With J:
The Hodge Podge Machine

Cliff Reiter

Cellular automata are collections of cells arranged in some manner such that each cell contains a value that is updated from generation to generation according to a local rule. The Game of Life [1-5,7,15] is perhaps the best known cellular automaton. It is based upon a rectangular arrangement of cells that are either 0 or 1, along with simple rules of evolution: if a cell is 0 and has exactly 3 immediate neighbors then it becomes a 1; if a cell is 0 and has exactly 2 or 3 immediate neighbors that are 1, then the cell remains 1; otherwise, the cell becomes or remains 0. The Game of Life is incredibly intriguing, giving rise to complex behavior that is visually stimulating, mathematically interesting and, moreover, it is known to be capable of universal computation. Despite the fact that at a basic level it was designed to model alive and dead cells, it is primarily a toy model in the sense that it does not model any physical behavior well.

In this note we describe the Hodge Podge machine which was developed to model an oscillating chemical reaction. One of the earliest oscillating chemical reactions described is the Belousov-Zhabotinski reaction. There is an interesting discussion of the history of this reaction in [14]. It seems that Belousov observed a reaction that oscillated dozens of times between being yellow and clear until finally reaching equilibrium after about an hour. Belousov was basically unsuccessful in his attempt to publish this chemical reaction which is sometimes viewed as a tragic rejection of a report of an unknown phenomenon [10,14]. Zhabotinski was successful in further researching the reaction and publishing that work. Since then, oscillating chemical reactions have become well known.

Gerhardt and H. Schuster [6], coined the term Hodge Podge Machine and introduced that automaton to model an oscillating chemical reaction. We will use a slight modifications of their automaton to facilitate using general scripts [12] designed for use with Fractals, Visualization and J [11] and the Image3 Addon [13]. In particular, these scripts implement automata where cells take values between 0 and 1. We will see a great number of different sorts of self-organizing and oscillating behavior, including dramatic spiral waves that evolve from random configurations. Other references that put this automaton and reaction into a scientific context include [8-9].

The General Idea of the Hodgepodge Rules

While the Hodgepodge rules are meant to describe a chemical reaction, it is convenient to describe the process in terms of healthy, infected and ill states, as per the original authors [6]. They used states numbered 0 to *n* where *n* was often around 100. Cells will contain a real valued between 0 and 1, although we will also maintain some of the discreteness of their process.

Very generally: healthy cells have a value of zero and may become infected (or ill) if a sufficient number of their neighbors are ill or infected; infected cells have values strictly in between 0 and 1 and typically tend toward becoming more infected (larger) or ill; an ill cell has value 1 and becomes healthy (but not immune) at the next step.

The rule can be applied using a variety of neighborhoods. We will describe a version that uses 3 by 3 neighborhoods on a rectangular arrangement of cells. Thus, we can describe the local behavior by describing the behavior on neighborhoods where the center cell is of each of the three types. While neighborhoods are literally 3 by 3 patches of cells, for convenience, we may view neighborhoods as consisting of a list of 9 values (from the allowed interval of 0 to 1) where the center value corresponds to the center of the 3 by 3 neighborhood.

The Hodgepodge Rules in J

Sample neighborhoods that allow us to illustrate the rules will be created. These neighborhoods are shown in the columns of `nei` below. The function `case` distinguishes the neighborhoods as healthy, infected or ill, by resulting in 0, 1, or 2 respectively by suitable tests on the center cells (which correspond to positions with index 4).

```
   t=:?.3 9$3

   nei=:(t=1)>.(0=t)*(?.3 9$0)

   (i.9),.|:nei
0 0.567446         0        1
1        0  0.568603  0.91203
2        0         0        0
3        1         1        1
4        0         1  0.63858
5 0.708477         1 0.463438
6        0         0 0.871524
7        1         1        0
8        1 0.0761661        1

   case=: (~:&0 + =&1)@(4&{)

   case 0{nei       healthy
0

   case 1{nei       ill
2

   case 2{nei       infected
1
```

Next we create functions that select the ill and infected neighbors from a neighborhood.

```
   ill=: #~ =&1

   inf=: #~ ~:&0 *. ~:&1

   ill 0{nei
1 1 1

   inf 0{nei
0.567446 0.708477

   #@inf 0{nei
2
```

