Cycles in the 2-ball Juggling Digraph

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Candace Schenk

May 2002

Approved for the Division
(Mathematics)

David Perkinson

Che cosa ho combinato?

I would like to thank my family and friends who have put up with my messes through the years.

I would also like to thank my advisor for being so meticulous throughout my thesis experience.

Contents

Abstract

This thesis studies juggling patterns. Through the use of directed graphs and other formal systems we count the number of 2-ball juggling cycles. A new result appears in 2.2.3, where the number of primitive 2-ball juggling patterns are counted.

Chapter 1

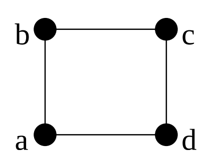
Basic graph theory

Introduction. This chapter will give the basic notation used for the rest of this thesis.

1.1 Notation

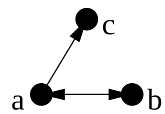
A *graph* is a ordered triple $G = (V, E, \Phi)$ where V and E are sets and Φ is a function from E into the collection of unordered, not necessarily distinct pairs of elements of V. Elements of V are called *vertices* of G and elements of E are called *edges* of G. If $e \in E$ and $\Phi(e) = \{v, v'\}$, we say that e is an edge joining vertices v and v'. If $v = v'$, then e is called a *loop*. We normally identify elements of E with their images under Φ.

Example: Let $G = (V, E, \Phi)$ where $V = \{a, b, c, d\}$, $E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}$:



A *digraph* is an ordered triple $D = (V, E, \Phi)$ where V and E are sets and Φ is a function from E into the collection of *ordered*, not necessarily distinct, pairs of elements of V.

Example: Let $V = \{a, b, c\}$, $E = \{(b, a), (a, b), (a, c)\}$, and $D = (V, E)$, then D looks like:



1.2 Walks Paths and Cycles

A *walk* W in a directed graph is a finite, ordered list of edges $(e_1, e_2, ..., e_k)$ where the endpoint of e_i is the initial point of e_{i+1} for all i, where the *endpoint* of e_i is defined as the second element of the edge e_i and the *initial point* is the first element of the edge e_i.

A *path* is a walk such that each vertex in the walk is only visited once, i.e. each vertex in the walk is distinct. A *cycle* is a walk with the property that the first and the last vertices are the same. A *primitive cycle* is a cycle with all but the first and the last vertices distinct. The *length*, $\ell(W)$, of a walk W on a graph is the number of edges used in that walk.

Definition: If there is a walk connecting vertex u to vertex v, then the *distance* between u and v is $d(u, v) = \min_W \ell(W)$, taking the minimum over all walks W from u to v. We will define $d(u, u) = 0$. Note $d(u, v) \neq d(v, u)$ in general.

Theorem 1.2.1. The Triangle Inequality*: Let u,v,w be vertices of a graph G. Let there exist walks from u to v, u to w, and w to v. Then*

$$d(u, v) \leq d(u, w) + d(w, v).$$

Proof. Let Q, P, R be walks of minimal length from u to v, u to w, and w to v, respectively. We can concatenate the walks P and R, which yields a walk from u to v, by way of w. This walk, call it PR, is not necessarily the shortest walk from u

to v. Therefore,

$$d(u,v) = \ell(Q) \leq \ell(PR) = \ell(P) + \ell(R) = d(u,w) + d(w,v).$$

$\square$

Chapter 2

Juggling Digraphs

Introduction. One application of digraphs is modeling juggling patterns. It is possible to encode some of the important information of a juggling pattern in a digraph. The main question I will answer is: how many 2-ball patterns are there of a given length? Analyzing adjacency matrices of the 2-ball juggling digraph, we recover the known result that there are $3^n - 2^n$ patterns having period dividing n. By analyzing the digraph in a different way, we get a new result: a formula for the number of primitive 2-ball patterns cf. Definition 2.1.2.

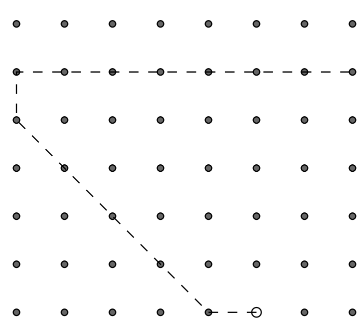
2.1 The juggling digraph of 2-balls

The system to describe 2-ball juggling patterns is relatively simple. Let each *juggling state* be an element of $\mathbf{F}_2^\infty = \{x_1x_2x_3\ldots | x_i \in \{0,1\}\}$ such that the number of 1s in the string is 2. Each 0 in a juggling state signifies the amount of time it will take the 1 to the right of that 0 to land. Out of convenience we will, for example, write the string 00010010000... as 0001001, omitting the trailing 0s. Given this string, we know that one ball will land in 4 seconds, and the other in 7 seconds.

It is possible to create a digraph that uses these juggling states as its vertices. The vertices are in the lattice $\mathbf{N}^2_{\geq 0} = \{(i,j) \in \mathbf{N} \times \mathbf{N} \mid i,j \geq 0\}$. There is a bijection between the juggling states and ordered pairs (i, j) in the lattice where i denotes the number of 0s between the first 1 and the second 1, and j denotes the number of 0s to the left of the first 1. For example the juggling state $b = 00001000001$ corresponds to the vertex $(5, 4)$, and the vertex $(3, 7)$ corresponds to the juggling state 000000010001.

Given this encoding there are two different sorts of juggling states that need to be defined. A juggling state b is said to be in the *ground state* if the leading digit is a 1, and in an *elevated state* if the leading digit is a 0. The edges in the digraph can be described as follows: if the balls are in an elevated state (a, b), i.e. where $b \neq 0$, the only state that can be moved to is $(a, b - 1)$. If the balls are at ground state, i.e. the state $(a, 0)$, the ball can be thrown to any state of the form (x, a) for any x or to any state $(x, a - 1 - x)$ for any $0 \leq x \leq a - 1$. The juggling state 00001000001 can only move to 0001000001, where as the juggling state 1001 can move to 101, 011, 0011, or 001n1 where n is a string of 0s.

