Mi Yu

SDS Writing Up Week 1

Our topic is about Sequential Dynamical System. However, before we begin the talk, let us play a game first. This game is called the game of life, sometimes it will just be referred as Life. The rule is following(The following rules are cited from Wikipedia):
We have a world of an infinite two-dimensional orthogonal grid of square cells. The cells can be in two states—either be dead or alive. We will use black color to represent the alive cells and white color to represent the dead cells. The lives of the cells are determined by its eight neighbors, with the following rules

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.

2. Any live cell with two or three live neighbors will still be alive.

3. Any live cell with more than three live neighbors will die, as if caused by over-population.

4. Any dead cell with three live neighbors will be alive. We can consider that as reproduction.

This game will show gorgeous patterns later. However, before we go further into the game, we find ourselves very confused because we have not yet formally encounter such problems associating changing patterns with time. We need to introduce the concept of Dynamical System to go further into the game of life. There are two types of dynamical system: Synchronous Dynamical System and Sequential Dynamical System. There are four basic components for a Sequential Dynamical System as following(The following rules and examples are made and cited from Henning S. Mortveit and Christian M. Reidys' "An Introduction to Sequential Dynamical Systems"):

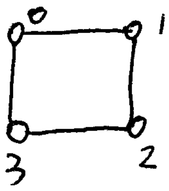
1. a finite graph Y

2. a state for each vertex v

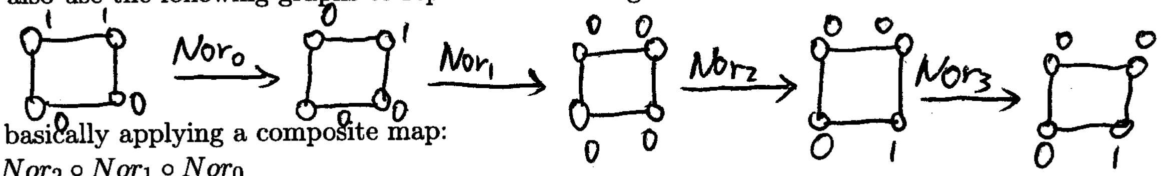
3. a function F_v, for each vertex v

4. an update order of the vertices

The components for a Synchronous Dynamical System are almost the same as a Sequential Dynamical System except it does not have an update order since based on its name, we probably already guessed that a Synchronous Dynamical System means the changes happen at the same time. In order to demonstrate the concepts of the two different systems and the difference between the two systems, let us make two examples. First, let us look at the following graph, which we denoted $Circ_4$, with vertices 0,1,2,3.



We assign each vertex a state from the set $\{0,1\}$. We write $x = \{x_1, x_2, x_3, x_4\}$ as the system states, and we will define the following function $K^3 \to K$:
$nor_3(x,y,z) = (x+1)(y+1)(z+1)$. nor function is basically an easy way to say not or. In our first quarter intro to higher math class, we already learned about the truth table. We know that for or, the only case when it is false is that when all the things are false. Since it is not or here, basically if we have one true thing this will become a false statement. Here 1 represents true and 0 represents false. If we have either x,y,z equals to 1, the entire thing will become 0. If all of them equal to 0, the entire thing will become true, which is 1. Let us defined the new function Nor as following:
$Nor_i : K^4 \to k^4$, for $0 \le i \le 3$, by
$Nor_0(x_0, x_1, x_2, x_3) = (nor_3(x_3, x_0, x_1), x_1, x_2, x_3)$
$Nor_1(x_0, x_1, x_2, x_3) = (x_0, nor_3(x_0, x_1, x_2), x_2, x_3)$
$Nor_2(x_0, x_1, x_2, x_3) = (x_0, x_1, nor_3(x_1, x_2, x_3), x_3)$
$Nor_3(x_0, x_1, x_2, x_3) = (x_0, x_1, x_2, nor_3(x_2, x_3, x_0)$
We also assign the ordering $\pi = (0,1,2,3)$. These rules are not as complicated as they look like. These rules are basically saying that if we have four states for the four vertices, we look at the state of the first vertex and its neighbors, and we can determined the first vertex's state after update. Then, we look at the second vertex and its neighbors, and then we can determined the second vertex's state after the update. The update rule is that unless the vertex and its neighbors are all 0s, the vertex will turn into 0. Let us assign a initial state for the graph. If we have the state for our above graph as $x(x_0, x_1, x_2, x_3) = (1,1,0,0)$, then applying our rules, we will have:
$(1,1,0,0) \xrightarrow{Nor_0} (0,1,0,0) \xrightarrow{Nor_1} (0,0,0,0) \xrightarrow{Nor_2} (0,0,1,0) \xrightarrow{Nor_3} (0,0,1,0)$
We can also use the following graphs to represent the changes:

