# ECA CLASSIFICATION USING CYCLES

Dillon Shelton

Computational Discrete Math

December 2025

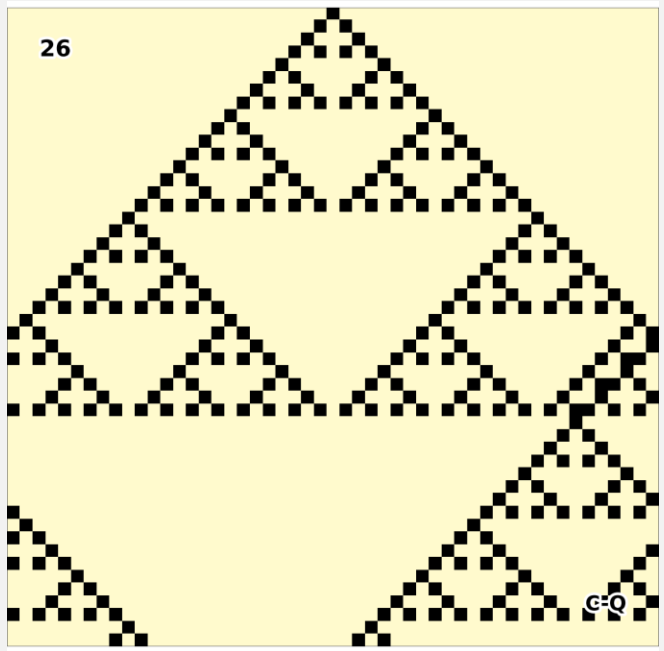
# PART I

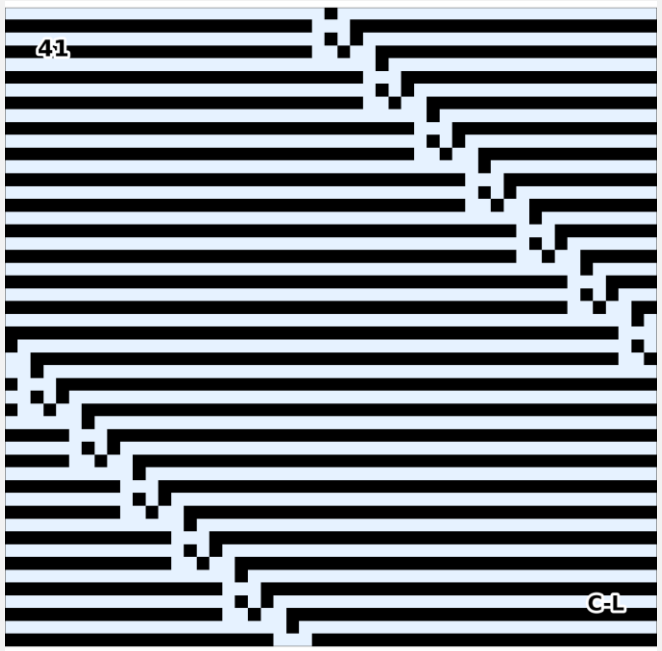
CYCLE DETECTION

# WHAT IS A CYCLE?

- We can define the function f as the global map of the ECA rule for a fixed-width string with wraparound. For some length n, we use the initial point 0…010…0 of length n.
- Since the state space is finite, a cycle is guaranteed when iterating over this initial point. We can use a few different methods to obtain it.

# EXAMPLE: N=51



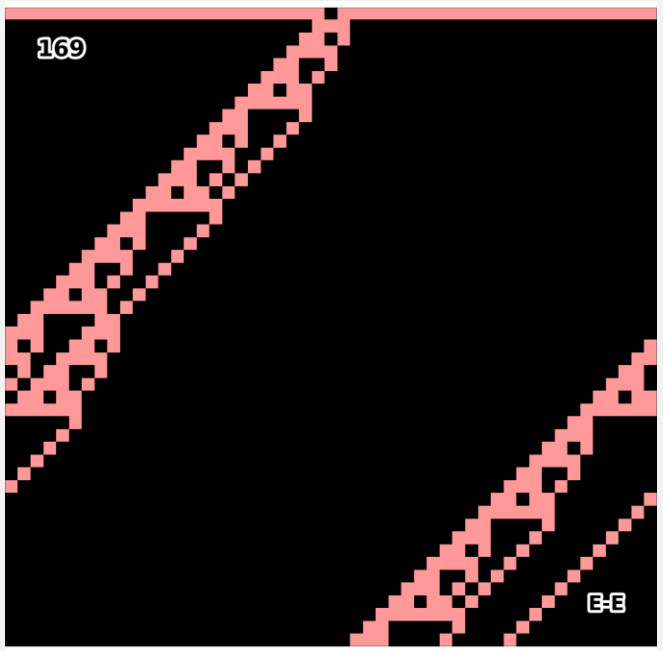




Illustration of wrap-around behavior of certain ECAs.
Top Left indicates the rule number. Bottom right shows the classification- we'll get to that later.