Now we are prepared to begin discussing the details of the automata rules. In particular, the automata depend upon 4 parameters, `a`, `b`, `c`, and `n`. The parameters `a` and `b` are thresholds over which the number of ill and infected neighbors must pass in order for a healthy cell to becomes infected (or ill). The level of subsequent infection is the number of times the thresholds are exceeded divided by `n`. Thus, a healthy cell is updated by the reciprocal of `n` times the sum of the number of times the two thresholds are exceeded

```
   'a b c n'=:2 3 0.17 100

   forh1=: <.@:(*&(%a))@#@ill

   forh1 0{nei
1

   forh1 1 1 1 1 0 1 1 0 0
3

   forh2=: <.@:(*&(%b))@#@inf

   forh2 0{nei
0

   forhea =. [: %&n forh1 + forh2

   forhea 0{nei
0.01
```

Infected cells are updated as the sum of the parameter `c` and the average of the infected cells in the neighborhood. Thus, `c` is a bias toward a higher level of infection and might be thought of as the internal growth rate of the infection.

```
   avg=: +/ % #

   forinf=: 1: <. avg@inf + c"_

   forinf 2{nei
0.891393
```

The ill cells become healthy at the next stage. This might seem unrealistic, but it corresponds to an illness where at a certain point the body successfully beats the illness. However, no immunity results after health is restored. In the chemical reaction, one can think of the reaction as running its course and the catalyst is once again free.

```
   forill=: 0:

   forill 1{nei
0
```

We put these together in an adverb where the adverb argument specifies the four parameters and the result is a function that applies the appropriate case. A version of this adverb is defined in *automata.ijs* [11].

```
   lhodge=: 1 : 0
'a b c n'=.m.
ill=. #~ =&1
inf=. #~ ~:&0 *. ~:&1
case=. (~:&0 + =&1)@(4&{)
avg=.+/ % #
forh1=. <.@:(*&(%a))@#@ill
forh2=. <.@:(*&(%b))@#@inf
forhea =. [: %&n forh1 + forh2
forinf=. 1: <. avg@inf + c"_
forill=. 0:
forhea`forinf`forill@.case@:, f.
)
```

```
   2 3 0.17 100 lhodge 0{nei
0.01
```

```
   2 3 0.17 100 lhodge 1{nei
0
```

```
   2 3 0.17 100 lhodge 2{nei
0.891393
```

We periodically extend the boundary conditions and spread the local processing onto 3 by 3 neighborhoods and illustrate its use on a small example as follows.

```
   perext=: {: , ] , {.

   perext2=: perext"1@:perext

   hodge=: 1 : 0
3 3&(m. lhodge;._3)@ perext2
)
```

```
      ]t=: ?. 4 4$3
   0 2 2 1
   2 0 2 1
   1 2 0 2
   1 1 1 2
```

```
      ]x0=: (t=1) >. (0=t)*?.4 4$0
0.567446        0        0 1
       0 0.708477        0 1
       1        0 0.568603 0
       1        1        1 0
```

```
   2 3 0.17 100 hodge x0
0.807962     0.01     0.02    0
    0.01 0.784842     0.01    0
       0     0.02  0.80854 0.02
       0        0        0 0.02
```

Notice that the ill cells immediately became healthy and, as it happens for these parameters, all the healthy cells became infected.

Experiments

We assume that *automata.ijs* [11] has been loaded and then define an initial configuration, `x0`, that randomly has 5% of the cells ill and 5% infected while all the rest are healthy.

```
   t=:?.200 200$20

   $x0=: (t=1)>.(0=t)*(?.200 200$0)
200 200
```

We then use `show_auto` to show the evolution of the automaton. Readers are strongly encouraged to obtain the script and run the experiments which are dynamic and in color. The grayscale figures that we include only give a hint at the drama.

```
   h=: 2 3 0.16 100 hodge

   600 0 h show_auto x0
0
```

Notice the oscillating spiral waves. Figure 1 shows the configuration after 80, 180 and 600 iterations. In that figure, healthy cells are shown in white, ill in black and infected in intermediate grayscales. It takes some time for the spirals to organize, but once they do, they evolve throughout the field. Figure 2 shows the percentage of infected cells at each step during that

process which is seen to initially oscillate but eventually stays around 0.73.