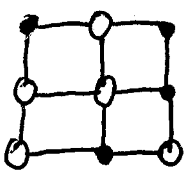


We are basically applying a composite map:
$Nor_3 \circ Nor_2 \circ Nor_1 \circ Nor_0$
By now we can recognize that this is a Sequential Dynamical System. We usually write the above map as $[(Nor_i, circ_4)_i, (0,1,2,3)]$ or $[Nor_{circ_4}, (0,1,2,3)]$. The above result then can be written as $[Nor_{circ_4}, (0,1,2,3)](1,1,0,0) = (0,0,1,0)$. Let me also defined the following two terms for Sequential Dynamical System, which later I will refer as SDS. The orbit is that if we applied a map repeatedly on a initial state, all the results until it repeats will be called an obit. For example, on the above SDS, if we apply the map repeatedly, we will get $(1,1,0,0), (0,0,1,0), (1,0,0,0), (0,1,0,1), (0,0,0,0), (1,0,1,0), (0,0,0,1), (0,1,0,0)$, and $(0,0,1,0)$, which then repeats. We call it an orbit. The phase space for a SDS with map $[F_Y, W]$ is a directed graph Γ defined by:

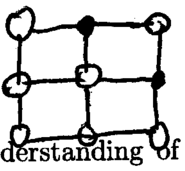
$v[\Gamma] = \{x \in k^n\}$
$e[\Gamma] = \{x, [F_Y, W](x) | x \in v[\Gamma]\}$
We already heard about graph many times in class. A phase space is a graph with all the possible states in the SDS as vertices, and its edges are connect based on the mapping of the SDS. In other words, a phase space is a visual representation of our SDS. The phase space for our above example is the following graph:

$$\begin{tikzcd}
 & & 1000 \arrow[ddd] & 0010 \arrow[l] & & 1100 \arrow[ll] \\
 & & & & 0100 \arrow[ul] & 1001 \arrow[l] \\
0011 \arrow[drr] & 0101 \arrow[dr] & & & & \\
0111 \arrow[rr] & & 0000 \arrow[r] & 1010 \arrow[r] & 0001 \arrow[uu] & 1110 \arrow[l] \\
1011 \arrow[urr] & & 1101 \arrow[u] & & & 0110 \arrow[ul]
\end{tikzcd}$$

In the next example we will use the same nor function. However, this time we will apply all the changes at the same time. We have initial state $x(x_0, x_1, x_2, x_3) = (1, 1, 0, 0)$. The x_0 position will become 0 because it is 1. Same reason, x_1 will also become 0. Since x_2 has x_1 as neighbor, x_2 will remain 0. Since x_0 is x_3's neighbor, and x_0 equals 1, thus x_3 will remain 0. Thus, our output will become $(0, 0, 0, 0)$ if we use a Synchronous Dynamical System. Last time when we used SDS the answer turned out to be $(0, 0, 1, 0)$. Comparing these two examples, people have an idea about how different a SDS is from a Synchronous Dynamical System. Now let us get back to Life. Traditionally this game is a Synchronous Dynamical System. Just reminding the audience, I will list out the rules again here. The rules for the game of life is that 1. Any live cell with fewer than two live neighbors dies. 2.Any live cell with two or three live neighbors will still be alive. 3.Any live cell with more than three live neighbors will die. 4.Any dead cell with three live neighbors will be alive. As the beginning says, we will set black color as alive and white color as dead. Let us have a 9×9 grid with the following initial state:



Apply the game of life rules, we will have:



Now we have a good understanding of the rules of the game of life. Next we want to introduce many interesting patterns in the game of life. Some of the patterns are still not fully understood by mathematicians. Also, we will talk about a concept that is closely related to the game of life — cellular automaton.