An intuitive way to think about the edges is to look at the following diagram. The vertices on the dashed line are all the possible vertices that could be reached from the state $(5, 0)$.



The only edge from an elevated state (a, b) is to the state $(a, b - 1)$, appearing directly below (a, b) in the digraph.

Definition 2.1.1. *A* 2-ball juggling pattern *is a cycle in the* 2*-ball juggling digraph.*

Definition 2.1.2. *A juggling pattern is* primitive *if it contains no sub-cycles.*

2.1.1 Distances

For each state $(x, y) \in \mathbf{N}^2$ in the digraph, the *distance diagram* $B(x, y) = (a_{ij})$ is the infinite array where $a_{ij} = d((x, y), (i, j))$ for all $i, j \in \mathbf{N}$. For example:

$$B(4,3) = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & \cdots \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & \cdots \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & \cdots \\ 4 & 5 & 5 & 5 & 0 & 5 & 5 & \cdots \\ 5 & 4 & 6 & 6 & 1 & 6 & 6 & \cdots \\ 6 & 5 & 4 & 7 & 2 & 7 & 7 & \cdots \\ 7 & 6 & 5 & 4 & 3 & 8 & 8 & \cdots \end{pmatrix} \qquad B(3,0) = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ 5 & 5 & 5 & 5 & 5 & \cdots \\ 5 & 5 & 5 & 5 & 5 & \cdots \\ 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & 2 & 2 & 2 & 2 & \cdots \\ 2 & 1 & 3 & 3 & 3 & \cdots \\ 3 & 2 & 1 & 0 & 4 & \cdots \end{pmatrix}$$

In general the distance diagram $B(k, 0)$ looks like:

$$\begin{pmatrix} \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ k+1 & \cdots & k+1 & k+1 & k+1 & k+1 & k+1 & k+1 & \cdots \\ k+1 & & k+1 & k+1 & k+1 & k+1 & k+1 & k+1 & \cdots \\ 1 & & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & & 2 & 2 & 2 & 2 & 2 & 2 & \cdots \\ 2 & & 3 & 3 & 3 & 3 & 3 & 3 & \cdots \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ k-2 & & 1 & k-1 & k-1 & k-1 & k-1 & k-1 & \cdots \\ k-1 & & 2 & 1 & k & k & k & k & \cdots \\ k & \cdots & 3 & 2 & 1 & 0 & k+1 & k+1 & \cdots \end{pmatrix}$$

From these examples we can see the following:

$$d((i,0),(j,0)) = \begin{cases} i+1 & \text{if } i < j, \\ i-j & \text{if } i \geq j \end{cases}$$

Theorem 2.1.3. *Let v be a cycle, and let (i, j) be a state occurring in v. Then i and j are strictly less then the length of the cycle.*

Proof. Let $v = v_1 v_2 \cdots v_{\ell(v)}$ where each v_i is a juggling state, $d(v_i, v_{i+1}) = 1$, and $\ell(v)$ is the length of v. Let $(i, 0)$ be the left-most (smallest first coordinate) ground state occurring in v, and $(i', 0)$ be the right-most. By the triangle inequality:

$$\ell(v) = \sum_{j=1}^{\ell(v)} d(v_j, v_{j+1}) \geq d((i,0),(i',0)) + d((i',0),(i,0)) = (i+1) + (i'-i) = 1 + i'$$

Therefore $(i', 0)$, the vertex farthest to the right, has first coordinate strictly less then $\ell(v)$. Thus the first coordinate of any state in v is strictly less than $\ell(v)$.

Further, as just stated, if (i, j) is an elevated state appearing in v, then $(i, j) \rightarrow (i, j-1) \rightarrow \cdots \rightarrow (i, 0)$ appears in v. Thus $j + 1 \leq \ell(v)$. The result follows. □

Remark 2.1.4. We now know that when searching for all cycles of length ℓ we need only to look at a $\ell \times \ell$ lattice of points.

2.1.2 Data

To find all the primitive juggling cycles of a given length, we wrote a program which did a standard depth-first search. [1] We found the following data:

cycle length	1	2	3	4	5	6	7	8	9
# of primitive cycles	1	2	5	10	23	48	105	216	467

Another useful way to look at these data is to see the number of cycles that use a given number of columns in the juggling digraph. The following table displays exactly that.

[1] See the first appendix for code

cycle length	Number of cycles using X columns								
	1	2	3	4	5	6	7	8	9 $\leftarrow X$
1	1								
2	1	1							
3	1	3	1						
4	1	4	4	1					
5	1	6	10	5	1				
6	1	7	18	15	6	1			
7	1	9	29	37	21	7	1		
8	1	10	40	70	58	28	8	1	
9	1	12	58	128	136	86	36	9	1

The next set of data shows the number of cycles passing through each point of the digraph. More formally, the (i, j) entry of each $\ell \times \ell$ matrix shows the number of primitive cycles of length ℓ containing the juggling state (i, j).

$$\begin{array}{c} 1 \end{array} \qquad \begin{array}{cc} 0 & 1 \\ 1 & 2 \end{array} \qquad \begin{array}{ccc} 0 & 0 & 1 \\ 2 & 0 & 2 \\ 3 & 3 & 4 \end{array} \qquad \begin{array}{cccc} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 2 \\ 4 & 2 & 0 & 4 \\ 6 & 5 & 6 & 8 \end{array} \qquad \begin{array}{ccccc} 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 2 \\ 4 & 4 & 0 & 0 & 4 \\ 9 & 7 & 6 & 0 & 8 \\ 14 & 13 & 13 & 12 & 16 \end{array}$$

