


















# Introduction to Complexity

## Homework 6

### Graded Part

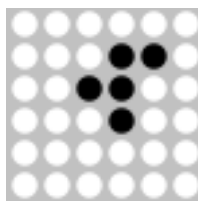
This part you will submit on the course webpage, by doing the exercises, and then clicking on the “Homework 6” link in Unit 6.8, filling out the answers, and then clicking on the “Submit” button at the bottom of the page.

For questions 1-2, use the following guide to simple stable Life patterns for reference:

3P2.1 Blinker 	4.1 Block 	4.2 Tub 	5.1 Boat 	5P4H1Y1.1 Glider 	6.2 Ship 
6.4 Beehive 	6.5 Barge 	6P2.1 Toad 	6P2.2 Beacon 	7.2 Long Boat 	7.4 Loaf 
8.7 Pond 	8.8 Mango 	8.9 Long Barge 	12.41 Half-Fleet 	14.533 Half-Bakery 	

(From <http://pentadecathlon.com/lifeNews/2006/06/collisions/trivial/trivial.ltbl?bits=8&columns=6>)

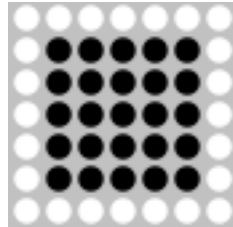
1. Using the **Mini-Life** NetLogo model, try the following initial pattern of five black cells, which Conway called the “R-pentomino”:



Put this pattern somewhere near the center of the lattice. Run the CA until the lattice has settled into a stable set of patterns. What are the final patterns you see? (Note that you may see patterns as rotated versions of the ones in the picture above. These still count as instances of the given pattern.)

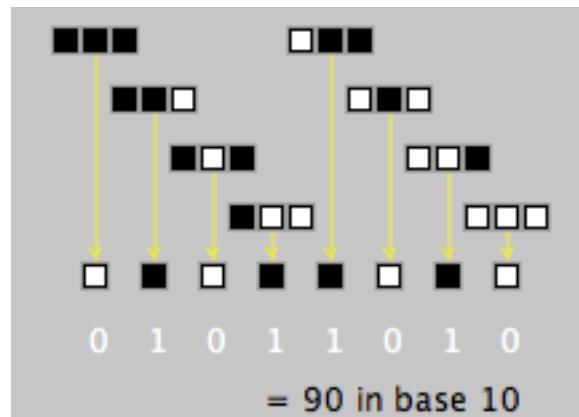
- A. Two blinkers, one beehive, and a block
- B. One beehive, three blocks, and a blinker
- C. One beehive, one mango, and a pond

2. Using Mini-Life.nlogo, create an initial pattern like the one below, placing it near the center of the lattice. Run the CA until the lattice has settled into a stable set of patterns. What are the final patterns you see? (Note that you may see patterns as rotated versions of the ones in the picture above. These still count as instances of the given pattern.)



- A. Four beehives
- B. Three blinkers
- C. One beehive, one mango, and a pond

3. Elementary CA rule 90 looks like this:



What is the Wolfram code number in base 10 of the *inverse-color* rule to rule 90, where 0s in the output states of rule 90 are changed to 1s, and vice versa? (That is, the *inverse-color* rule is **1 0 1 0 0 1 0 1**.)

- A. 85
- B. 165
- C. 205
- D. 213

4. According to the definitions given in the lectures, which Wolfram class does the behavior of ECA rule 40 fall into? (Investigate this by running ElementaryCAs.nlogo with this rule on several random initial configurations.)

- A. Class 1
- B. Class 2
- C. Class 3
- D. Class 4

5. Same question as the previous one, but this time for ECA rule 144.

- A. Class 1
- B. Class 2
- C. Class 3
- D. Class 4

6. What is the Lambda value of ECA rule 90?

- A.  $3/8$
- B.  $4/8$
- C.  $5/8$
- D.  $6/8$

7. What is the Lambda value of ECA rule 32?

- A.  $1/8$
- B.  $2/8$
- C.  $3/8$
- D.  $4/8$

8. Langton's hypothesis was that an ECA with Lambda close to 1/2 should typically have more complex behavior than an ECA with a much lower Lambda value. Is this true when you compare the typical behavior of rule 32 and rule 90?

A. Yes

B. No

## Ungraded Part

### Beginner options:

1. In the Code tab of Mini-Life, change the rules of Life to be:

- If a cell has  $< 4$  black neighbors, then turn black, otherwise turn white.

To do this, find the following line of code in the procedure `count-black-nbrs` (near the bottom):

```
if (n = 3 or (color = black and n = 2)) [set label "●"]
```

and change it to

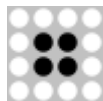
```
if (n < 4) [set label "●"]
```

Try this new rule out on some random initial lattices (using “randomize” with 12% black). What is the effect of this rule change on the behavior of the cellular automaton after several time steps?

2. In Mini-Life.nlogo's Code tab, change the same line of code you changed for question 3; this time to:

```
if (n = 2 or n = 3 or (color = black and n = 4)) [set label "●"]
```

If you start this new CA from a ‘seed’ pattern of a “block”



what kind of behavior do you see?

A. The pattern grows into symmetrical shapes, ending in a repeated cyclic pattern

B. The pattern grows into a chaotic pattern

C. The block quickly dies away

3. In the Code tab of Mini-Life.nlogo, change these three lines in the `go` procedure:

```
ifelse ticks = int ticks  
  [ask cells [count-black-nbrs]]  
  [ask cells [change-color]]
```

to the following single line:

```
ask cells [count-black-nbrs change-color]
```

This is now an *asynchronous* system, where cells (in random order, once per generation) change their colors immediately upon counting their neighbors, without waiting for them to finish their own counts. Is this new system *deterministic*: does it always calculate the same resulting patterns from the same initial conditions? How does this new system's behavior compare with that of the original Game of Life cellular automaton?

### **Intermediate options:**

1. In the Game of Life, how many different ways can a glider collide with a block, and what happens in each case? (You can save some work by thinking about symmetry!)
2. Change ElementaryCAs.nlogo to be asynchronous in the same sense as you did for the Game of Life (if you did Beginner Option 3). How does the behavior of various rules change from the synchronous version?

### **Advanced option:**

Implement a version of the Evolving Cellular Automata with Genetic Algorithms project described in Unit 6.6 and in the paper “Evolving Cellular Automata to Perform Computations: A Review of Recent Results” (linked from the Course Materials page). See what results you obtain from evolving with small one-dimensional lattices (otherwise, the computation time can get very large!).