from $\lceil \frac{2}{3}(2^n-1)\rceil - GtoB(\frac{c}{a})$. We have an FST $BtoG$ that reads from the higher-order end and takes any number in binary to its corresponding binary reflected Gray code string, as described by Figure 2.3 and Equation 2.1. Hence, we can simply calculate $\mathcal{F}(BtoG(d))$, and by the previous lemma we see that

$$\mathcal{F}(BtoG(d)\cdot a) \equiv \lceil \frac{2}{3}(2^n-1)\rceil - GtoB\left(\frac{BtoG(d)\cdot a}{a}\right) = \lceil \frac{2}{3}(2^n-1)\rceil - d,$$

as desired. Note that this computation occurred in linear time and constant memory, since it was the composition of two finite state machines reading from the higher order end.

However, now for even n we can choose $d_1 = 0010\dots 1010$ and $d_2 = 0010\dots 1011$ and subtract $\lceil \frac{2}{3}(2^n-1)\rceil - d_1$ and $\lceil \frac{2}{3}(2^n-1)\rceil - d_2$ without requiring memory for a carry chain of size $\approx n$, a contradiction. A similar contradiction can be obtained by choosing $d_1' = 0010\dots 10111$ and $d_2' = 0010\dots 10100$ for odd n. Hence, we conclude that no FST can compute the index of a configuration of Generalized Spin-Out, as desired. □

4. Hamiltonian Path Existence and Uniqueness

Definition 4.1. A *Hamiltonian path* is a path in an undirected graph that visits each vertex exactly once.

For a general graph G, deciding whether G has a Hamiltonian path is an NP-Complete problem. In the introduction, we defined a class of graphs, called iterated complete graphs, which correspond directly to the allowed moves in our puzzles. That is, the vertices represent game configurations, and an edge between two vertices is present if and only if it is possible to transition between the two configurations in a single move.

In 2009 Baun and Chauhan showed that iterated complete graphs K_d^n always have Hamiltonian paths, and that these paths are unique for $d = 2, 3$ [1]. In this section we will review some of these results, and show that uniqueness does not hold for $d > 3$.

4.1. Hamiltonian Path Existence for Iterated Complete Graphs.

Theorem 4.2. *There exists some Hamiltonian path between two corner vertices on K_d^n for all $d \in \mathbb{N}$.*

The Hamiltonian path is constructed recursively. Clearly, there exists a Hamiltonian path between any two vertices in K_d^1, since it is complete. Then, in the K_d^2 level, we can take this path and rotate it, then fill in the path between the K_d^1 subcopies with edges (see Figure 4.1). In this way, the Hamiltonian path on K_d^{n+1} can be constructed using that on K_d^n. For a more detailed, rigorous proof of this theorem, see Baun and Chauhan [1]. We offer a pictorial outline of the proof in Figure 4.1.

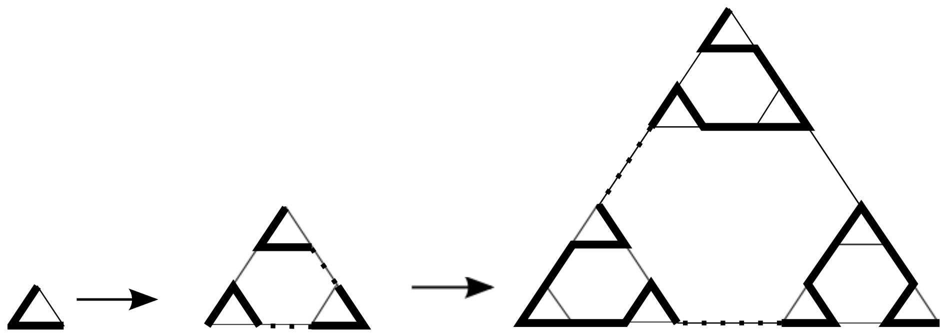


FIGURE 4.1. We can construct a Hamiltonian path on K_d^n recursively, by first constructing one on K_d^{n-1} and then rotating it on the subcopies and connecting the edges.

Note also that as a corollary of Theorem 4.2, a Hamiltonian circuit must also exist on K_d^n, since in the case of $n = 1$ the graph is complete and for $n > 1$ a circuit may be constructed using the paths on K_d^{n-1} (see Figure 4.2).

4.2. Uniqueness for $d = 2, 3$.

Theorem 4.3. *There is an unique Hamiltonian path between two corner vertices in the iterated complete graph K_2^n.*

Proof. This is obvious from the construction of the graph. $\square$

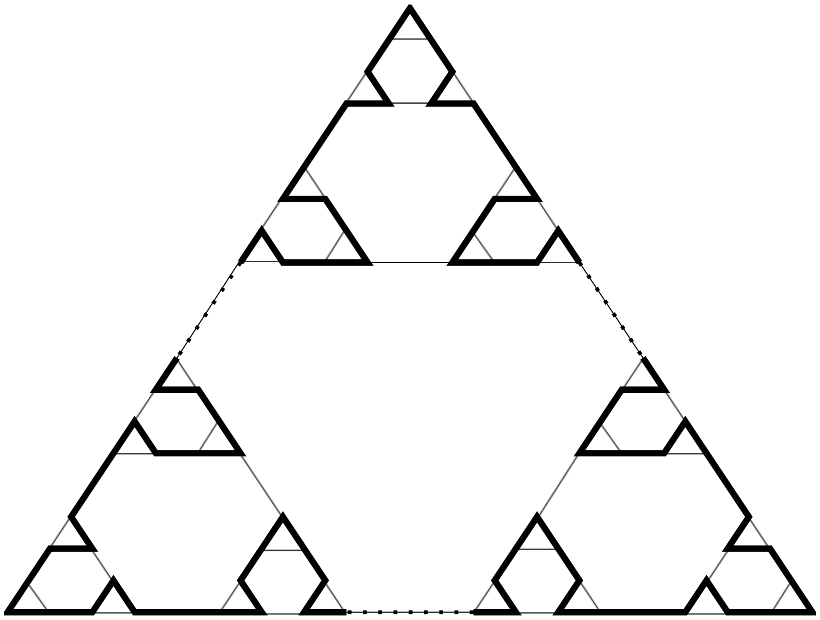


FIGURE 4.2. A Hamiltonian circuit on K_d^n can also be constructed from rotated copies of the path on K_d^{n-1}.

The following theorem and proof are based on work done by Leanne Merrill and Tony Van in 2010 [8].

Theorem 4.4. *There is an unique Hamiltonian path between any two corner vertices in the iterated complete graph K_3^n.*

Proof. Proof by induction.
Base case: Let $n = 1$. On the graph K_3^1, the complete graph on three vertices, choose a starting vertex. Whichever vertex you choose to be the endpoint, you must travel to the remaining vertex first and then to your destination. Therefore, there is a unique Hamiltonian path on K_3^1.
Inductive Hypothesis: Assume there exists a unique Hamiltonian path between any two corner vertices of K_3^{n-1}.

Consider the three K_3^{n-1}-subcopies of K_3^n and label them A, B and C. We can assume that the Hamiltonian path of K_3^n will travel from the corner of A to the corner of C without loss of generality. By the definition of iterated complete graphs, we know that A, B and C are connected by three edges: A — B, A — C, and B — C. Since each DK_3^{n-1}-subcopy only has two external edges, then C, the K_3^{n-1}-subcopy with the target vertex, must be visited last. By our inductive hypothesis, we know that there exist unique Hamiltonian paths in each of the subgraphs A, B and C between any two corner vertices of the subgraph. Start at the corner vertex of K_3^n that is also a corner vertex of A. The endpoint of the Hamiltonian path on A must be part of the edge A — B. So we travel the unique Hamiltonian path from the corner of A to the corner of A that is part of A — B. Then we enter B and travel the unique Hamiltonian path from the corner we enter to the corner that is part of B — C since there is no other way to exit B. Finally, we enter C and travel the unique Hamiltonian path to the corner vertex of C that is our endpoint. Since all the Hamiltonian paths we traveled are unique as well as the edges used to travel between subgraphs, the path on K_3^n between any corner vertices is unique. $\square$

Figure 4.3 shows the unique Hamiltonian path on K_3^3.

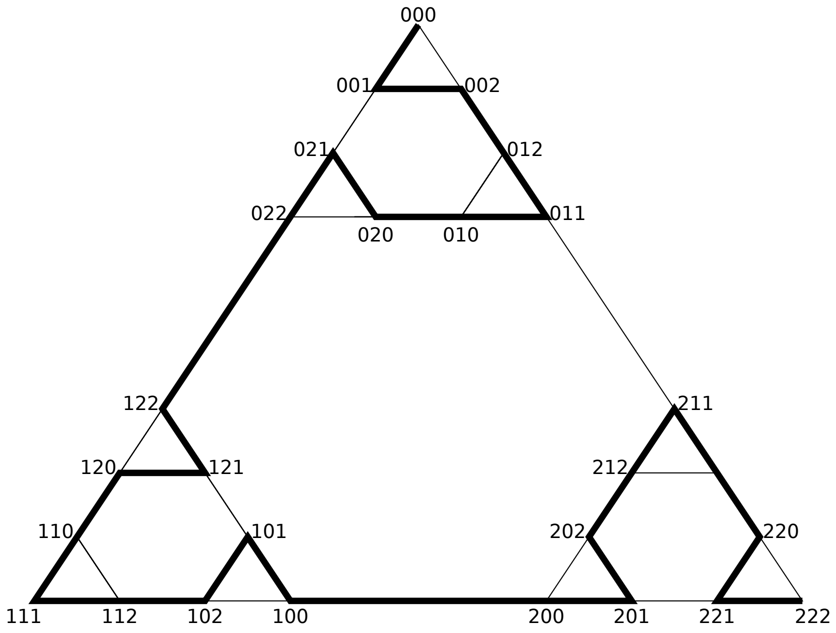


FIGURE 4.3. The unique Hamiltonian path on the graph K_3^3 from 000 to 222 is outlined.

