Counting the Number of Fixed Points in the Phase Space of Circ$_n$

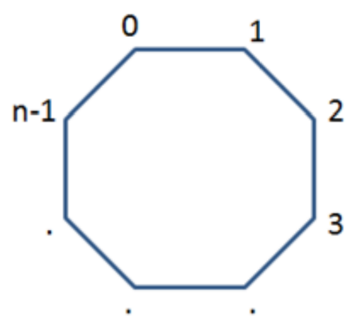
Adam Reevesman

February 24, 2015

This paper will discuss a method for counting the number of fixed points in the phase space of the Circ$_n$ graph that was described in James M. W. Duvall's article "Characterization of Fixed Points in Sequential Dynamical Systems."

Counting Fixed Points By Drawing Phase the Space

Let us begin by defining Circ$_n$. The **Circ$_\mathbf{n}$ Graph** is a graph on n vertices $(v_0, v_1, v_2 \ldots v_{n-1})$ such that each v_i is connected by an edge only to the vertices v_{i-1}, v_{i+1}. The following is a diagram of Circ$_n$.



So, we have a graph, Circ$_n$. In order to make this an SDS we need a few more things. Let the initial states for each vertex be 0,1, and let the update order for the vertices be $(v_0, v_1, v_2 \ldots v_{n-1})$. The final element we need is a function to update the vertices. Let us use the *majority* function, abbreviated *maj*.

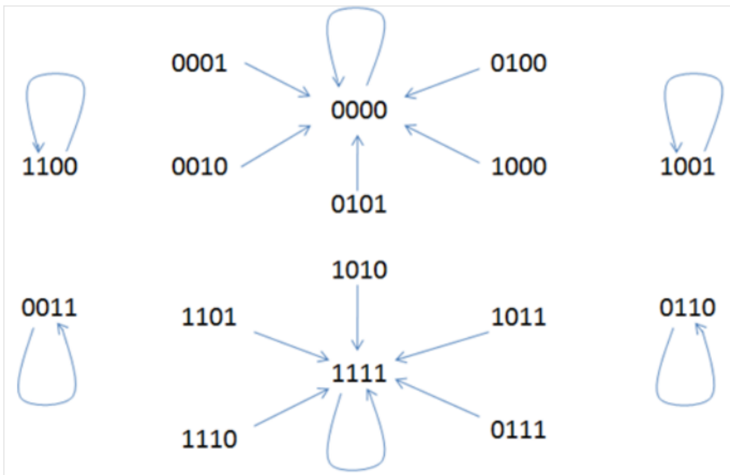
$maj_k : \{0,1\}^k \to \{0,1\}$ is a function such that the input is a k-element string of 0's and 1's and the output is either a 0 or a 1. The output is 0 if there are more 0's than 1's, and 1 if more 1's than 0's. We will only use this function when k is odd in order to avoid cases where the number of each element is the same.

To provide a few examples, $maj_5(00101) = 0$ and $maj_3(110) = 1$.

If we have a Circ_n graph, we can use the maj_3 function to update its vertices. For example, take the Circ_4 graph with the configuration 0101.

First update v_1 by taking $maj_3(v_4, v_1, v_2)$. This is the same as $maj_3(101)$, which equals 1, so v_1 is updated to 1. Next, update v_2. $maj_3(110) = 1$, so v_2 is updated to 1. In a similar fashion, we update v_3 and v_4 to result in one system update where 0101 becomes 1111.

The phase space of Circ_4 with the maj_3 function is shown below.



As we can see from this phase space, there are 6 fixed points: 1100, 0011, 0000, 1111, 1001, and 0110. In order to figure out how many fixed points there are, we first had to generate the phase space. This requires a large amount of work, that increases by a power of 2 for a step from Circ_n to Circ_{n+1}. If we want to know how many fixed points a Circ_n graph has without having to draw the phase space, we need some other method.

Counting Fixed Points With the Fixed Points Graph

By the title of this section, it is safe to assume that we will be creating something called a fixed points graph. So, what is this graph?

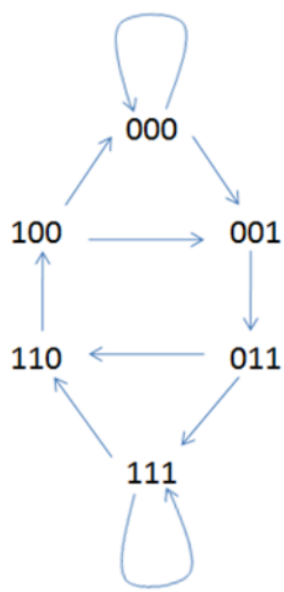
Before we can make the Fixed Points Graph, we need to define another term. A **local fixed point** a neighborhood that does not change the state of the vertex being updated when the function is applied to it. For example, if v_i is being updated and we have that the states of v_{i-1}, v_i, v_{i+1} are all 0, then the state of v_i remains 0 after being updated because $maj_3(000) = 0$. So, 000 is a local fixed point.