A significantly different behavior occurs for the Hodge Podge Machine with the following parameters.

```
   h=: 6 2 0.01 100 hodge

   800 0 h show_auto x0
0
```

Figure 3 shows behavior after 180 500 and 800 steps. Initially the configuration evolves in an infected plasma cloud but suddenly virtually every cell goes from ill to healthy except a couple tiny fragments; they induce perfectly square growths; those lead to slightly less structured growths and eventually to random growth. Figure 4 shows the percentage of infected cells for this machine; notice that it contains spikes downward. While this example is somewhat unusual, it is intriguing because a random configuration leads to highly structured growth which leads back to random behavior, all caused by a deterministic automaton.

Figure 5 shows a frame from each of the Hodge Podge Machines with parameters 2 6 0.16 100, 4 1 0.2 100, 3 2 0.25 100, and 3 2 0.35 100. The first gives rise to highly granular behavior, the second gives rise to square patterns with strange, rough edges oriented in the same direction as the cellular lattice. The third evolves to vague square patterns oriented along the diagonals of the cellular arrangement while the fourth gives rise to globally dominant concentric structures that arise around spirals in a granular field.

Conclusion

We see that Hodge Podge Machines are simple cellular automata. These were developed to model an oscillating chemical reaction involving a catalyst. Remarkably diverse self-organizing features arise from random configurations, including oscillating spiral patterns. These machines are interesting, simple cellular automata that model physical behavior and ought to be better known.

References

[1] E. Berlekamp, J. Conway, and R. Guy, *Winning Ways For Your Mathematical Plays*, Academic Press, 1982.

[2] P. Callahan, Patterns, Programs, and Links for Conway's Game of Life, *http://www.radicaleye.com/lifepage/lifepage.html#catback*

[3] P. Callahan, What is the Game of Life? *http://www.math.com/students/wonders/life/life.html*

[4] M. Gardner, The fantastic combinations of John Conway's new solitare game of “life”, *Scientific American* 223, (1970) 120-123.

[5] M. Gardner, On cellular automata, self-replication, the Garden of Eden and the game “life”, *Scientific American*, 224 4 (1971) 112-117.

[6] M. Gerhardt and H. Schuster, A cellular Automaton describing the formation of spatially ordered structures in chemical systems, *Physica D* 36 (1989) 209-221.

[7] A. Hensel, Conway's Game of Life, *http://www.ibiblio.org/lifepatterns/*

[8] Andrew Ilachinski, Cellular Automata, *A Discrete Universe*, World Scientific, 2001.

[9] Kunihiko Kaneko and Ichiro Tsuda, *Complex Systems: Chaos and Beyond*, Springer, English translation, 2000.

[10] A. Pechenkin, How to Understand the History of the Belousov-Zhabotinsky Reaction, *http://www.ut.ee/flfi/ISPC/Pechenkin.doc*.

[11] C. A. Reiter, *Fractals, Visualization and J*, second edition, Jsoftware, 2000.

[12] C. A. Reiter, Scripts for Special Fall 2005 update to Fractals, Visualization, and J, *http://www.lafayette.edu/~reiterc/j/fvj2/index.html*.

[13] Z. X. Reiter, C. A. Reiter, Image 3 Addon, *http://www.jsoftware.com*.

[14] Steven H. Strogatz, *Nonlinear Dynamics and Chaos*, Perseus Books Publishing, LLC, 1994.

[15] J. Summers, Jason's Life Page, *http://entropymine.com/jason/life/*.