4.3. **Non-uniqueness of Hamiltonian Paths for $d > 3$.** For convenience, we first define some notation.

Definition 4.5. Let $\mathcal{H}_p(G,x,y)$ be a Hamiltonian path on graph G, where the path starts at vertex x and ends at vertex y.

We use this notation when the graph G is clearly Hamiltonian (e.g. when G is complete).

Theorem 4.6. *Hamiltonian paths between two corner vertices in K_d^2 are not unique for $d > 3$.*

Proof. Consider the graph K_d^n. We will construct two distinct Hamiltonian paths on K_d^n that start at vertex u and end at vertex v.

We construct the first Hamiltonian Path, H_1, recursively, as described in Theorem 4.2. See figure 4.4 for an example of H_1 on K_5^2.

We now construct a second Hamiltonian path H_2. First, label the K_d subcopies $C_1,\dots,C_d$ based on the order in which the copies are visited in H_1, with $u \in C_1$ and $v \in C_d$.

Since H_1 travels from C_1 to C_2 via a single edge, there must be some vertex y_2 in C_2 that is adjacent to a vertex x_1 in C_1. Because C_1 is a complete graph the edge u–x_1 exists. We let this be the first edge in H_2. Now, we exit C_1 early, departing from the path topology of H_1, by letting the second edge in H_2 be x_1–y_2 (see figure 4.4 for an example). We then traverse copies $C_2,\dots,C_{d-2}$ exactly as in H_1, and so all vertices in $C_2,\dots,C_{d-2}$ are visited.

Now, we travel from C_{d-2} to C_{d-1} by the same edge e used in H_1 (see figure 4.4), and we let y_{d-1} be e's endpoint in C_{d-1}. Now, by the structure of K_d^2 there must be some vertex $x_{d-1} \in C_{d-1}$ such that x_{d-1} is adjacent to a vertex $y_1 \in C_1$. Furthermore, since $\deg(x_1) = d$, x_1 can only be adjacent to one vertex outside of C_1, hence $y_1 \neq x_1$ and y_1 has not been visited ($\star$). Then we add $\mathcal{H}_p(C_{d-1}, y_{d-1}, x_{d-1})$ to H_2, so that all vertices in C_{d-1} are visited. We also add the edge x_{d-1}–y_1.

Now, by an argument similar to that in ($\star$), there must exist some vertex $z_1 \in C_1$ such that z_1 is adjacent to some vertex $y_d \in C_d$ and hence z_1 has not yet been visited. Because C_1 is a complete graph, we can append $\mathcal{H}_p(C_1 - u - x_1, y_1, z_1)$ to H_2, and thus every vertex in C_1 is visited. Finally, we add the edge $z_1 y_d$ and $\mathcal{H}_p(C_d, y_d, v)$ to get a second Hamiltonian path H_2 on K_d^2.

Clearly, the distance between the corner vertices of $k \in C_{d-1}$ and $v \in C_d$ is different in H_1 and H_2; H_1 and H_2 use the same number of vertices from C_{d-1} and C_1 on $P(k,v)$, but H_2 also uses an extra $d-2$ vertices from C_1, which H_1 does not. Hence, H_1 and H_2 are not isomorphic, as desired. □

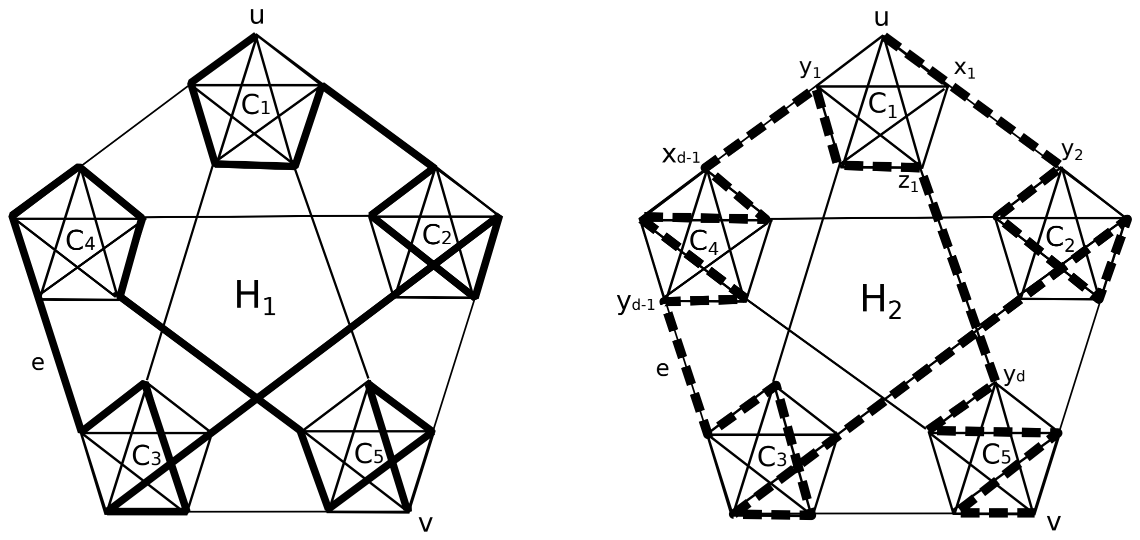


FIGURE 4.4. An example of two non-isomorphic Hamiltonian paths on K_5^2, as obtained by the construction in Theorem 4.6.

Theorem 4.7. *Hamiltonian paths between any two corner vertices u,v in K_d^n are not unique for $n > 1$, $d > 3$.*

Proof. We first construct two copies of the Hamiltonian path H as described in Theorem 4.2, and then modify one of them by swapping one of the K_d^2-subcopy's Hamiltonian path segments with the path H_2 described above. Since the subpaths are not isomorphic, the paths are not isomorphic, as desired. □

5. HAMILTONIAN PATH FST

In the previous section, we verified the existence of Hamiltonian paths on K_d^n for all $d,n \in \mathbb{N}$. In this section, we describe Finite State Transducers that give Hamiltonian paths on K_d^n for all $d,n \in \mathbb{N}$; that is, the FST's map a vertex's label to the vertex's index along a Hamiltonian path.

5.1. **Combined Puzzle.** We begin by describing a Hamiltonian path FST for K_d^n that works for any $d \in \mathbb{N}$, using the rules of the Combined Puzzle. We know that $d = 2^m \cdot q$ for some $m \in \mathbb{N}$ and odd q, so creating an FST where $m = 0$ gives an FST for the odd case, or GTH, while creating an FST where $q = 1$ gives and FST for the powers of two case, or Spin-Out.

Construction 5.1. We construct a Finite State Transducer $\mathcal{H}$ that takes as input a vertex's label in K_d^n, $d = 2^m \cdot q$ for $m \in \mathbb{N}$, q odd, and returns its position along a Hamiltonian path between the corner with leading digit a and the corner with leading digit b, with $a, b \in \{0, \dots, d-1\}$ and $a \neq b$.

The construction is as follows:

(1) Let the input string be $x_{n-1}x_{n-2}\dots x_0$, with each $x_i \in \{0, \dots, d-1\}$.
(2) Let the output $\mathcal{H}(x_{n-1}x_{n-2}\dots x_0;\ (a \to b))$ be the index in d-ary of the vertex labeled $x_{n-1}\dots x_0$ in the Hamiltonian path if $\mathcal{H}$ is started in state $(a \to b)$.
(3) The machine will have $d(d-1)$ states, corresponding to all ordered pairs of distinct integers in $\{0, \dots, d-1\}$. Let S be the set of states, then

$$S = \{(a \to b)\ :\ a, b \in \{0, \dots, d-1\},\ a \neq b\}$$

The state $(a \to b)$ represents a case in which one wishes to travel from a corner of the graph with leading digit $a = 2^m \cdot t_a + r_a$ to a corner of the graph with leading digit $b = 2^m \cdot t_b + r_b$.
(4) For each state $(a \to b)$, let the map $INDEX_{a \to b} : \mathbb{Z}_d \to \mathbb{Z}_d$ designate the order in which the subcopies are traveled. We define $INDEX_{a \to b}$ as follows: if $a < b$, then

$$INDEX_{a\to b}(x) \quad = \quad \begin{cases} 0 & : x = a \\ x+1 & : x < a \\ x & : a < x < b \\ x-1 & : x > b \\ d-1 & : x = b \end{cases}$$

Otherwise, if $a > b$ then we define $INDEX_{a \to b}$ as:

$$INDEX_{a\to b}(x) \quad = \quad \begin{cases} 0 & : x = a \\ -2-x & : x < b \\ -1-x & : b < x < a \\ -x & : x > b \\ d-1 & : x = b \end{cases}$$

We considered very few constraints when defining this function. Clearly, we wanted to map $INDEX_{a\to b}(a) = 0$ and $INDEX_{a\to b}(b) = d-1$. We also wanted $INDEX_{a\to b}$ to be 1-1 and onto. Finally, we decided that the relationship $INDEX_{a\to b}(x) = d-1-INDEX_{b\to a}(x)$ should hold, in order to make our path look more symmetric.

For later convenience, we also define the functions

$$PREV_{a\to b}(x) = \begin{cases} INDEX^{-1}_{a\to b}\Big(INDEX_{a\to b}(x)-1\Big) & : INDEX_{a\to b}(x) \geq 1 \\ a & : \text{otherwise.} \end{cases}$$

and

$$NEXT_{a\to b}(x) = \begin{cases} INDEX^{-1}_{a\to b}\Big(INDEX_{a\to b}(x)+1\Big) & : INDEX_{a\to b}(x) \leq d-2 \\ b & : \text{otherwise.} \end{cases}$$

We note that these functions give the leading digit of the labels in previous and next subcopy of the graph traversed.