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 2 \\ 4 & 4 & 0 & 0 & 0 & 4 \\ 10 & 10 & 4 & 0 & 0 & 8 \\ 20 & 15 & 11 & 12 & 0 & 16 \\ 30 & 28 & 25 & 26 & 24 & 32 \end{array} \qquad \begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 4 & 4 & 0 & 0 & 0 & 0 & 4 \\ 10 & 10 & 8 & 0 & 0 & 0 & 8 \\ 22 & 20 & 14 & 12 & 0 & 0 & 16 \\ 44 & 33 & 29 & 28 & 24 & 0 & 32 \\ 66 & 62 & 57 & 59 & 52 & 48 & 64 \end{array}$$

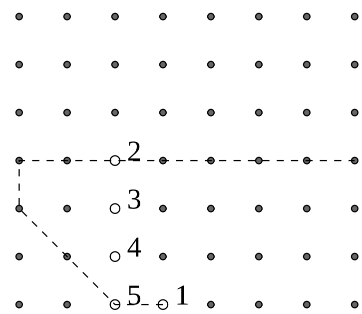
$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 4 & 4 & 0 & 0 & 0 & 0 & 0 & 4 \\ 10 & 10 & 8 & 0 & 0 & 0 & 0 & 8 \\ 24 & 20 & 20 & 8 & 0 & 0 & 0 & 16 \\ 48 & 43 & 33 & 22 & 24 & 0 & 0 & 32 \\ 94 & 68 & 63 & 49 & 56 & 48 & 0 & 64 \\ 138 & 128 & 120 & 111 & 118 & 104 & 96 & 128 \end{array}$$

0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	2
4	4	0	0	0	0	0	0	4
10	10	8	0	0	0	0	0	8
24	20	20	16	0	0	0	0	16
52	46	46	28	24	0	0	0	32
106	100	71	61	56	48	0	0	64
204	154	141	129	116	112	96	0	128
300	282	266	251	249	236	208	192	256

2.2 The dot game

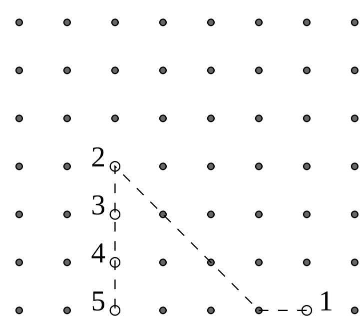
There is yet another way to think about 2-ball juggling cycles. Given any juggling cycle, that cycle can be written as a string of integers and dots, where the numbers correspond to columns of our digraph, and the dots tell how the balls move. More formally: any juggling cycle can be written as $a_1 b_1 a_2 b_2 \ldots a_n b_n$ where $a_i \in \mathbf{N}$ and $b_i \in \{^{\bullet}, _{\bullet}\}$. We will call $^{\bullet}$ an up-dot, and $_{\bullet}$ a down-dot. The number a_i in this new notation stands for the ground state $(a_i, 0)$ in the digraph. The next vertex in the cycle will occur in the column containing $(a_{i+1}, 0)$. Note that there are two ways to walk— we'll often say *throw*— from $(a_i, 0)$ directly to a vertex of the form (a_{i+1}, x); namely there is a throw to $(a_{i+1}, a_i - a_{i+1} - 1)$ and a throw to (a_{i+1}, a_i). The former we denote with a down-dot, $a_{i\bullet} a_{i+1}$, and the latter with an up-dot, $a_i^{\bullet} a_{i+1}$. Note that when b_i is a down-dot, then $a_i > a_{i+1}$. This also means that if $a_i = 0$ then b_i is an up-dot. The following pictures give a better description of this system:

$3^\bullet 2$ looks like:

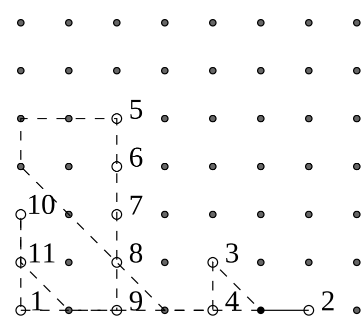


Where the dots with the white centers show the walk through the digraph and the numbers are the order of the walk.

$6_\bullet 2$ looks like:



So the *cycle* $0^\bullet 6_\bullet 4^\bullet 2^\bullet$ would look like:



Note. The cycle represented by the string $a_1 b_1 \ldots a_n b_n$ is primitive if and only if each a_i is distinct.

2.2.1 Length

Given a cycle in the dot notation, there is a simple algorithm to find its length. Namely, if a_i is followed by an up-dot, then that column adds $a_i + 1$ to the total cycle length. If a_i is followed by a down-dot, then it adds $a_i - a_{i+1}$ to the total cycle length. So $\ell(0^\bullet 6_\bullet 4^\bullet 2^\bullet) = (0+1) + (6-4) + (4+1) + (2+1) = 11$.

2.2.2 Substitutions

Having represented a juggling cycle as a finite string, we can create more juggling cycles via the following three string substitutions:

This rule adds nothing to the cycle's length:

(1.0) $a^\bullet \leftrightarrow a_\bullet b^\bullet$ for any $b < a$

These rules add one to the cycle's length:

(1.1) ${}^\bullet a^\bullet \rightarrow {}^\bullet(a+1)^\bullet$

(1.2) $a_\bullet 0^\bullet \rightarrow a^\bullet 0^\bullet$

Example: Starting from $0^\bullet 6^\bullet 2^\bullet$, rule 1.0 produces the cycles $0^\bullet 6_\bullet 4^\bullet 2^\bullet$ and $0^\bullet 6_\bullet 2^\bullet 2^\bullet$ without changing length; the latter cycle is not primitive.

Theorem 2.2.1. *Starting with the cycle* $0^\bullet$*, these three rules produce all juggling cycles. If we only substitute distinct columns then these three rules produce all primitive juggling cycles.*

Proof. We will prove this statement by induction over the cycle length ℓ.