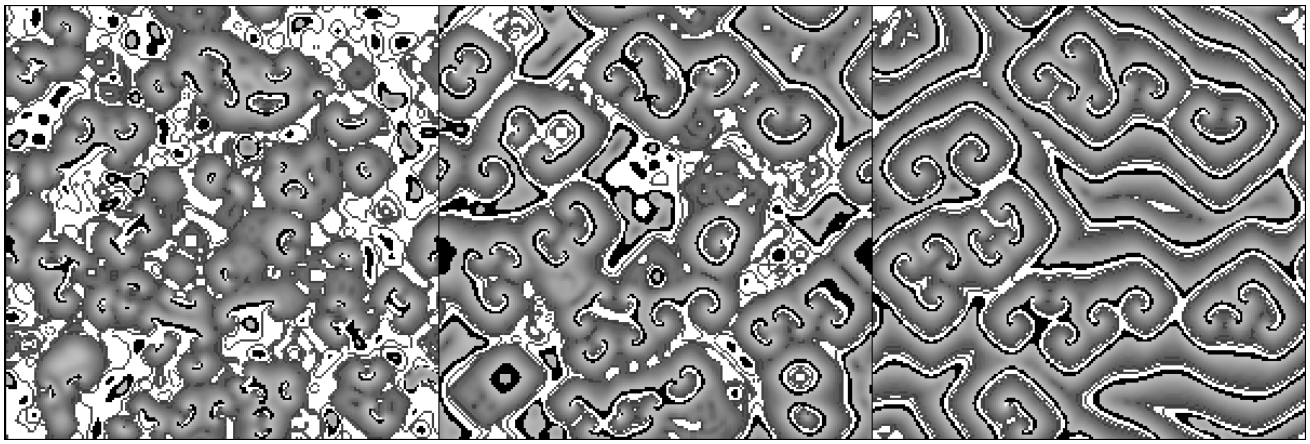

Figure 1. Evolution of the Hodge Podge Machine with parameters 2 3 0.16 100.

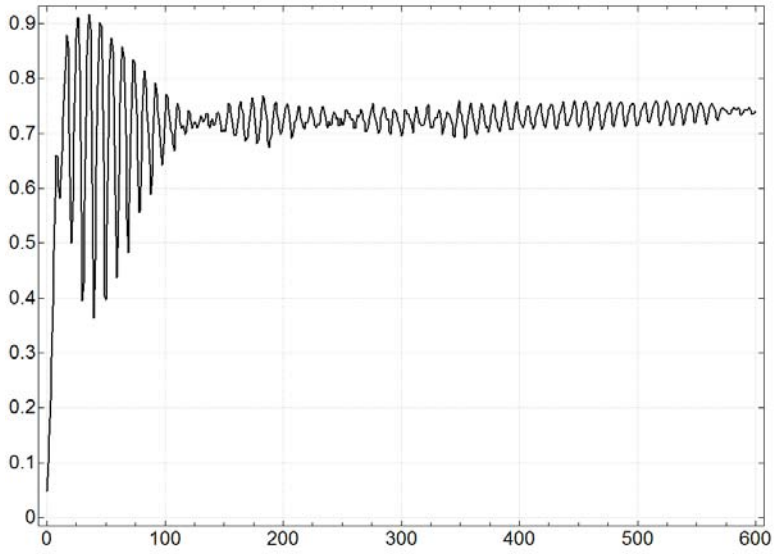

Figure 2. The Percentage of Infected for the Machine in Figure 1.

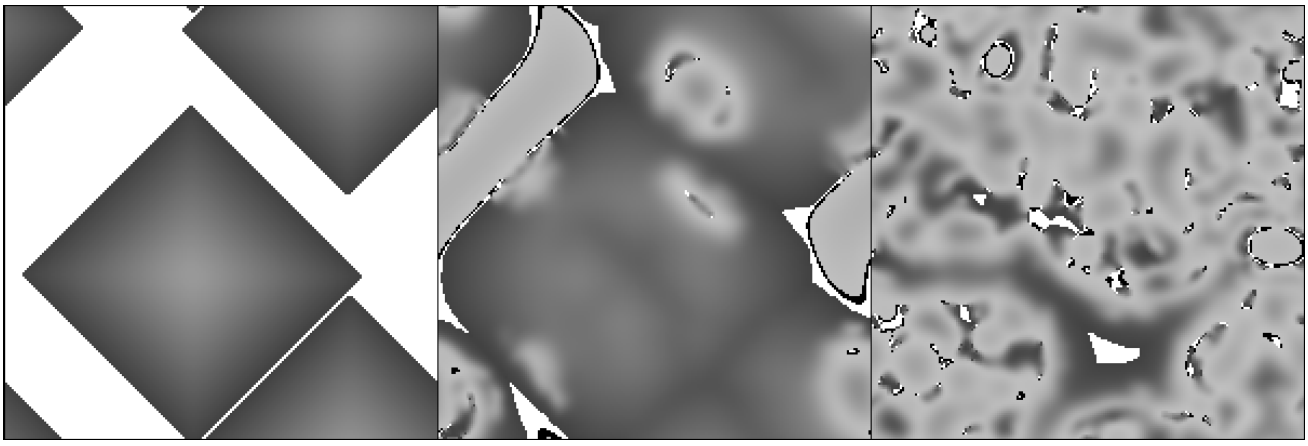

Figure 3. Evolution of the Hodge Podge Machine on 6 2 0.01 100.