In every neighborhood on the graph Circ_n, there are 2 possible states for each vertex, and 3 vertices. So there are $2^3 = 8$ possible neighborhoods. We can check too see which of those are local fixed points, and display the results in the following table.

$\mathbf{v_{i-1}v_iv_{i+1}}$	$\mathbf{maj_3}$	**local fixed point?**
000	0	yes
001	0	yes
010	0	no
011	1	yes
100	0	yes
101	1	no
110	1	yes
111	1	yes

Using this information, we can construct a directed graph G, the Fixed Point Graph. The vertices of G are the neighborhoods that are the local fixed points.

To draw G, first choose one of the vertices. Look at the last two digits of this vertex. Then, look for other vertices that begin with those two digits, and draw a directed edge from the original vertex to the new ones. For example, look at 000 as a vertex of G. The last two digits are 00. The vertices that begin with 00 are 000 and 001, so one directed edge is from 000 to itself, and another from 000 to 001. Repeat this process for all of the vertices. After completing this process, we get the following graph.



From this graph, we can construct an adjacency matrix.

An **adjacency matrix** of G is a matrix created by listing out all of the vertices in the rows and columns. If the i^{th} element in the row connects to the j^{th} element in the column, then put a 1 in entry i, j of the matrix. This concept is best illustrated with an example.The following is an adjacency matrix A for the graph G.

	000	001	011	111	110	100
000	1	1	0	0	0	0
001	0	0	1	0	0	0
011	0	0	0	1	1	0
111	0	0	0	1	1	0
110	0	0	0	0	0	1
100	1	1	0	0	0	0

We have a graph G and its adjacency matrix A. Now we need some way to relate them.

Claim: If a graph λ has an adjacency matrix M, then the number of paths from vertex v_a to vertex v_b in λ that have length r is $(M^r)_{ab}$, meaning the entry in the a^{th} row and b^{th} column of M^r.

Proof. By Induction on r.

Base Case: $r = 1$.

$M^1 = M$. By definition, M is the matrix that that displays the number of paths from v_a to v_b in the entry in the a^{th} row and b^{th} column.

Inductive Step: Assume that there is some t for $1 \leq t \leq r$ such that $(M^t)_{ab}$ displays the number of paths from v_a to v_b of length t in λ.

Then, we can show that $(M^{t+1})_{ab}$ must display the number of paths from v_a to v_b that have length $t+1$. Let P_{ab}^{t+1} denote the total number of such paths. Assume that there is a possible intermediate vertex v_c on the certain paths from v_a to v_b.

$$P_{ab}^{t+1} = \sum_{v_c \in V(\lambda)} P_{ac}^{t} \cdot P_{bc}^{1}$$

This is because if there is a vertex that is length t away from v_a, there are P_{ac}^t ways to get from v_a to v_c. If v_c is distance 1 from v_b, there are P_{bc}^1 ways to get from v_b to v_c. Multiplying these together gives the number of paths from v_a to v_b that go through v_c. This happens for every intermediate vertex that v_c can be, so we add all of these products together.

By our inductive step, we can make the following substitution to get

$$P_{ab}^{t+1} = \sum_{v_c \in V(\lambda)} (M^r)_{ac} \cdot (M^1)_{bc}$$

By matrix multiplication, is above expression is equal to

$$(M^{r+1})_{ab}$$

So, the number of paths from v_a to v_b that have length r is $(M^r)_{ab}$ for $r \geq 1$. $\square$

The reasoning for creating the local fixed point graph is because a cycle of length in G corresponds to a fixed point in Circ_4. So, we want to count the number of paths from v_a to v_a that have length 4 in G. We can use the result of our proof to count the number of fixed points in the phase space of Circ_4 with the maj_3 function.

If we have the adjacency matrix of G,

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

then by matrix multiplication,

$$A^4 = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 2 & 2 & 1 \end{bmatrix}$$

The entries $(A^4)_{aa}$, (the diagonal entries) in this matrix are the ones that mean that there is a path from vertex v_a to itself, because fixed points begin and end at the same point. The sum of all of the diagonal entries is $1{+}1{+}1{+}1{+}1{+}1 = 6$, which means there are 6 fixed points.

This is exactly what the phase space showed, but we did not have to draw it out. For large values of n, this method is much more convenient because phase spaces become very large, but the size of the neighborhood is always 3 in Circ_n.