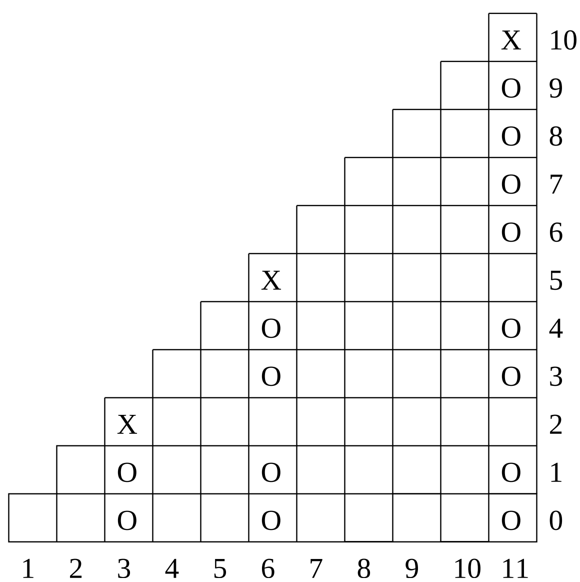
Base case: $\ell = 1$. There is one cycle with cycle length one, that is $0^\bullet$.

Induction: Let $c = a_1 b_1 \ldots a_n b_n$ be a cycle of length $\ell > 1$. Since c is a finite cycle, there will be a smallest number among the a_i, i.e. a left-most column. This column must be followed by an up-dot. Therefore there is at least one up-dot in c. Note that if the smallest column is not distinct this still holds.

By repetitively applying rule 1.0, we may assume that all dots are up. We may then apply rule 1.1 in reverse, yielding a new cycle c' of length $\ell - 1$. By induction, c' is derived from $0^\bullet$ from the three rules, and c follows from c' by rule 1.1. If c is primitive, c' can be chosen to be primitive too except, possibly, in the case where $0^\bullet$ is in c. A typical problem case would be $c = 0^\bullet 1^\bullet 2^\bullet$. However, in that case, we may apply rule 1.2 in reverse to yield a new primitive cycle c' of length $\ell - 1$, and the result follows similarly by induction. $\square$

2.2.3 Counting

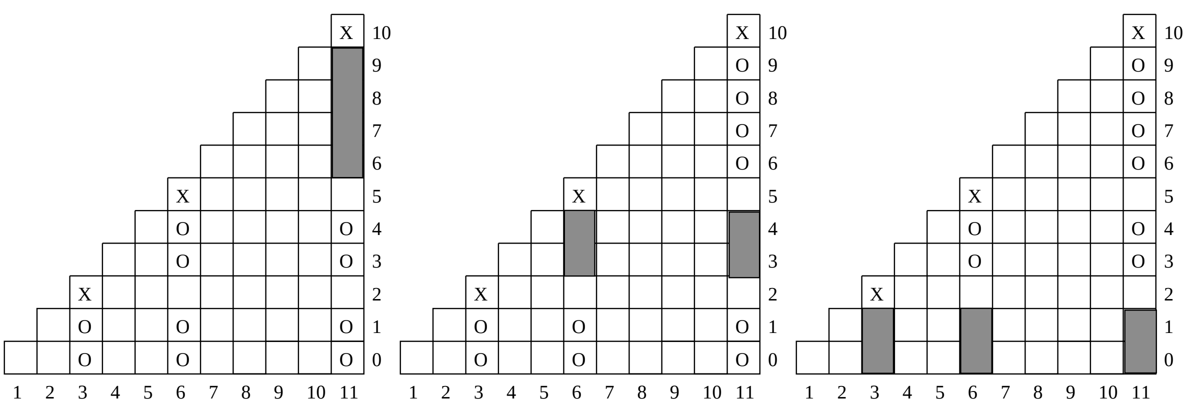
It is possible to find all cycles of a given length ℓ that have only up-dots when expressed in dot notation, and then by repetitively applying rule 1.0 we produce all cycles of length ℓ. Consider the following picture representation of $10^\bullet 5^\bullet 2^\bullet$. The squares containing an X denotes columns that are followed by an up-dot. The squares containing O denote the possible columns that can be added to the cycle by rule 1.0.



Hence, adding 8 and 4 in column numbered 11 in the diagram and 3 in column 6 represents the pattern $10_\bullet 8_\bullet 4^\bullet 5_\bullet 3^\bullet 2^\bullet$. So adding *rows* numbered 8 and 4 in column 11 adds *columns* 8 and 4 to our juggling pattern, written in dot-notation. Note that the addition of new columns by rule 1.0 does not change the cycle length. By numbering the columns 1–11 instead of 0–10, we can find the length by just adding the column numbers:

$$\ell(10^\bullet 5^\bullet 2^\bullet) = 11 + 6 + 3 = 20 = \ell(10_\bullet 8_\bullet 4^\bullet 5_\bullet 3^\bullet 2^\bullet).$$

We now want to count the number of primitive patterns that can be formed from $10^\bullet 5^\bullet 2^\bullet$ by applying rule 1.0. First consider the possible added columns after $10^\bullet$ that are numbered greater than 5: there are $2^{(10-5-1)}$ possible combinations. Now consider the possible columns added numbered below 5 and above 2: there are $3^{(5-2-1)}$ possible combinations. This is because, the added rows would need to be in columns 6 or 11. For each added row we have three choices: either choose column 6, choose column 11 or choose neither. Now consider the possible rows added below 2; there are 4^2. The following picture might be useful:



Possible rows after 10 Possible rows after 5 Possible rows after 2

So there are $(2^{(10-5-1)})(3^{(5-2-1)})(4^2)$ primitive cycles using exactly columns $3, 6,$ and 11 if the order of the columns is not counted. To count order we simply multiply by $2!$.

In general, let λ be a strict partition of the cycle length, ℓ. So $\lambda = \{\lambda_1, \dots \lambda_n\}$ where $\lambda_1 > \lambda_2 > \cdots \lambda_n$ and $\sum_{i=1}^{n} \lambda_i = \ell$. Also, let $k_\lambda = n$ be the number of parts in the partition. We find that the number of primitive 2-ball juggling patterns of length ℓ is

$$\sum_{\lambda \vdash \ell} (k_\lambda - 1)! \prod_{i=1}^{k_\lambda} (i+1)^{\lambda_i - \lambda_{i+1} - 1}$$

where the sum is over all strict partitions λ of ℓ and $\lambda_{k_\lambda + 1}$ is defined to be 0. Since we are talking about *cycles*, the factor $(k_\lambda - 1)!$ accounts for re-ordering the k_λ chosen columns, else we would multiply by $k_\lambda!$. More information about primitive juggling patters can be found in [3].