(5) We define $\mathcal{H}$'s transitions as follows:

$$\mathcal{H}(x_{n-1}x_{n-2}\ldots x_0;\ (a\to b)) = INDEX_{a\to b}(x_{n-1})\mathcal{H}\left(x_{n-2}\ldots x_0;\ (a'\to b')\right)$$

where a' and b' are defined so that if we let

$$\begin{aligned} x_{n-1} &= 2^m\cdot t_x + r_x, \\ PREV_{a\to b}(x_{n-1}) &= 2^m\cdot t_{prev} + r_{prev}, \\ NEXT_{a\to b}(x_{n-1}) &= 2^m\cdot t_{next} + r_{next}, \end{aligned}$$

then $a' = 2^m\cdot t_{a'} + r_{a'}$ such that

$$\begin{aligned} t_{a'} &= \frac{1}{2}(t_x + t_{prev}) \bmod q, \\ r_{a'} &= r_x \ominus r_{prev}, \end{aligned}$$

and where $b' = 2^m\cdot t_{b'} + r_{b'}$ such that

$$\begin{aligned} t_{b'} &= \frac{1}{2}(t_{next} + t_x) \bmod q, \\ r_{b'} &= r_{next} \ominus r_x. \end{aligned}$$

Lemma 5.2. *Let v be a corner vertex with highest-order digit a in the graph K^n_d with $d = 2^m\cdot q$ for $m\in\mathbb{N}$, q odd. Then v has label $acc\ldots c$, where $a = 2^m\cdot t + r$ and $c = 2^m\cdot t$ for $t\in\{0,\ldots,q-1\}$ and $r\in\{0,\ldots,2^m-1\}$.*

Proof. Suppose that v is a corner of K^n_d, with $d = 2^m\cdot q$ for $m\in\mathbb{N}$, q odd. Furthermore, let the highest-order digit in v's label be some $a\in\{0,\ldots,d-1\}$. Then by Definition 2.3, $a = 2^m\cdot t + r$ for some $t\in\{0,\ldots,q-1\}$ and $r\in\{0,\ldots,2^m-1\}$.

Since v is a corner in K^n_d, it only has edges to vertices that differ in the least significant digit. Recalling that v's label represents a configuration in the Combination Puzzle, we realize that if only the least significant digit can change, then only the smallest piece's spinners and tower can change. This means that all the disks must be stacked on the same tower, and that all spins except the largest spin must be 0. Hence, the remaining digits of v must all be $c = 2^m\cdot t$. □

Proposition 5.3. *The FST $\mathcal{H}$ from Construction 5.1 maps vertex labels to positions along a Hamiltonian path from the corner vertex with leading digit a to the corner vertex with leading b in K^n_d for $d = 2^m\cdot q$ with $m\in\mathbb{N}$ and q odd.*

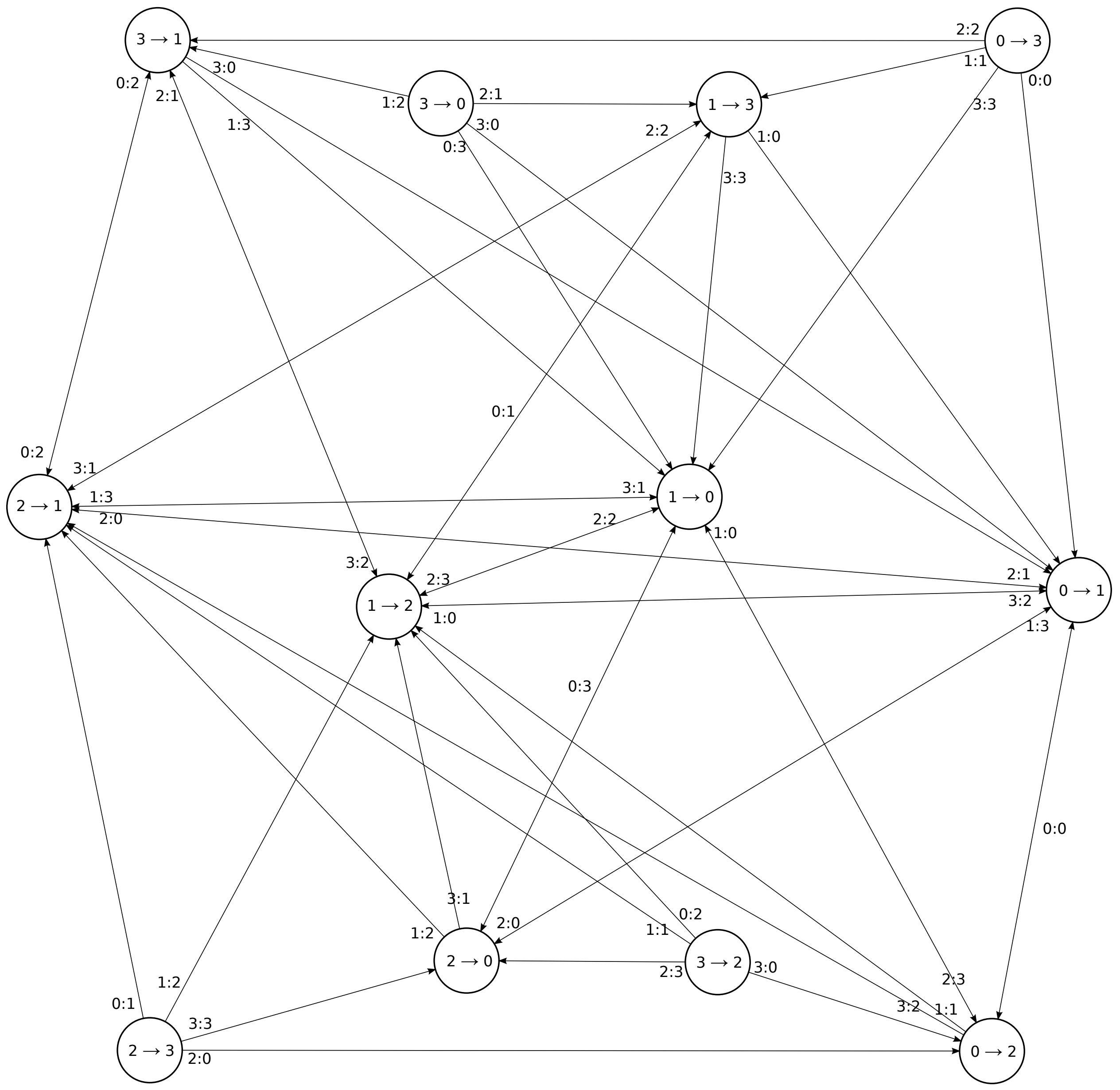


FIGURE 5.1. This if the FST $\mathcal{H}$ for $d = 4$. Note that some of the states are only used if they are start states. In other words, they are unnecessary except in very special cases.

Proof. We proceed by induction on k, the iteration of the graph.

Base Case. Let $k = 0$. Then $\mathcal{H}(x;(a \to b)) = INDEX_{a\to b}(x)$. Since $INDEX_{a\to b}$ assigns the lowest value to a and the greatest to b, the path starts with a and ends with b, as desired. Every vertex is visited because $INDEX_{a\to b}$ is bijective, and the order gives a valid path since K_d^1 is complete.

Inductive Step. Assume the statement holds for k, and consider the traversal of K_d^{k+1}. Then

$$\mathcal{H}(x_k x_{k-1} \dots x_0;\ (a \to b)) = INDEX_{a \to b}(x_k)\mathcal{H}\left(x_{k-1} \dots x_0;\ (a' \to b')\right),$$

with $a' = 2^m \cdot t_{a'} + r_{a'}$ and $b' = 2^m \cdot t_{b'} + r_{b'}$ as defined in step 5 of Construction 5.1.

Let C be the subcopy K_d^k with leading digit x_k. Since $INDEX_{a \to b}(x_k)$ corresponds to the index of C in the Hamiltonian path between K_d^k subcopies, outputting $INDEX_{a \to b}(x_k)$ will give the number of vertices already traversed in other copies of K_d^k. Hence, the leading digit output is correct.

It remains to show that $\mathcal{H}(x_{k-1} \dots x_0;\ (a' \to b'))$ will give the correct index of any vertex $v = x_{k-1} \dots x_0$ in C, the subgraph K_d^k with leading digit x_k. To this end, we must verify that the state transition is correct. That is, we must verify that the first vertex $v_{first} = x_k y_{k-1} y_{k-2} \dots y_0$ in C has $y_{k-1} = a'$, and that the last vertex $v_{last} = x_k z_{k-1} z_{k-2} \dots z_0$ in C has $z_{k-1} = b'$.

First, we note that v_{first} is either the first vertex in the path altogether, or it must be a corner vertex that allowed entry from C_{prev}, the copy traversed directly before C.

In the former case, $x_k = a$, since the path starts with the subcopy with leading digit a. Also, note that $a = 2^m \cdot t_a + r_a$ for some particular $t_a \in \{0, \dots, q-1\}$, $r_a \in \{0, \dots, 2^m - 1\}$ by Definition 2.3.Then by Lemma 5.2, since v_{first} is a corner of K_d^{k+1} the second digit of v_{first} must be $\gamma = 2^m \cdot t_a$. Hence, by the definitions of $PREV_{a \to b}$ and a' in Construction 5.1, we see that

$$\begin{aligned} t_{a'} &= \frac{1}{2}(t_x + t_{prev}) \bmod q = \frac{1}{2}(t_a + t_a) = t_a, \\ r_{a'} &= r_x \ominus r_{prev} = r_a \ominus r_a = 0. \end{aligned}$$

Thus, $a' = 2^m \cdot t_a = \gamma$, and so v_{first} is the corner of C with leading digit a'.