# WHAT ALGORITHMS ARE THERE?

## HASH-MAP

- Stores every generated state in a lookup table to check for duplicates.
- Simplest implementation, uses the least amount of functions applications.
- Cons: Memory intensive. Will struggle to detect large cycles.

## FLOYD'S TORTOISE & HARE

- Two pointers move at different speeds. A collision is guaranteed and indicated a loop.
- Uses constant memory but performs ~3x as many function evaluations as hash-map.

# WHAT ALGORITHMS ARE THERE?

## BRENT'S

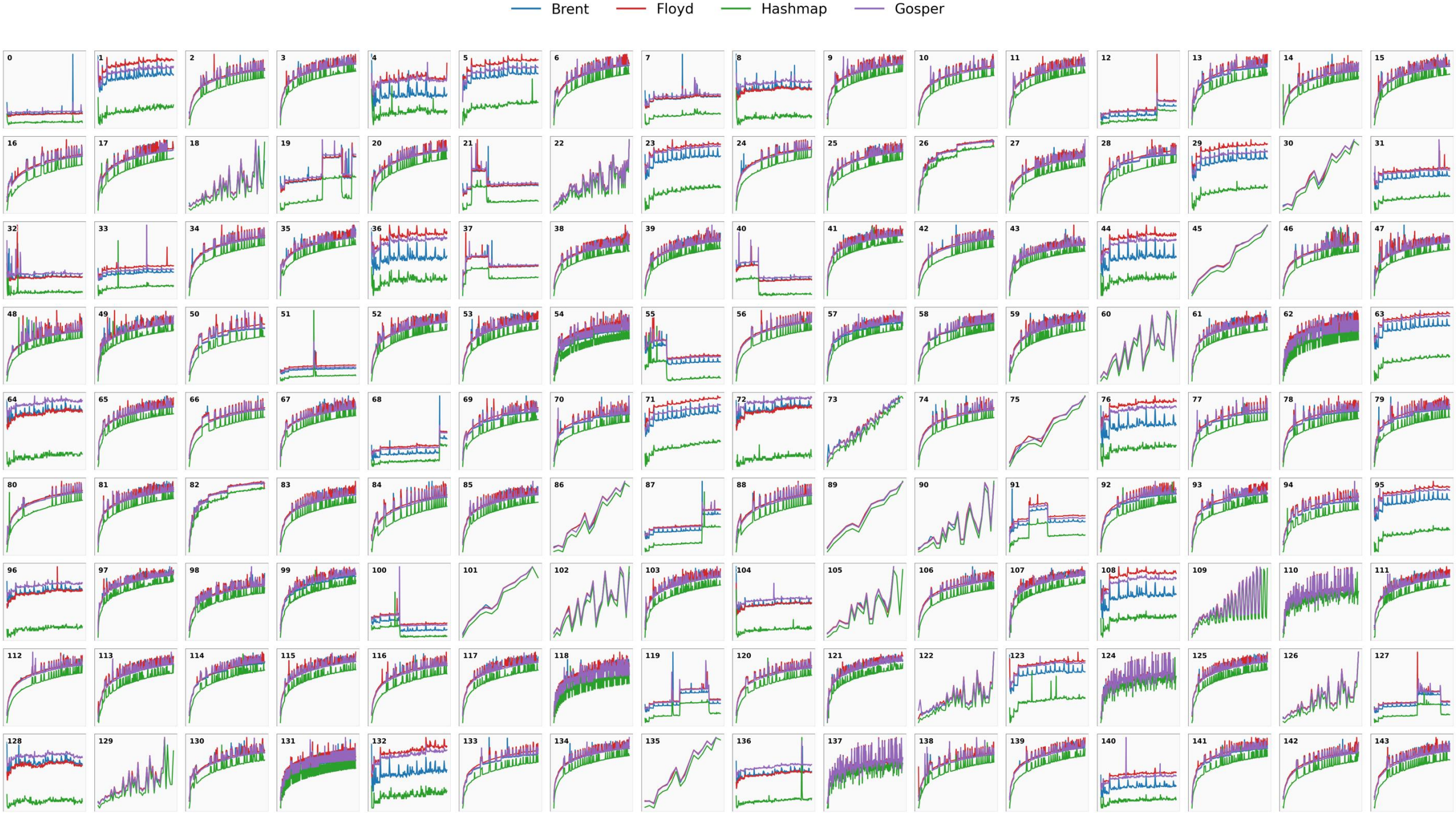
- The "slow" pointer waits stationary for exponential intervals before jumping to the "fast" pointer's location.
- Evaluates much fewer steps that Floyd with no memory needed.
- Cons: Requires secondary calculation to find transient