2.3 Adjacency Matrix Analysis

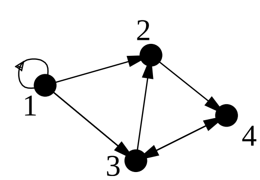
Consider a digraph with vertices labeled $1, \dots, k$. The *adjacency matrix* $A = (a_{ij})$ is the $k \times k$ matrix, where

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge from } i \text{ to } j \\ 0 & \text{if there is not an edge from } i \text{ to } j \end{cases}$$

Note an adjacency matrix is dependent on the way in which the vertices are labeled.

The ij entry of A^n is the number of walks, of length n, from i to j. Thus we can find the number of cycles, not necessarily primitive, by inspecting the diagonal entries of the adjacency matrix. The number of cycles of length n is the trace of A^n.

Example: Consider the following digraph:



Its adjacency matrix is

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

The trace of A is 1, and it is easy to see that there is only one cycle of length one, passing through the vertex 1. Now consider

$$A^3 = \begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

The trace of A^3 is 4; therefore there are 4 cycles of length 3. From the matrix, we see there is actually one cycle of length 3 starting at each vertex.

If v is a vector in a k-dimensional vector space, then the i-th component of Av is the sum of all possible edges leaving i. We will use the following notation:

$$(Av)_i = \sum_{i \to j} v_j \tag{2.1}$$

In the previous example we have:

$$Av = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = (v_1 + v_2 + v_3, v_4, v_2 + v_4, v_3)$$

The vector v is an eigenvector with eigenvalue λ if

$$\sum_{i \to j} v_j = \lambda v_i \quad \text{for all } i.$$

We will now analyze the 2-ball juggling digraph using this tool. Consider the sub-digraph, $D_n = \{(a, b) \in \mathbf{N}^2 \mid 0 \leq a, b \leq n - 1\}$. Note that we showed in section 2.1.1 that every cycle in the 2-ball digraph of length at most n will actually occur in D_n. We label the points of D_n left to right, top to bottom, to form its adjacency matrix, A_n. The first few examples are:

$$A_1 = [1] \qquad A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad A_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Our goal in this section is to find the number of (not necessarily primitive) 2-ball juggling patterns of length n by calculating the trace of A_n^n. The trace will be the sum of the n-th powers of the eigenvalues of A, counting multiplicities. First, we will calculate the characteristic polynomial of A_n.

Theorem 2.3.1. *The characteristic polynomial of A_n is*

$$f_n(x) = (-1)^n x^{n^2-n}(x^n - \textstyle\sum_{i=0}^{n-1} 2^i x^{n-i-1}).$$

Proof. Consider each vector $v \in \mathbf{C}^{n^2}$ as a labeling of the vertices of D_n, from left to right, top to bottom. Using 2.1, an eigenvector for A_n with eigenvalue λ has the form:

$$\begin{array}{ccccc} a_0\bullet & a_1\bullet & \cdots & a_{n-2}\bullet & a_{n-1}\bullet \\ \lambda a_0\bullet & \lambda a_1\bullet & \cdots & \lambda a_{n-2}\bullet & \lambda a_{n-1}\bullet \\ \vdots & \vdots & & \vdots & \vdots \\ \lambda^{n-2}a_0\bullet & \lambda^{n-2}a_1\bullet & \cdots & \lambda^{n-2}a_{n-2}\bullet & \lambda^{n-2}a_{n-1}\bullet \\ \lambda^{n-1}a_0\bullet & \lambda^{n-1}a_1\bullet & \cdots & \lambda^{n-1}a_{n-2}\bullet & \lambda^{n-1}a_{n-1}\bullet \end{array} \qquad (\star)$$

where $a_0, \ldots, a_{n-1}$ are arbitrary. The labeling is not complicated since each elevated state had only one edge leaving it, namely the state directly below. Applying formula 2.1 to the ground states produces the following necessary and sufficient conditions on $(\star)$ so that it is an eigenvector with eigenvalue λ:

$$
\begin{aligned}
\lambda(\lambda^{n-1}a_0) &= \lambda^{n-1}a_0 + \lambda^{n-1}a_1 + \lambda^{n-1}a_2 + \ldots + \lambda^{n-1}a_{n-1} \qquad (2.2)\\
\lambda(\lambda^{n-1}a_1) &= \lambda^{n-1}a_0 + \lambda^{n-2}a_0 + \lambda^{n-2}a_1 + \lambda^{n-2}a_2 + \ldots + \lambda^{n-2}a_{n-1}\\
\lambda(\lambda^{n-1}a_2) &= \lambda^{n-1}a_1 + \lambda^{n-2}a_0 + \lambda^{n-3}a_0 + \lambda^{n-3}a_1 + \lambda^{n-3}a_2 + \ldots + \lambda^{n-3}a_{n-1}\\
&\vdots\\
\lambda(\lambda^{n-1}a_{n-2}) &= \lambda^{n-1}a_{n-3} + \lambda^{n-2}a_{n-4} + \ldots + \lambda^2 a_1 + \lambda a_0 + \lambda a_1 + \lambda a_2 + \ldots + \lambda a_{n-1}\\
\lambda(\lambda^{n-1}a_{n-1}) &= \lambda^{n-1}a_{n-2} + \lambda^{n-2}a_{n-3} + \ldots + \lambda a_1 + a_0 + a_1 + a_2 + \ldots + a_{n-1}
\end{aligned}
$$

This yields n equations and $n+1$ unknowns. Multiplying the i-th equation by λ and subtracting the $(i-1)$-th equation, we get:

$$\lambda^{n+1}a_i = 2\lambda^n a_{i-1} \qquad \text{for} \quad i = 1, \ldots, n-1.$$

Suppose $\lambda \neq 0$. If $a_0 = 0$, it follows that the eigenvector is the zero vector. Otherwise, we may assume $a_0 = 1$. It then follows that $a_i = (2/\lambda)^i$ for $i = 0, \ldots, n-1$.