In the latter case, we note that C_{prev} has leading digit $PREV_{a \to b}(x_k) = 2^m \cdot t_{prev} + r_{prev}$. Hence, by the rules of the Combined Puzzle, the leading digit of v_{first} must be $y = 2^m \cdot t_y + r_y$ such that

$$\begin{aligned} 2t_y - t_{prev} &= t_x \bmod q, \\ t_y &= \frac{1}{2}(t_x + t_{prev}) \bmod q, \end{aligned}$$

and

$$\begin{aligned} r_{prev} \oplus r_y &= r_x, \\ r_y &= r_x \ominus r_{prev}. \end{aligned}$$

But then $y = a'$, and so v_{first} is the corner of C with leading a'.

Hence, a' is always the leading digit of the label on the first corner of C.

Similarly, we note that v_{last} must either be the last vertex in the entire path, or it must be a corner vertex that allows travel to C_{next}, the copy traversed directly after C.

In the former case, $x_k = b = 2^m \cdot t_b + r_b$, since the path ends with the subcopy with leading digit b. Again by Lemma 5.2, v_{last} has leading digit $\delta = 2^m \cdot t_b$ in C. Thus,

$$\begin{aligned} t_b &= \frac{1}{2}(t_{prev} + t_x) \bmod q = \frac{1}{2}(t_b + t_b) = t_b, \\ r_b &= r_{prev} \ominus r_x = r_b \ominus r_b = 0. \end{aligned}$$

Hence, we see that $b' = 2^m \cdot t_b = \delta$, and so v_{last} is the corner of C with leading digit b'.

In the latter case, we note that C_{next} has leading digit $NEXT_{a\to b}(x_m) = 2^m \cdot t_{next} + r_{next}$. Hence, by the rules of Spin-Out, the leading digit of v_{last} must be $z = 2^m \cdot t_z + r_z$ such that

$$\begin{aligned} 2t_z - t_x &= t_{next} \bmod q, \\ t_z &= \frac{1}{2}(t_{next} + t_x) \bmod q, \end{aligned}$$

and

$$\begin{aligned} r_z \oplus r_x &= r_{next}, \\ r_z &= r_{next} \ominus r_x \end{aligned}$$

But then $z = b'$, and so the corner with leading b' is the last vertex in the traversal of C.

Therefore, b' is always the leading digit of the label of the last corner of C along the path.

Hence, by the Inductive Hypothesis, $\mathcal{H}(x_{k-1}\dots x_0;\ (a' \to b'))$ will give the correct index of the label $x_{k-1}\dots x_0$ within our subcopy C of size K_d^k. Hence, the statement holds for $k+1$. □

We now have a FST that defines a Hamiltonian path on K_d^n for any $d \in \mathbb{N}$. A simple counting argument will suffice to show that the machine has $d(d-1)$ states. Though this number of states is clearly sufficient, we were unable to show that it is necessary. In fact, when we assembled the example for $d = 4$ (shown in Figure 5.1 above), we realized that some states were redundant in some of the cases. Proving or disproving the minimality of this machine is an important direction for future work.

5.2. **Hamiltonian Path FST for Odd** d**.** For the case where d is odd, it is possible to exploit the GTH rules to build a Hamiltonian path FST with only two states.

Say that we want to build a machine with fewer states. Then we must have fewer "orders" in which the subgraphs are visited. In the even d case, we have a function $INDEX_{a\to b}$ for every $a,b \in \{0,\dots,d-1\}$, and so a different state must be assigned to each (a,b) to account for the different indices of a and b in that sequence.

Given K_d^n with d odd, if we wish to cut down the number of states, we ideally want the order in which we visit the K_d^{n-1} subcopies to be identical to the order in which we visit the K_d^{n-m} subcopies for $2 \le m \le n-1$.

Having only one order is not technically possible for every subcopy; suppose we are moving from the subcopy with leading a to the subcopy with leading b. Then say we exit the subcopy with leading a through its corner with label $acc\dots c$; if we want the order on subcopy b to be $a,\dots,c$, then we must enter via vertex $baa\dots a$. However, this is clearly impossible, since adjacent vertices differ in only one digit. Then we must enter copy b through the edge $bcc\dots c$. However, maybe we can exit copy b through vertex $baa\dots a$, and so only have the original order and that same order reversed.

Now, we must choose the order in which to visit the subcopies. If we want to move from vertex $00\dots0$ to vertex $44\dots4$ in K_5^n, the K_5^{n-1} subcopies with a leading $1,2$, and 3 must be traversed as well.

But we can't just choose any order; if we traverse the K_5^{n-1} subcopies in the order $0,1,2,3,4$, we must traverse the first subcopy in the order $00,01,02,03,04$, and the second copy in the order $14,13,12,11,10$. However, it is not possible to preserve this particular order for the subcopies of K_5^n. As the figure below demonstrates, the order in which the K_5^{n-1} subcopies are traversed is inconsistent with the order in which the K_5^{n-2} subcopies are traversed.

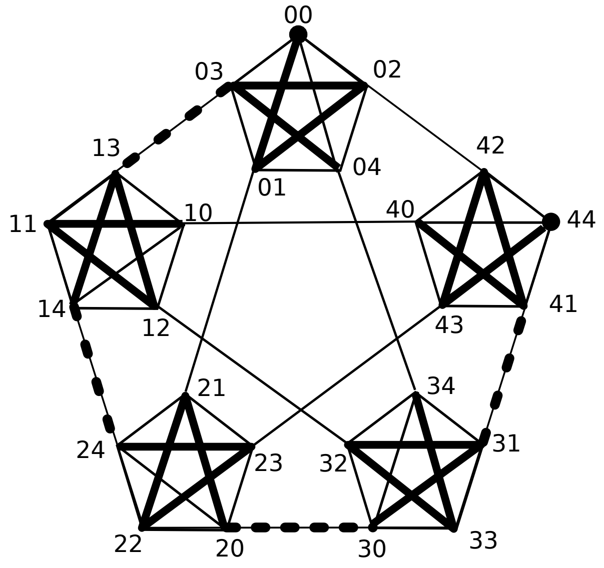


FIGURE 5.2. The dotted path, which represents the order in which the K_5^1 subcopies are traversed, is clearly incompatible with the solid paths, which are traversed in the same implied order.

Here, we define an order that will allow us to recursively traverse the graph in a Hamiltonian path.

Definition 5.4. Let a,b be two numbers such that $(a-b)$ is relatively prime to d. Define the sequence S, where

$$\begin{aligned} s_0 &= a, \\ s_{i+1} &= \begin{cases} 2b - s_i & : i \equiv 0 \bmod 2 \\ 2a - s_i & : \text{otherwise.} \end{cases} \end{aligned}$$

We also observe that the closed form for s_i is

$$s_i = \begin{cases} 2b-(2a-\ldots-(2b-a)\ldots) = (i+1)b - ia & : i \equiv 1 \bmod 2 \\ 2a-(2b-\ldots-(2b-a)\ldots) = (i+1)a - ib & : \text{otherwise.} \end{cases}$$

The first d terms of this sequence will give us an order in which we can traverse the K_d^{n-1} subcopies so that the order will be preserved in the traversal of the smaller subcopies, up to reflection. The definition of this sequence was motivated by the observation that if we expect the path on the first of the subcopies K_d^{n-1} to start at the vertex $aa\ldots a$ and end at the vertex $ab\ldots b$, the rules of GTH restrict the movement of the highest-order disk to $2b-a$. Hence, this is the only edge available in K_d^n, and so the next subcopy visited will be the subcopy with $(2b-a) \bmod d$ as the highest-order digit. We will use this intuition in Proposition 5.3.

We can observe in figure 5.3 that this sequence does work recursively for K_5^2; we have $S = 0,3,2,1,4,\ldots$, which allows the K_5^1 subcopies to be traversed in the same order as K_5^2.

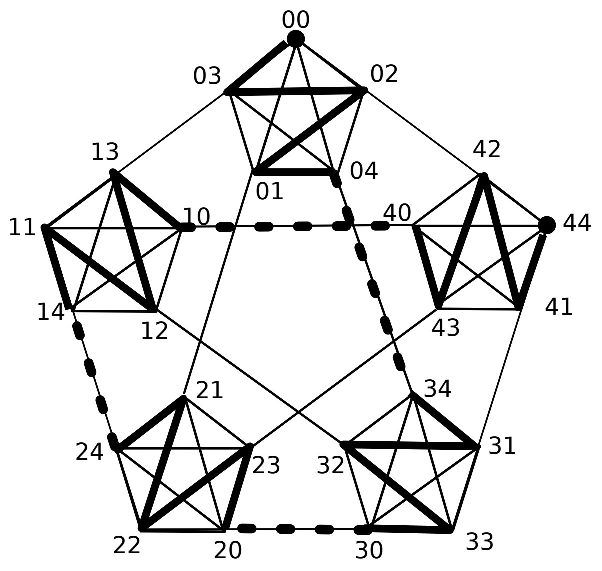


FIGURE 5.3. Here, we see that when we traverse the graph in the order implied by S, the dotted path that traverses the K_5^2 graph and the solid paths that traverse the K_5^1 subcopies are compatible and form a Hamiltonian path on K_5^2.

We first verify that the $(d-1)^{\text{th}}$ term in the sequence is indeed b, so that the traversal of the graph does indeed end with the copy with leading digit b.