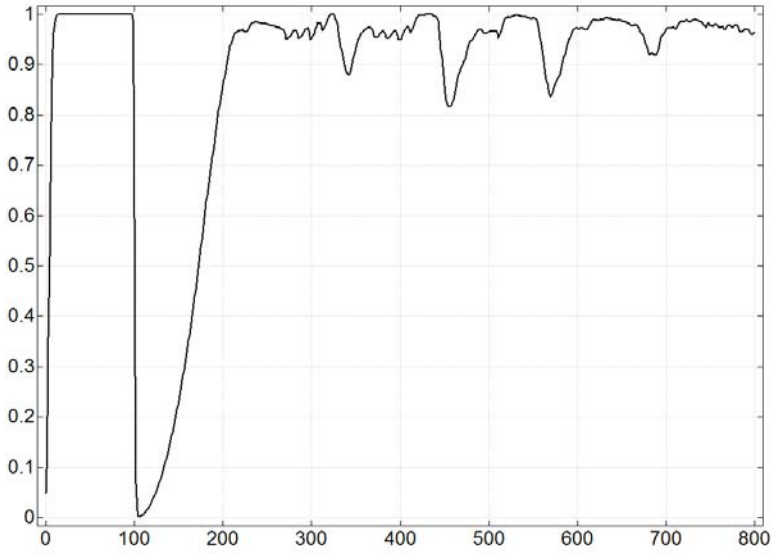

Figure 4. The Percentage of Infected for the Machine in Figure 3.

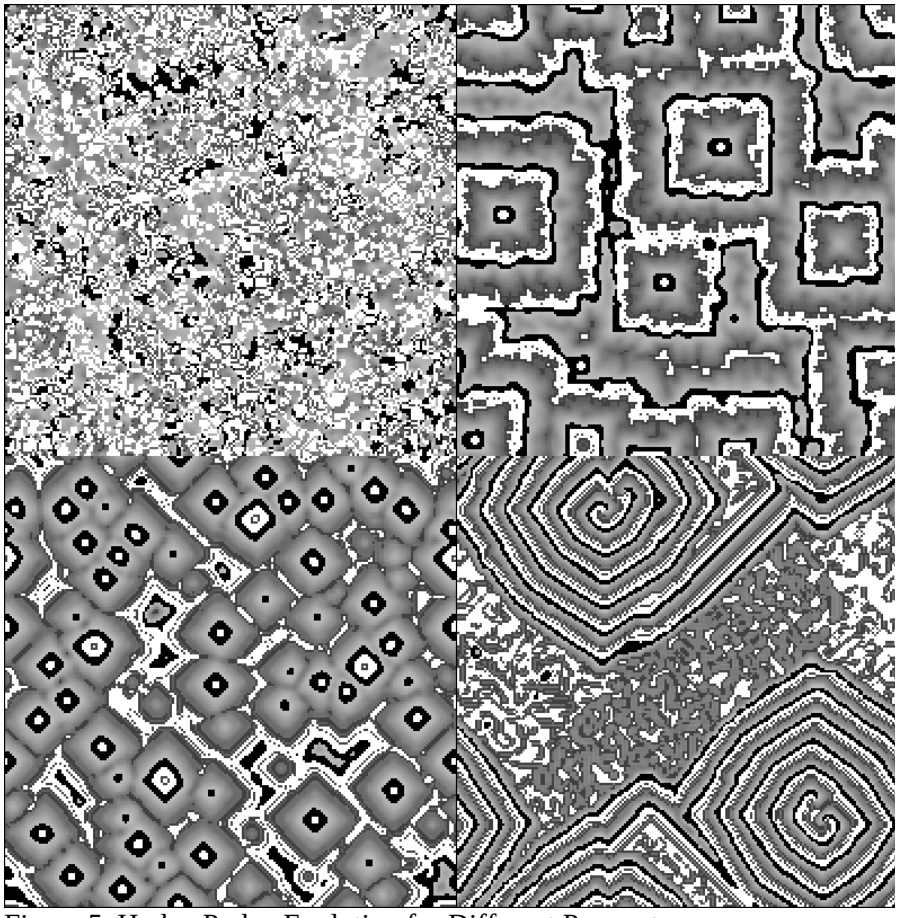
Figure 5. Hodge Podge Evolution for Different Parameters.