Substituting into 2.2 we get:

$$\lambda^n(2/\lambda)^0 = \lambda^{n-1}(2/\lambda)^0 + \lambda^{n-1}(2/\lambda)^1 + \lambda^{n-1}(2/\lambda)^2 + \ldots + \lambda^{n-1}(2/\lambda)^{n-1}$$

or

$$\lambda^n - \lambda^{n-1} - 2\lambda^{n-2} - 2^2\lambda^{n-3} - \ldots - 2^{n-1} = 0.$$

Thus $x^n - x^{n-1} - 2x^{n-2} - 2^2x^{n-3} - \ldots - 2^{n-1}$ divides the characteristic polynomial, $f_n(x)$.

To finish, we need to know the multiplicity of 0 as an eigenvalue for A_n. Consider the size of the Jordan blocks of A_n. A good method for divining these sizes can be found in [4, p. 124]. Let λ be an eigenvalue of A_n. The *deficiency indices* are defined to be

$$\delta_k = \dim \ker(A_n - \lambda I)^k.$$

Now let ν_k be the number of $k \times k$ Jordan blocks for λ. Then

$$\begin{aligned}
\nu_1 &= 2\delta_1 - \delta_2 \\
\nu_k &= 2\delta_k - \delta_{k+1} - \delta_{k-1} \quad \text{for } 1 < k < n^2 \\
\nu_{n^2} &= \delta_{n^2} - \delta_{n^2-1}.
\end{aligned}$$

Suppose $\lambda = 0$ and let us now consider the case when $n = 4$. If we consider labeled graphs, as done earlier in this proof, we get the following chain of generalized eigenspaces:

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \xleftarrow{A_n} \begin{array}{c} \begin{array}{cccc} a_0 & a_1 & a_2 & a_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \\ \{\sum a_i = 0\} \\ \delta_1 = n-1 \\ \nu_1 = 0 \end{array} \xleftarrow{A_n}$$

$$\begin{array}{c} \begin{array}{cccc} b_0 & b_1 & b_2 & b_3 \\ a_0 & a_1 & a_2 & a_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \\ \left\{ \begin{array}{c} \sum a_i = 0 \\ a_0 + \sum b_i = 0 \end{array} \right\} \\ \delta_2 = 2n-2 \\ \nu_2 = 0 \end{array} \xleftarrow{A_n} \begin{array}{c} \begin{array}{cccc} c_0 & c_1 & c_2 & c_3 \\ b_0 & b_1 & b_2 & b_3 \\ a_0 & a_1 & a_2 & a_3 \\ 0 & 0 & 0 & 0 \end{array} \\ \left\{ \begin{array}{c} \sum a_i = 0 \\ a_0 + \sum b_i = 0 \\ a_1 + b_0 + \sum c_i = 0 \end{array} \right\} \\ \delta_3 = 3n-3 \\ \nu_3 = 0 \end{array} \xleftarrow{A_n}$$

$$\begin{array}{c} d_0 \quad d_1 \quad d_2 \quad d_3 \\ c_0 \quad c_1 \quad c_2 \quad c_3 \\ b_0 \quad b_1 \quad b_2 \quad b_3 \\ a_0 \quad a_1 \quad a_2 \quad a_3 \\ \left\{ \begin{array}{c} \sum a_i = 0 \\ a_0 + \sum b_i = 0 \\ a_1 + b_0 + \sum c_i = 0 \\ a_2 + b_1 + c_0 + \sum d_i = 0 \end{array} \right\} \\ \delta_4 = 4n - 4 \\ \nu_4 = n - 1 = 3 \end{array} \xleftarrow{A_n} \begin{array}{c} e_0 \quad e_1 \quad e_2 \quad e_3 \\ d_0 \quad d_1 \quad d_2 \quad d_3 \\ c_0 \quad c_1 \quad c_2 \quad c_3 \\ b_0 \quad b_1 \quad b_2 \quad b_3 \\ \left\{ \begin{array}{c} \sum b_i = 0 \\ b_0 + \sum c_i = 0 \\ b_1 + c_0 + \sum d_i = 0 \\ b_2 + c_1 + \sum e_i = 0 \end{array} \right\} \\ \delta_5 = 4n - 4 \\ \nu_5 = 0 \end{array}$$

To explain the last set of equations note that 2.1 requires

$$\begin{aligned} \sum b_i &= a_0 \\ b_0 + \sum c_i &= a_1 \\ b_1 + c_0 + \sum d_i &= a_2 \\ b_2 + c_1 + \sum e_i &= a_3 \end{aligned}$$

Coupled with the equations

$$\begin{aligned} \sum a_i &= 0 \\ a_0 + \sum b_i &= 0 \\ a_1 + b_0 + \sum c_i &= 0 \\ a_2 + b_1 + c_0 + \sum d_i &= 0 \end{aligned}$$

we are forced to take $a_0 = a_1 = a_2 = a_3 = 0$. This gives necessary and sufficient conditions on the last labeled diagram to be in the kernel of A_4^5. Since $\delta_4 = \delta_5$, i.e. $\ker A_4^4 = \ker A_4^5$, we know that $\delta_4 = \delta_k$ for all $k \geq 4$. Therefore $\nu_n = 0$ for all $k \geq 5$. Therefore there are three 4×4 Jordan blocks with eigenvalue 0.