Lemma 5.5. *In the sequence S defined in Definition 5.4, the $(d-1)^{th}$ term $s_{d-1} = b$.*

Proof. We note that the closed form for $s_{d-1} = ((d-1)+1)a - (d-1)b = da + (-d+1)b = b$ mod d, as desired. □

Now, we prove that no repetitions occur in the first d digits in the sequence, which is crucial because we must visit each vertex exactly once.

Lemma 5.6. *If $(a-b)$ is relatively prime to d, then the sequence $S = s_0, \ldots, s_i, \ldots$ defined above does not have any repetitions for $0 \le i \le d-1$.*

Proof. Let $(a-b)$ for $a, b \in \{0, \ldots, d-1\}$ be relatively prime to d. Suppose by way of contradiction that there exist s_i, s_j where $s_i \equiv s_j$ mod d and $0 \le i < j \le d-1$. Then by definition of S, we have

$$s_i = \begin{cases} (i+1)b - ia & : i \equiv 1 \bmod 2 \\ (i+1)a - ib & : \text{otherwise,} \end{cases}$$

and

$$s_j = \begin{cases} (j-i)a - (j-i)b - s_i & : j-i \equiv 1 \bmod 2 \\ (j-i+1)a - (j-i-1)b - s_i & : \text{otherwise,} \end{cases}$$

where a and b may be exchanged depending on the parity of i.

First, we note that if $j-i$ is even, then since $s_i \equiv s_j$ mod d

$$0 \equiv s_j - s_i \equiv (j-i)a - (j-i)b \equiv (j-i)(a-b).$$

Now, since $(a-b)$ is not a zero divisor, we see that $(j-i)$ must be equivalent to 0. However, $j \leq d-1$ and $0 \leq i < j$, clearly $j-i < d$. Thus, $j-i$ cannot be a multiple of d. Therefore, we have a contradiction, and $j-i$ cannot be even.

Then let $j-i$ be odd. Now, assume i is odd. Since $s_i \equiv s_j$,

$$\begin{aligned} 2s_i &\equiv s_j + s_i, \\ 2\big((i+1)b - ia\big) &\equiv (j-i+1)a - (j-i-1)b \\ 0 &\equiv (j+i+1)(a-b). \end{aligned}$$

Because $(a-b)$ is not a zero divisor, we realize that $(j+i+1)$ must be equivalent to d. Furthermore, since $j-i$ is odd, $j-i+2i = j+i$ must also be odd, so $(j+i+1)$ is even. However, $0 < i < j < d$. Therefore $j+i < 2d-1$, so we have reached a contradiction and i must be even.

If i is even, then since $s_j \equiv s_i$,

$$0 \equiv s_j - s_i \equiv \big((j+1)b - (j)a\big) - \big((i+1)a - (i)b\big) \equiv (j+i+1)(b-a).$$

Now, $(b-a)$ is relatively prime to d, hence $j+i+1$ must be equivalent to 0 mod d. Again, since $j-i$ is odd $j+i$ must be even, but since $0 < j+i < 2d-1$, we reach a contradiction.

We have exhausted all possible cases, and we realize that the sequence S cannot repeat in the first d terms, as desired. $\square$

Construction 5.7. We construct a Finite State Transducer $\mathcal{H}^{\text{odd}}_{a\to b}$ that takes as input a vertex's GTH-label in K^n_d, d odd, and returns its position along a Hamiltonian path between corner $aa\dots a$ and corner $bb\dots b$, with $a,b \in \{0,\dots,d-1\}$, $a \neq b$ and $(a-b)$ relatively prime to d.

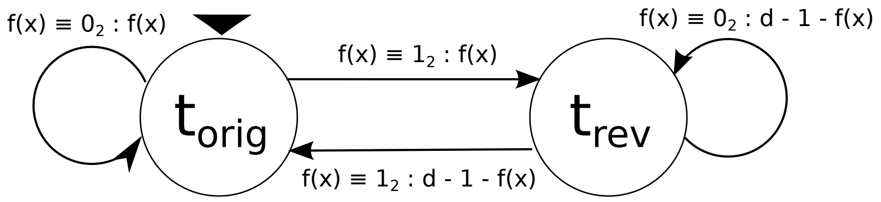


FIGURE 5.4. The FST $\mathcal{H}^{\text{odd}}_{a\to b}$ that takes a vertex label and returns the vertex's index along a Hamiltonian path from $aa\dots a$ to $bb\dots b$ in K^n_d.

The construction is as follows:

(1) First, we construct the "order" in which the vertices are visited by computing the first d terms of the sequence S described above, with $s_0 = a$ and $s_{d-1} = b$. Since Lemma 5.6 guarantees that the first d terms of S will include every number in the range $\{0,\dots,d-1\}$, we create a function f that maps a number to its first index in S.
(2) Let the input string be $x_{n-1}x_{n-2}\dots x_0$, with each $x_i \in \{0,\dots,d-1\}$.
(3) The machine will have two states,

$$t_{\text{orig}} \text{ and } t_{\text{rev}}.$$

These states denote the order in which the vertices are traveled in the particular subcopy of the graph, i.e. whether the order is the reverse of the original order. The start state will be t_{orig}.

(4) Let $\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-1}x_{n-2}\dots x_0;t_{\text{orig}})$ be the index of the vertex labeled $x_{n-1}\dots x_0$ in the Hamiltonian path if $\mathcal{H}^{\text{odd}}$ is started in state t_{orig}. Let $\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-1}x_{n-2}\dots x_0;t_{\text{rev}})$ be the index of the vertex labeled $x_{n-1}\dots x_0$ in the Hamiltonian path if $\mathcal{H}^{\text{odd}}$ is started in state t_{rev}.

(5) We define $\mathcal{H}^{\text{odd}}_{a\to b}$'s transitions as follows:

$$\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-1}x_{n-2}\dots x_0;\ t_{\text{orig}}) = \begin{cases} f(x_{n-1})\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-2}\dots x_0;\ t_{\text{orig}}) & : f(x_{n-1}) \equiv 0 \bmod 2 \\ f(x_{n-1})\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-2}\dots x_0;\ t_{\text{rev}}) & : \text{otherwise.} \end{cases}$$

$$\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-1}x_{n-2}\dots x_0;\ t_{\text{rev}}) = \begin{cases} (d-1-f(x_{n-1}))\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-2}\dots x_0;\ t_{\text{rev}}) & : f(x_{n-1}) \equiv 0 \bmod 2 \\ (d-1-f(x_{n-1}))\mathcal{H}^{\text{odd}}_{a\to b}(x_{n-2}\dots x_0;\ t_{\text{orig}}) & : \text{otherwise.} \end{cases}$$

Proposition 5.8. *The FST $\mathcal{H}^{odd}_{a\to b}$ described in Construction 5.4 maps vertex labels to positions along a Hamiltonian path from vertex $aa\dots a$ to vertex $bb\dots b$ in K^n_d for odd d.*

Proof. We prove that the order of vertices implied by the FST $\mathcal{H}^{\text{odd}}_{a\to b}$ gives a Hamiltonian path beginning at the vertex $aa\dots a$ and ending at the vertex $bb\dots b$. We proceed by induction on m, the iteration of the iterated complete graph.

Base Case. Consider K^1_d. The order given by $\mathcal{H}^{\text{odd}}_{a\to b}$ will include the labels of all vertices, since the first d terms of the sequence S are non-repeating and there are d of them. Clearly, since K^1_d is fully connected this path is valid and all vertices are visited, so the order implied by $\mathcal{H}^{\text{odd}}_{a\to b}$ gives a Hamiltonian path. Furthermore, since we set $s_0 = a$ and $s_{d-1} = b$, the start and end vertices are as desired.

Inductive Step. Suppose that the statement holds for K^m_d. Consider K^{m+1}_d. The path starts with vertex $aa\dots a = s_0s_0\dots s_0$. Call the subcopy K^m_d that contains the start vertex C_0, call the copy that contains the end vertex C_{d-1}, and the intermediate subcopies will be indexed by the order in which they are visited. By our Inductive Hypothesis, the FST will imply a path on K^m_d that follows the sequence S defined in Definition 5.4 and ends at $s_0s_{d-1}s_{d-1}\dots s_{d-1}$.

Now, by the rules of the GTH, the larger disk on tower t is only able to move if all of the smaller disks are on a single tower t^* for $t,t^* \in \{0,1\cdots,d-1\}$. Furthermore, the larger disk must move to tower $(2t^*-t) \bmod d$. Thus, because the edges in K^{m+1}_d are based on allowed GTH game moves, and because all of the vertices in the subcopy C_0 have been visited already, the only edge available is the edge from $s_0s_{d-1}s_{d-1}\dots s_{d-1}$ to $s_1s_{d-1}s_{d-1}\dots s_{d-1}$, where $s_1 = (2s_{d-1}-s_0) \bmod d$. Note that this is the edge that is used by the finite state machine as well.

Now, we must continue the path from vertex $s_1s_{d-1}\dots s_{d-1} \in C_1$. By our inductive hypothesis, the FST will imply a path on C_1 from $s_1s_{d-1}s_{d-1}\dots s_{d-1}$ to $s_1s_0s_0\dots s_0$ that traverses C_1 in the opposite order of the path in C_0. Now, again by the rules of GTH, the only available edge is the edge to vertex $s_2s_0s_0\dots s_0$, where $s_2 \equiv 2s_0 - s_1 \bmod d$; note that this is the next vertex specified by the FST.

By Lemma 5.6, this inductive argument repeats for each C_i for even and odd i. Clearly, this process will create a Hamiltonian path, since each C_i is visited entirely by the inductive hypothesis. Furthermore, since we set $s_0 = a$ the path will start at $aa\dots a$, and by Lemma 5.5 the parity of d the path will end at $bb\dots b = s_{d-1}s_{d-1}\dots s_{d-1}$. The copies $C_0,\dots,C_{d-1}$ are visited in an order such that C_i contains leading bit s_i, and so the FST's assigned order defines a Hamiltonian path with the desired start and end vertices for K^{m+1}_d. □

A complete worked example of $\mathcal{H}^{\text{odd}}_{0\to 2}$ with $d=3$ is shown in Appendix A.