## GOSPER'S

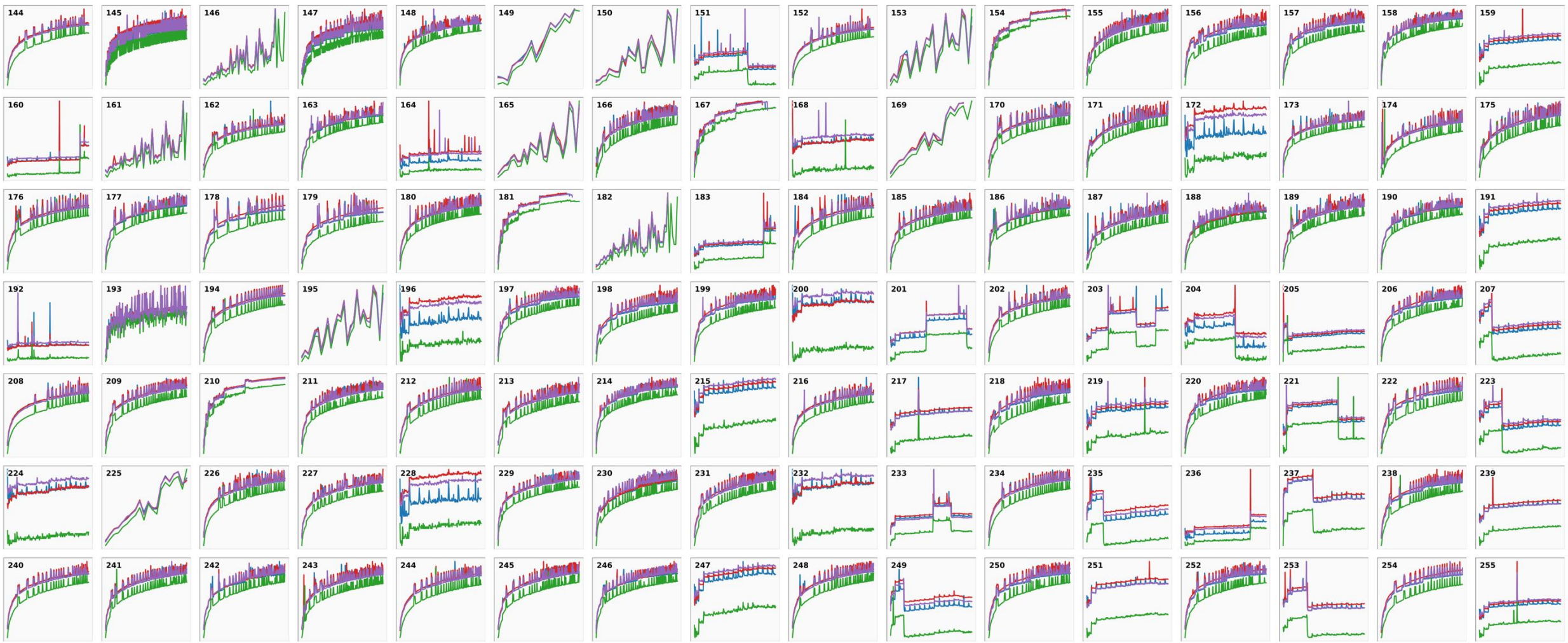
- Saves specific states (at power-of-2 intervals) in a small table for comparison.
- Detects transient and period in single pass
- Uses log(L) memory

# LETS TEST IT

- For each ECA, start from n=3 and increment n+=2 to keep the initial config symmetrical. Determine the transient and cycle using each algorithm.
- Repeat until:
  - We hit a limit (default n=500, to prevent constant or linear from eating up all my time)
  - It takes longer than TIMEOUT (default t=60s) to compute the cycle
- Plot it as **Time (log scale) vs N (linear scale)**


Brent
Floyd
Hashmap
Gosper
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143

Ponder deeply.



# RESULT

- Hash-map wins for small n as expected
- Hash-map wins for large n, unexpectedly, as I fully expected my computer to run out of memory- some rules can blow up to cycle lengths of millions within that allotted time.
- Brent's comes in second place. I use brent's for future simulations to prevent any memory related issues, even though none seem to come up.
  - In the end, hash-map can only computer one or two more values compared to the other approaches.