This same argument can be extended to the general case, to show there are $n-1$ Jordan blocks with eigenvalue 0 each of size $n \times n$. Therefore 0 has a multiplicity of $n^2 - n$; so, $x^{n^2-n}|f_n(x)$. By comparing degrees we have shown up to sign that $f_n = (x)^{n^2-n}(x^n - \sum_{i=0}^{n-1} 2^i x^{n-i-1})$. We also know that the leading term of the characteristic polynomial has sign $(-1)^n$ by looking at the determinant of $A_n - xI$. Hence, $f_n = (-1)^n (x)^{n^2-n}(x^n - \sum_{i=0}^{n-1} 2^i x^{n-i-1})$. □

Since the coefficients of the characteristic polynomial are symmetric functions in the eigenvalues of A_n, we can use Newton's identities to find the trace of A_n^n.

Let e_i be the i-th elementary symmetric function of n variables $x_1, \cdots, x_n$, and $p_k = \sum_{i=1}^n x_i^k$. For example, if we let $n = 3$, then

$$\begin{aligned} e_0 &= 1 \\ e_1 &= x_1 + x_2 + x_3 \\ e_2 &= x_1x_2 + x_1x_3 + x_2x_3 \\ e_3 &= x_1x_2x_3. \end{aligned} \qquad \begin{aligned} p_1 &= x_1 + x_2 + x_3 \\ p_2 &= x_1^2 + x_2^2 + x_3^2 \\ p_3 &= x_1^3 + x_2^3 + x_3^3 \end{aligned}$$

We will denote $e_0 = 1$. Then Newton's identities say:

$$\sum_{j=0}^{k-1} (-1)^j p_{k-j} e_j + (-1)^k k e_k = 0,$$

for $k = 0, \ldots, n$.

Example 2.3.2. Let $n = 2$. The characteristic polynomial of A_2 is $x^2(x^2 - x - 2)$. By letting λ_i be the roots to the polynomial we can rewrite

$$x^2 - x - 2 = \prod_{i=1}^{n} (x - \lambda_i) = x^2 + e_1 x + e_2$$

$$\Rightarrow e_0 = 1,\ e_1 = -1,\ e_2 = -2$$

where here e_0, e_1, e_2 denote the elementary symmetric functions in the roots of the polynomial.

By Newton's identities we know that

$$p_2 e_0 - p_1 e_1 + 2e_2 = 0 \quad \Rightarrow \quad p_2 = \frac{p_1 e_1 - 2e_2}{e_0} \quad \Rightarrow \quad p_2 = 5$$

There are 5 cycles of length 2, and they are $(0,0) \to (0,0) \to (0,0)$, $(0,0) \to (1,0) \to (0,0)$, $(1,0) \to (0,0) \to (1,0)$, $(1,0) \to (1,1) \to (1,0)$, $(1,1) \to (1,0) \to (1,1)$. This counts some cycles more then once.

Let us now consider the generic case.

Theorem 2.3.3. *The number of* 2*-ball juggling patterns with period dividing* n *is* $3^n - 2^n$.

Proof. The proof goes by induction over n. Let $g_n(x) = \prod_\lambda (x - \lambda)$ where the product is taken over the non-zero eigenvalues of the adjacency matrix, A_n, for the 2-ball juggling digraph. By 2.3.1, $g_n(x) = x^n - \sum_{i=0}^{n-1} 2^i x^{n-i-1}$. Let e_i denote the i-th elementary symmetric function in the roots of g_n. So $e_i = (-1)^{i-1} 2^{i-1}$. We need to find the trace of A_n, which is just the power sum, p_n, in the roots of g_n.

In the base case, $n = 1$, Newton's identity tells us

$$p_1 e_0 + e_1 = 0 \Rightarrow p_1 = e_1 = 1 = 3^1 - 2^1.$$

Now assume $p_{n-1} = 3^{n-1} - 2^{n-1}$. Consider Newton's identity for n:

$$\sum_{j=0}^{n-1} (-1)^j p_{n-j} e_j + (-1)^n n e_n = 0$$

$$\begin{aligned}
p_n &= \sum_{j=1}^{n-1} (-1)^{j-1} p_{n-j} e_j - (-1)^n n e_n \\
&= \sum_{j=1}^{n-1} [(-1)^{j-1}(3^{n-j} - 2^{n-j})(-1)^{j-1} 2^{j-1}] - (-1)^n n (-1)^{n-1} 2^{n-1} \\
&= \sum_{j=1}^{n-1} [3^{n-j} 2^{j-1} - 2^{n-1}] + n 2^{n-1}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{3^n}{2}\left[\sum_{j=1}^{n-1}\left(\frac{2}{3}\right)^j\right] - \sum_{j=1}^{n-1} 2^{n-1} + n2^{n-1} \\
&= \frac{3^n}{2}\left[\frac{2}{3}\left(\frac{1-(\frac{2}{3})^{n-1}}{1-\frac{2}{3}}\right)\right] - (n-1)2^{n-1} + n2^{n-1} \\
&= 3^n - 3\cdot 2^{n-1} + 2^{n-1} \\
&= 3^n - 2^n.
\end{aligned}
$$

□

We have recovered, using different methods, a special case of the main theorem in [2]:

Theorem 2.3.4. *[2] The number of period-n juggling patterns with fewer then b balls is b^n.*

Therefore there are $(b+1)^n - b^n$ patterns of period n with b balls, counting rotations distinctly. As indicated in [2] we can count the number of juggling patterns for 2-balls with period exactly n, not counting rotations, by using Möbius inversion. Let $M(d)$ be the number of juggling patterns for two balls with period exactly d, not counting rotations. Then

$$3^n - 2^n = \sum_{d|n} dM(d)$$

and by the Möbius inversion formula,

$$M(n) = \frac{1}{n}\sum_{d|n}\mu\left(\frac{n}{d}\right)(3^d - 2^d),$$

where μ is the Möbius function:

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1 \\ (-1)^k & \text{if } n \text{ is the product of } k \text{ distinct primes} \\ 0 & \text{if } n \text{ is divisible by the square of some prime.} \end{cases}$$