Remark 5.9. We note that $\mathcal{H}^{\text{odd}}$ is clearly minimal; there is no way a 1-state Hamiltonian path FST can be defined, because the transition from one subcopy to another requires the order to switch, so that you cant always assign the start corner's character the output 0.

6. Hamiltonian Circuit

We now use the FSTs $\mathcal{H}$ and $\mathcal{H}^{\text{odd}}$ from Constructions 5.1 and 5.7 to build an FST that takes a vertex's label as input and returns its index along a specific Hamiltonian circuit on K_d^n. We define the machines separately for even and odd d.

Construction 6.1. Define the FST $\mathcal{HC}^{\text{odd}}$ as follows:

(1) Let the inputs be vertex labels of the form $x_{n-1}x_{n-2}\dots x_0$, where $x_i \in \{0,\dots,d-1\}$.
(2) Let the outputs be d-ary numbers representing the index of the input vertex along the Hamiltonian circuit.
(3) K_d^n is composed of d copies of K_d^{n-1}; assign each subcopy the name C_i, where i is the leading digit that all vertices in the subcopy have in common.
(4) We will visit the subcopies in the order $C_0, C_1, \dots, C_{d-1}$. The order in which C_i is traversed will be determined by the starting corner $ia_ia_i\dots a_i$ through which C_i is entered and the end corner $(i+1)b_ib_i\dots b_i$ through which C_i is exited. The GTH rules force our choices of a_i and b_i; to move from copy C_i to copy C_{i+1}, we must move from the configuration $ixx\dots x$ to the configuration $(i+1)xx\dots x$, and so $x = \frac{2i+1}{2} \bmod d$. Hence, for a given C_i, $a_i = \frac{2i-1}{2} \bmod d$ and $b_i = \frac{2i+1}{2} \bmod d$. Note also that $(b_i - a_i) = 1$.
(5) Construct a start state, t_{start}. Then, for each C_i, construct $\mathcal{H}_{a_i \to b_i}$ as described in Construction 5.1.
(6) The transitions of $\mathcal{HC}^{\text{odd}}$, as described in the Figure 5.4, are as follows:

$$\mathcal{HC}^{\text{odd}}(x_{n-1}x_{n-2}\dots x_0;\ t_{\text{start}}) = x_{n-1}\mathcal{H}^{\text{odd}}_{a_{x_{n-1}} \to b_{x_{n-1}}}(x_{n-2}\dots x_0;\ t_{\text{orig}}).$$

Figure 6.1a shows an example of $\mathcal{HC}^{\text{odd}}$ for $d = 5$, along with the Hamiltonian circuit that it implies on K_5^2.

Proposition 6.2. *The FST $\mathcal{HC}^{odd}$ maps a vertex label to its index along a Hamiltonian circuit.*

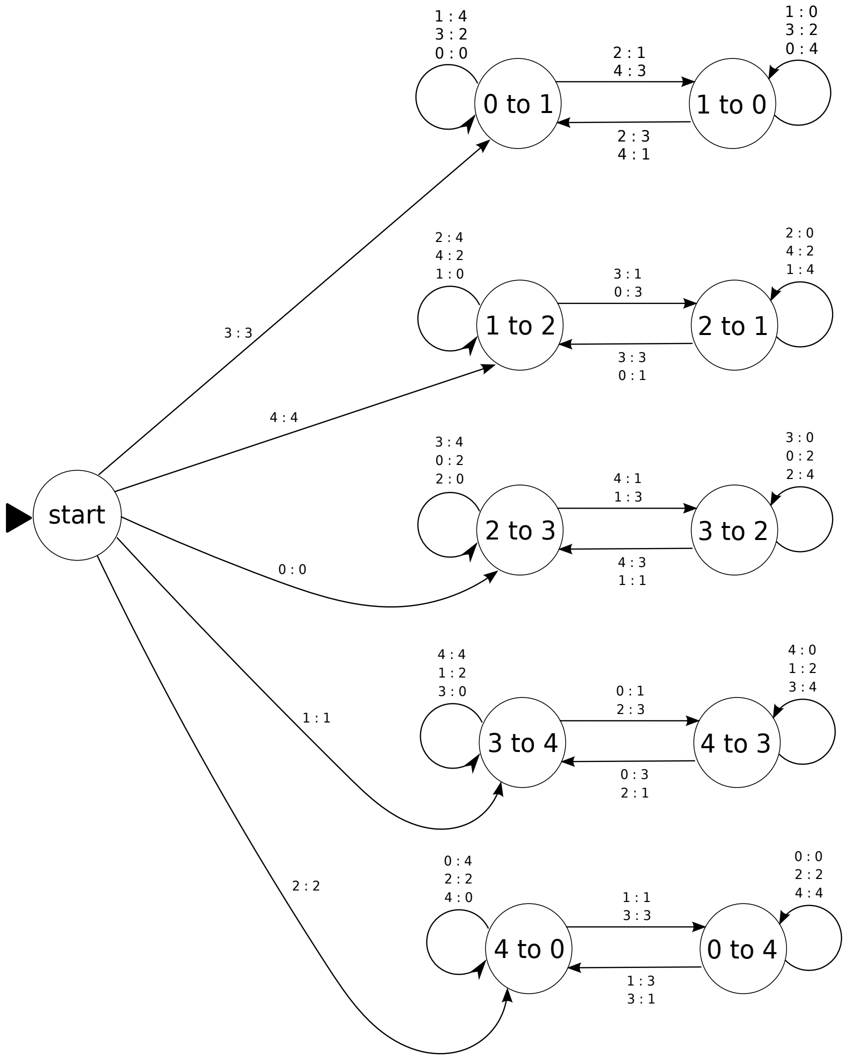
Proof. The first digit assigned by $\mathcal{HC}^{\text{odd}}$ for input $x_{n-1}x_{n-2}\dots x_0$ is x_{n-1}; this is clearly the correct starting digit because $x_{n-1}x_{n-2}\dots x_0$ is a vertex in $C_{x_{n-1}}$ and the vertices in K_d^n are traversed beginning with C_0 and continuing in that order.

Then, $\mathcal{HC}^{\text{odd}}$ acts on $x_{n-2}\dots x_0$ just as $\mathcal{H}^{\text{odd}}_{a_{x_{n-1}} \to b_{x_{n-1}}}$ would. By Proposition 5.8, this substring is assigned its appropriate order on the Hamiltonian path between $a_{x_{n-1}}$ and $b_{x_{n-1}}$, since $(b_{x_{n-1}} - a_{x_{n-1}}) = 1$. It remains to show that this path on $C_{x_{n-1}}$ is appropriately linked to its neighboring copies. Note that

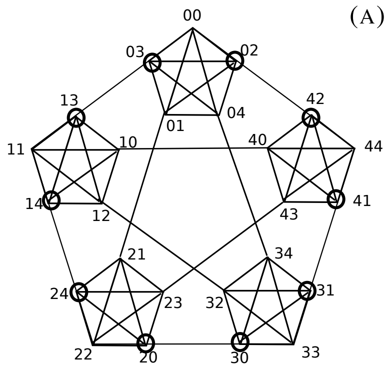
$$a_{x_{n-1}} = \frac{2(x_{n-1})-1}{2} = \frac{2(x_{n-1}-1)+1}{2} = b_{x_{n-1}-1},$$

and similarly

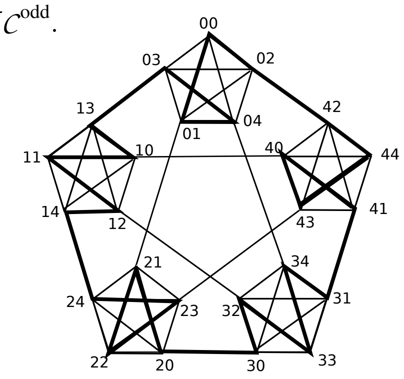
$$b_{x_{n-1}} = \frac{2(x_{n-1})+1}{2} = \frac{2(x_{n-1}+1)-1}{2} = a_{x_{n-1}+1}.$$



(A) The FST $\mathcal{HC}^{\text{odd}}$.



(B) The entrance and exit vertices of each subcopy.



(C) The Hamiltonian circuit given by $\mathcal{HC}^{\text{odd}}$.

FIGURE 6.1. Here we see our FST $\mathcal{HC}^{\text{odd}}$ and the circuit it generates. Clearly, it is composed of 5 distinct copies of $\mathcal{H}^{\text{odd}}$ that are assigned an order by the arrows emanating from the start state.

Thus, we see that the endpoints of these paths are the endpoints of the adjacent paths, and so the FST defines a circuit. □

Remark 6.3. It would have been possible to build $\mathcal{HC}^{\text{odd}}$ with fewer states, by trying to choose some $a, b \in \{0, \dots, d-1\}$ such that as many subcopies as possible are traversed from corner a to corner b or vice-versa. However, the machine we have set forth is particularly nice to describe.

For the case in which d is even, we can use a single copy of the FST $\mathcal{H}$ from Construction 5.1, since the machine already has state $(i \to j)$ for each $i, j \in \{0, \dots, d\}$. We set forth the construction below.

Construction 6.4. Define the FST $\mathcal{HC}$ for the graph K_d^n with $d = 2^m \cdot q$ as follows:

(1) Let the inputs be vertex labels of the form $x_{n-1}x_{n-2}\dots x_0$, where $x_i \in \{0, \dots, d-1\}$.
(2) Let the outputs be d-ary numbers representing the index of the input vertex along the Hamiltonian circuit.
(3) K_d^n is composed of d copies of K_d^{n-1}; assign each subcopy the name C_i, where i is the leading digit that all vertices in the subcopy have in common.
(4) We will visit the subcopies in the order $C_0, C_1, \dots, C_{d-1}$. The order in which C_i is traversed will be determined by the starting corner $ia_ia_i\dots a_i$ through which C_i is entered and the end corner $(i+1)b_ib_i\dots b_i$ through which C_i is exited. The Combination Puzzle rules force our choices of a_i and b_i.

   To move from copy C_i to copy C_{i+1}, we must move from the configuration $ixx\dots x$ to the configuration $(i+1)xx\dots x$. Hence, let $x = 2^m \cdot t_x + r_x$, let $i = 2^m \cdot t + r$, and let $i+1 = 2^m \cdot t' + r$. Then by the rules of the Combination Puzzle, we see that $t_x = \frac{t+t'}{2} \bmod q$ and $r_x = r \oplus r'$. Hence, for a given C_i with $i-1 = 2^m \cdot t^* + r^*$, $i = 2^m \cdot t + r$, and $i+1 = 2^m \cdot t' + r'$, then

$$\begin{aligned} a_i &= 2^{m-1}((t+t^* \bmod q) + (r \oplus r^*), \\ b_i &= 2^{m-1}((t+t') \bmod q) + (r \oplus r'). \end{aligned}$$

(5) Construct a start state, t_{start}.
(6) The transitions of $\mathcal{HC}^{\text{odd}}$, as described in the figure 5.4, are as follows:

$$\mathcal{HC}^{\text{odd}}(x_{n-1}x_{n-2}\dots x_0;\ t_{\text{start}}) = x_{n-1}\mathcal{H}(x_{n-2}\dots x_0;\ (a_{x_{n-1}}, b_{x_{n-1}}, orig).$$

   The transitions of $\mathcal{H}$ are as defined in Construction 5.1.

Proposition 6.5. *The machine $\mathcal{HC}$ described in Construction 6.4 correctly creates a Hamiltonian circuit on K_d^n for any d.*

Proof. The proof is almost identical to that of Proposition 6.2; one need only verify that the transitions between the subcopies are correct, and then invoke the correctness of $\mathcal{H}$. □

7. Codevertex Recognition

Definition 7.1. A *perfect one error correcting code*, often abbreviated as *p1ecc*, on a graph is a subset C called the codevertices of the vertex set which satisfies the following properties:

(1) No two codevertices are adjacent.
(2) Every noncodevertex is adjacent to exactly one codevertex. [1]

Along the Hamiltonian path of K_2^n, the order of the vertices is described by a binary reflected Gray code. It is fairly obvious that by the construction of the graph K_2^n that every 3^{rd} vertex along the graph will be a codevertex. Figure 7.1 gives a pictorial proof for this.

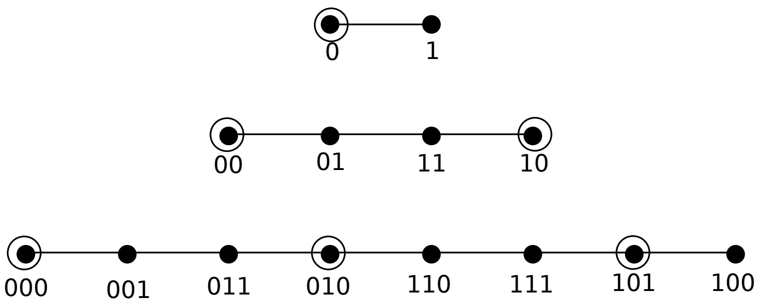


FIGURE 7.1. The graphs of K_2^1, K_2^2, and K_2^3 labeled in binary reflected Gray code.

Our conjecture is that if the index of a vertex along the Hamiltonian path on K_3^n is divisible by 4, then that vertex is a codevertex in the p1ecc. Note that this cannot hold for a circuit on K_3^n. The order of the vertices of K_3^n along the Hamiltonian path is described by a sequence of n bit strings in ternary reflected Gray code. If we travel from corner 00...0 to corner 22...2 we can construct the FST in Figure 7.2 by Construction 5.7.

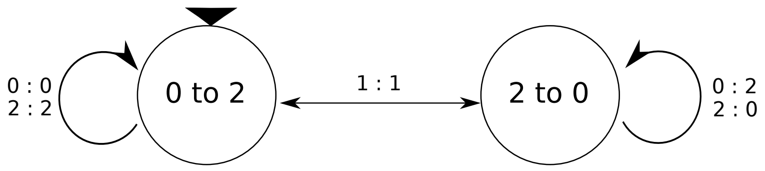


FIGURE 7.2. The FST takes a vertex label in ternary reflected Gray code and returns the vertex's index along the Hamiltonian path in ternary.

We can use the FST created by Baun and Chauhan shown in Figure 7.3, which takes as inputs ternary numbers and outputs a string that represents the quotient when the number is divided by 4. [1] The states represent the remainder and therefore the number is divisible by four if the final state is 0, the accepting state.

We compose 7.2 and 7.3 to get the FST whose input is ternary reflected Gray code and whose output is the quotient. The notation of the states is that the top number represents the summation mod 2 and the bottom number represents the remainder when divided by four. The two accepting states are $\substack{1\\0}$ and $\substack{0\\0}$ because the bottom digit is 0 and therefore the index is divisible by four. However, there are two disjoint FSTs in this figure so we can separate it into two FSTs in Figure 7.4. For the Hamiltonian path from configuration 00...0 to 22...2, the machine on the left of Figure 7.4 with starting state $\substack{0\\0}$ will accept exactly those strings whose indices along this path are divisible by 4. This is shown in Table A.1.

In order to prove that only the codevertices of K_3^n are divisible by four, we use a theorem by Ingrid Nelson in 1999.

Theorem 7.2. *In p1ecc, the codewords are those which have an even number of 1's and 2's.* [3]

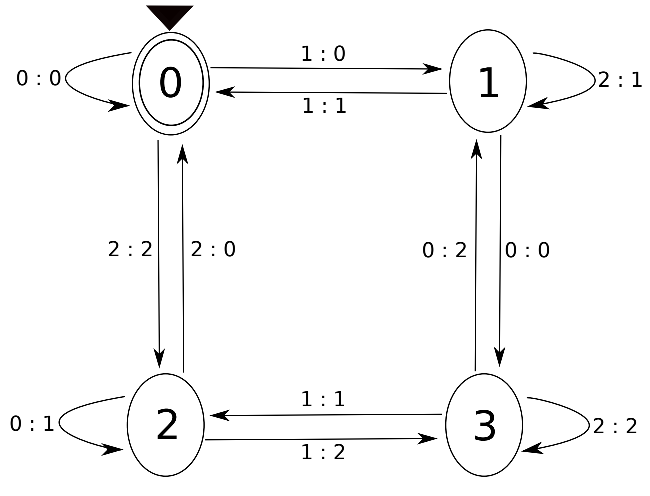


FIGURE 7.3. FST which divides ternary numbers by 4.

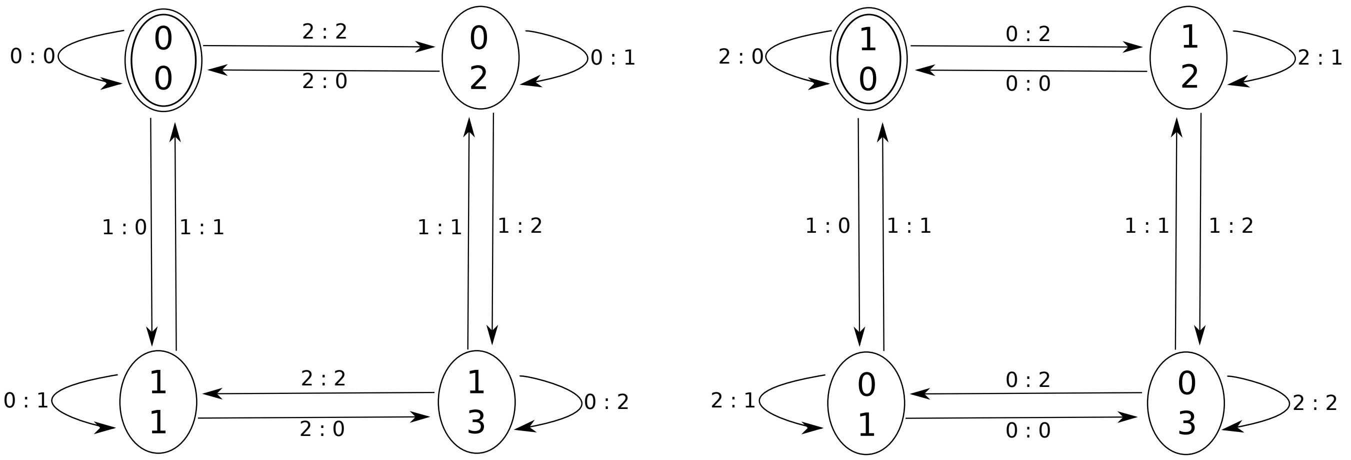


FIGURE 7.4. FSTs which deduce whether or not a vertex along the Hamiltonian path of K_3^n will have an index that is divisible by 4.

Using Theorem 7.2 , we can create Figure 7.5. The FSM in Figure 7.5 obviously is similar to those in Figure 7.4. Therefore, if we compare the two we can see that the vertices whose index along the path is divisible by 4 are codevertices of the p1ecc. However, if we try to generalize this to K_d^n for $d \geq 3$ we cannot find a way to recursively construct a Hamiltonian path in which the index of the codevertices along the Hamiltonian path are divisible by $d+1$. This is possibly because Hamiltonian paths between two corner vertices in K_d^2 are not unique for $d > 3$, as proven in Theorem 4.6.