Appendix A

Depth-First Search

This program was written in [5].

```
--input a starting vertex and period length
--outputs all cycles containing the starting vertex
Define FindStartingAt(Vertex,Period)
  Results:=[];
  Marks:=NewList(Period,NewList(Period,0));
  C:=Coordinates(Vertex)+[1,1];
  Marks[C[1],C[2]]:=1;
  Results :=[];
  Path:=[Vertex];
  Search(Period,Path,Marks,Results);
  Return Results;
End;

--test each daughter to see if, when added, will yield a new cycle,
--given a fixed starting vertex
Define Search(Period,Var Path,Var Marks,Var Results)
  L := Len(Path);
  CurrentNode := Path[L];
  D := Daughters(CurrentNode,Period);
  Foreach X In D Do
    If Marked(X,Marks) Then
      If (Len(Path)<=Period) And (Path[1]=X) Then  -- we have a cycle!
Append(Results,Path);
      End;
    Elsif Len(Path)<Period Then
```

```
      Mark(X,Marks);
      Append(Path,X);
      Search(Period,Path,Marks,Results);
    End;
  End; --Foreach
  -- remove last node from path, unmark it, and return
  Path := First(Path,Len(Path)-1);
  UnMark(CurrentNode,Marks);
End;

--finds all edges leaving a vertex
Define Daughters(L,Period)
  If L[1] = 0 Then
    Return [Tail(L)];
  Else
    Result := [];
    X := Tail(L);
    For I:=1 To Len(X) Do
      If X[I]=1 Then NextOne:=I; Break; End;
    End;
    For I := 1 To NextOne-1 Do
      Y:=X;
      Y[I]:=1;
      Append(Result,Y);
    End;
    For I:= 1 To Period Do
      Y:=X;
      Y:=Concat(Y,NewList(I,0));
      Y[Len(Y)]:=1;
      Append(Result,Y);
    End;
  End;
  Return Result;
End; -- Daughters

-- coordinates of a vertex
Define Coordinates(L)
  I := 0;
  While L[I+1]<>1 Do
    I := I+1;
```

```
  End;
  J := 0;
  While L[I+2+J]<>1 Do
    J := J+1;
  End;
  Return [I,J];
End; -- Coordinates

-- find the vertex given the coordinates  X=[X[1],X[2]]
Define CoordsToVerts(X)
  Return Concat(NewList(X[2],0),[1],NewList(X[1],0),[1]);
End; -- CoordsToVerts

-- get rid of mark on CurrentNode
Define UnMark(CurrentNode,Var Marks)
  C:=Coordinates(CurrentNode)+[1,1];
  Marks[C[1],C[2]]:=0;
End; -- UnMark

--sees if the vertex is used in the cycle already
Define Marked(X,Marks)
  C:=Coordinates(X)+[1,1];
  If Marks[C[1],C[2]]=0 Then
    Return False;
  Else
    Return True;
  End;
End; -- Marked

--marks the vertex, as so it is only used once
Define Mark(X,Var Marks)
  C:=Coordinates(X)+[1,1];
  Marks[C[1],C[2]]:=1;
End; -- Mark

--print cycles
Define PPrint(L)
  Foreach X In L Do
    PrintLn(X);
  End;
```

```
End; -- PPrint

--finds all cycles of a given length N, from all possible starting
--vertexes without repeats.
Define Test(N)
  D:=Daughters([1,1],N);
  TestedSoFar:=[];
  Results:=[];
  Foreach X In D Do
    L:=FindStartingAt(X,N);
    Foreach Y In L Do
      Add:=True;
      Foreach Z In TestedSoFar Do
If Z IsIn Y Then Add:=False; Break; End;
      End;
      If Add Then Append(Results,Y); End;
    End;
    Append(TestedSoFar,X);
  End;
  Return SortedBy(Results,Function('ByLength'));
End; -- Test

ByLength(X,Y):=Len(X)<Len(Y);

--The following commands are used for the dot notation.

-- number of columns in a cycle
Define NoOfColumns(C)
  T:=0;
  Foreach X In C Do
    If X[1]=1 Then T:=T+1; End;
  End;
  Return T;
End; -- NoOfColumns

--Input: a set of cycles S and a vertex V
--Output: number of cycles from S passing through V
Define NoPassingThru(S,V)
  Return Len([C|C In S And V IsIn C]);
End;
```

```
  -- L: set of cycles, all of the same length
Define NoThruEachVertex(L)
  I:=Len(L[1]);
T:=[[NoPassingThru(L,CoordsToVerts([I,J]))|I In 0..(I-1)]|J In Reversed(0..(I-1))];
  Return T;
End;

--converts vertex notation to dot notation
Define NewCycleNotation(C)
  Result:='';
  For I :=1 To Len(C) Do
    X:=C[I];
    If X[1]=1 Then -- this is a column base, record number
      Y:= Comp(Coordinates(X),2);
      Result:=Result+Sprint(Y);
      -- now check if the next throw is to the top
      If I=Len(C) Then J:=1 Else J:=I+1 End; -- get next index
      Z:=Comp(Coordinates(C[J]),1);
      If Z=Y Then Result:=Result+'^'; Else Result:=Result+'_'; End;
    End;
  End;
  Return Result;
End;
```

Bibliography

[1] I. G. Macdonald, *Symmetric Functions and Hall Polynomials*, Clarendon Press, 1979.

[2] Joe Buhler, David Eisenbud, Ron Graham, and Colin Wright, *Juggling Drops and Descents*, American Mathematical Monthly, Volume 101, Issue 6 (Jun. -Jul., 1994), pp. 507-519.

[3] B. Tombaugh, *Juggling State Digraphs and Multi-Person Patterns*, Reed College Thesis, 2001.

[4] Morris W. Hirsch and Stephen Smale, *Differential equations, dynamical systems, and linear algebra*, Academic Press, 1974.

[5] A. Capani, G. Niesi and L. Robbiano, *CoCoA, a system for doing Computations in Commutative Algebra*, Available via anonymous ftp from `cocoa.dima.unige.it`, edition 4.0, 2000.