8. MINIMUM NUMBER OF DISKS REQUIRED SO THAT ALL TOWERS ARE UTILIZED.

Lemma 8.1. *In a game of GTH with n disks, a tower will be used if and only if it is at one point occupied by the smallest disk.*

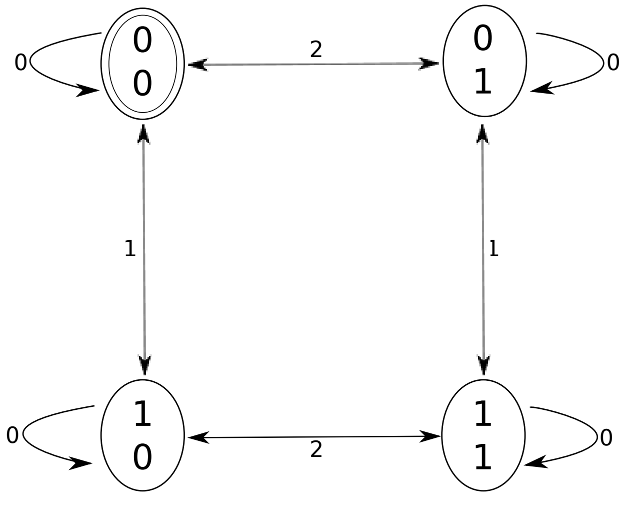


FIGURE 7.5. FSM based on Theorem 7.2.

Proof. Clearly, if a tower is occupied by the smallest disk, then it becomes used. Conversely, if in the course of the game k^{th} smallest disk was moved onto tower t for some $0 < k \leq n$, then by the GTH game rules the $k-1$ smaller disks were stacked on tower t $2^{k-1}-1$ moves later. Since this stack of the k smallest disks includes the smallest disk, the smallest disk occupied the used tower. $\square$

Proposition 8.2. *In a game of GTH with n disks and d towers, when moving the disks from tower* 0 *to tower* $d-1$*, all towers will be used if and only if* $d-1 \leq 2^{n-1}$.

Proof. First, suppose $d-1 \leq 2^{n-1}$. Since d is odd, 2 has a multiplicative inverse mod d. Since $\gcd(d, d-1) = 1 = \gcd(d, \frac{1}{2})$, the product $\frac{d-1}{2^{n-1}}$ is also relatively prime to d. By the equal increment theorem [8], with each move the smallest disk's position is incremented by $\frac{d-1}{2^{n-1}}$, and since $\gcd(\frac{d-1}{2^{n-1}}, d) = 1$ it takes d moves until the smallest disk repeats a tower. Since the smallest disk makes $2^{n-1} \geq d-1$ moves starting on the 0th tower, it will thus occupy each of the d towers.

Contrapositively, let $d-1 > 2^{n-1}$. Because the smallest disk moves 2^{n-1} times, it occupies at most $d-1$ of the d towers. By Lemma 8.1, we conclude that not all of the towers are utilized. $\square$

9. CONCLUSION

For GTH, the Combination Puzzle, and corner-to-corner Generalized Spin-Out we were able to create a FST $\mathcal{C}to\mathcal{I}$ that maps a puzzle configuration to its index in the solution sequence. We also proved the minimality of these Transducers. Finally, we showed that it is impossible to construct a FST for the Generalized Spin-Out sequence starting with $11\ldots1$ and ending with $00\ldots0$.

At the start of our research, we had conjectured that Hamiltonian paths on K_d^n may be unique for all d. However, for all $d > 3$, $n > 1$ we were able to construct two distinct Hamiltonian paths on K_d^n for $d > 3$.

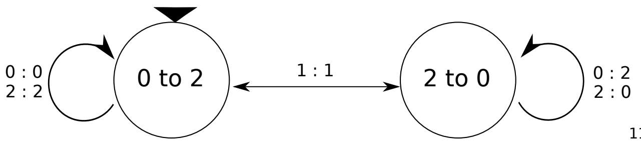
Realizing that Hamiltonian paths were not unique, we defined constrains for our Hamiltonian paths and were thus able to construct a finite-state-computable Hamiltonian path on K_d^n. In the case of even d, we have a $d(d-1)$-state FST, $\mathcal{H}$, that maps vertex label to index along the Hamiltonian path. We are unsure of whether this number of states is minimal; this is an open question that deserves attention in future research. For the case in which d is odd, we were able to find a 2-state FST, $\mathcal{H}^{\text{odd}}$, that describes a Hamiltonian path on K_d^n.

We were then able to use the Hamiltonian path FSTs to build FSTs that describe Hamiltonian circuits on K_d^n for $n > 1$.

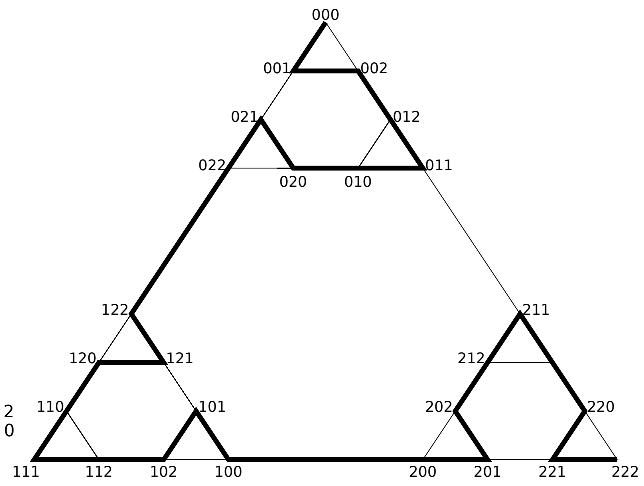
At the outset, we were hoping to describe a finite-state-computable Hamiltonian path on K_d^n on which p1ecc codevertices have indices divisible by $d+1$. We were able to describe such FSTs for $d = 2, 3$. However, we were not able to find such an FST for $d > 3$. This difficulty may arise from the fact that Hamiltonian paths are no longer unique for $d > 3$. We conjecture that the requirement that the K^{n-1}-subcopies are visited one at a time is incompatible with the codevertex requirement, and that such a path is not finite-state computable. Verifying or disproving this conjecture is an open problem for future research.

Appendix A. An Example of the Hamiltonian Path FST and Codevertex Recognition for $d = 3$

In this appendix we show an example of the Hamiltonian path FST for odd d, with the case $d = 3$. We construct the machine $\mathcal{H}^{\text{odd}}_{0\to 2}$ as described in Construction 3.5. This machine is identical to the one from Figure 7.2, but is shown here again in Figure A.1a for convenience.



(A) The FST $\mathcal{H}^{\text{odd}}_{0\to 2}$.



(B) The corresponding Hamiltonian path on K_3^3.

Figure A.1. The FST $\mathcal{H}^{\text{odd}}_{0\to 2}$ and the corresponding Hamiltonian path from corner 000 to corner 222 in K_3^3.

Using the unique Hamiltonian path from Figure A.1b and Table A.1 below, we can see that $\mathcal{H}^{\text{odd}}_{0\to 2}$ does indeed map a vertex's label to its index along the Hamiltonian path. We can also note that if we reverse the inputs and outputs, the resulting FST $\mathcal{G}^{odd}$, takes the index along the Hamiltonian path to the vertex label. Every fourth vertex in Table A.1 is marked and it can be easily verified that these correspond to the codevertices which have an even number of 1's and an even number of 2's in their label.

index along path input to $\mathcal{G}^{odd}$ output of $\mathcal{H}^{odd}$	vertex label from figure A.1b input to $\mathcal{H}^{odd}$ output of $\mathcal{G}^{odd}$
000	000 *
001	001
002	002
010	012
011	011 *
012	010
020	020
021	021
022	022 *
100	122
101	121
102	120
110	110 *
111	111
112	112
120	102
121	101 *
122	100
200	200
201	201
202	202 *
210	212
211	211
212	210
220	220 *
221	221
222	222

TABLE A.1

Since the output of $\mathcal{H}^{\text{odd}}$ is the index along the path and the output of $\mathcal{G}^{odd}$ is the vertex label, our table clearly shows that $\mathcal{H}^{\text{odd}}$ and $\mathcal{G}^{odd}$ work for K_3^3.

REFERENCES

[1] Lindsay Baun and Sonia Chauhan. Puzzles on graphs: The towers of hanoi, the spin-out puzzle, and the combination puzzle. 2009.

[2] P. Cull and E.F. Ecklund Jr. Towers of Hanoi and Analysis of Algorithms. *American Mathematical Monthly*, 92(6):407–420, June-July 1985.

[3] Paul Cull and Ingrid Nelson. Error-correcting codes on the Towers of Hanoi graphs. *Discrete Math.*, 208/209:157–175, 1999.

[4] Kathleen King. A new puzzle based on the SF labelling of iterated complete graphs. 2004.

[5] Sandi Klavžar, Uroš Milutinović, and Ciril Petr. On the Frame-Stewart algorithm for the multi-peg Tower of Hanoi problem. *Discrete Applied Mathematics.*, 120(1-3):141–157, 2002.
[6] Stephanie Kleven. Perfect Codes on Odd Dimension Serpinski Graphs. 2003.
[7] Zvi Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, 1978.
[8] Leanne Merrill and Tony Van. A tale of two puzzles. 2010.
[9] Carla Savage. A survey of combinatorial Gray codes. *SIAM Review*, 39:605–629, 1996.
[10] Nick Stevenson and Beth Skubak. A new puzzle on complete iterated graphs with dimension 2^m. 2008.
[11] Elizabeth Weaver. Gray codes and puzzles on iterated complete graphs. 2005.