# PART 2

ECA CLASSIFICATION

# PERIOD LENGTH

- Before crunching the numbers, here's some intuition behind why period length might matter and how we might classify them.
- A ruleset that consistently reaches a fixed point (length = 1) will have rules that cause it to 'die out'.
- A ruleset whose period increases linearly with n probably has rules that depend on the time it takes to reach the boundary condition, but not much else.
- A ruleset whose period increase exponentially with n probably has randomness or chaoticity involved, as it near almost the entire state space because forming a cycle.

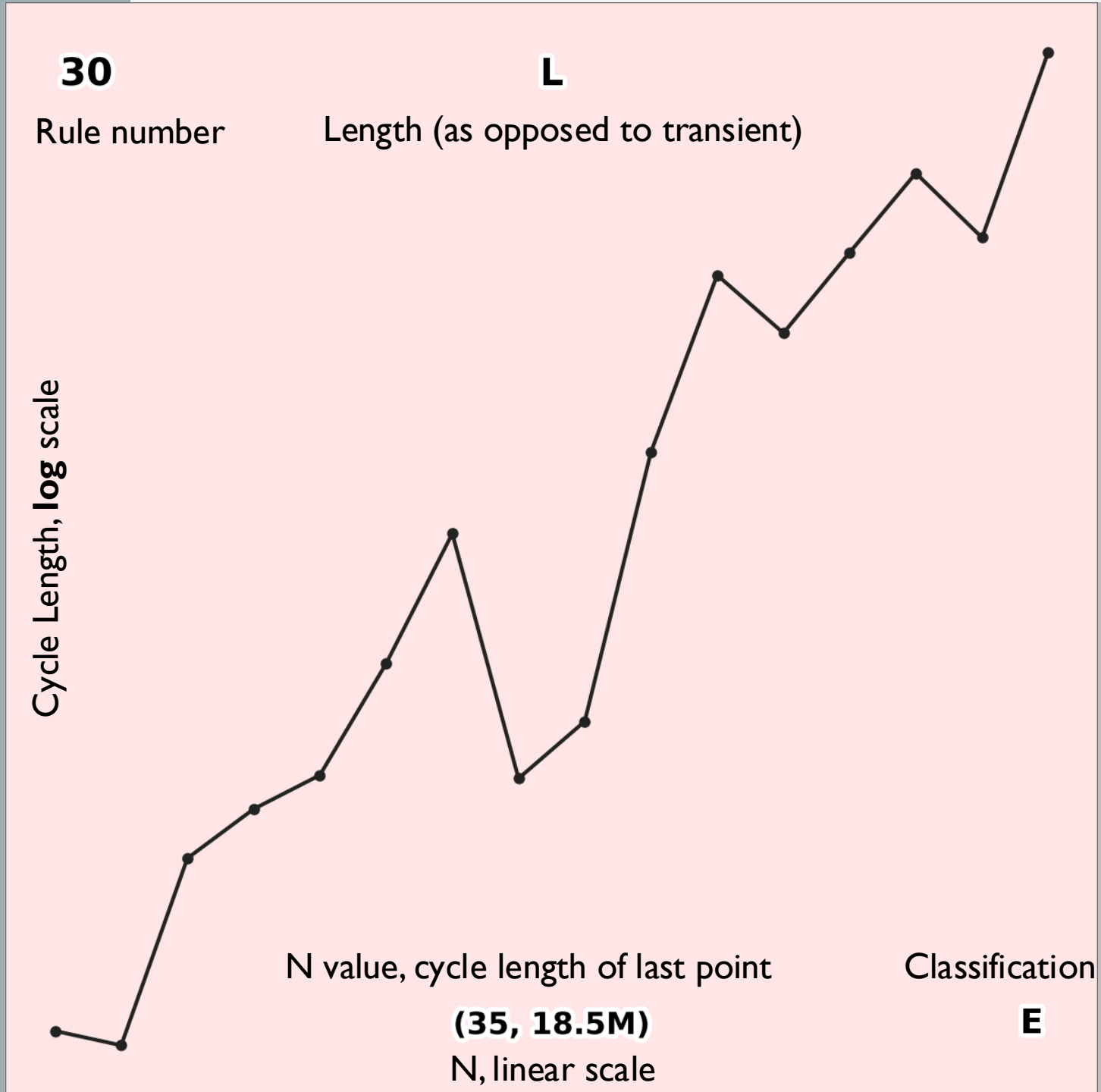
# SIMPLE CLASSIFICATION

- C – graph is about-constant.
- L – graph is about-linear
- Q – graph is about-quadratic
- E – graph is about-exponential

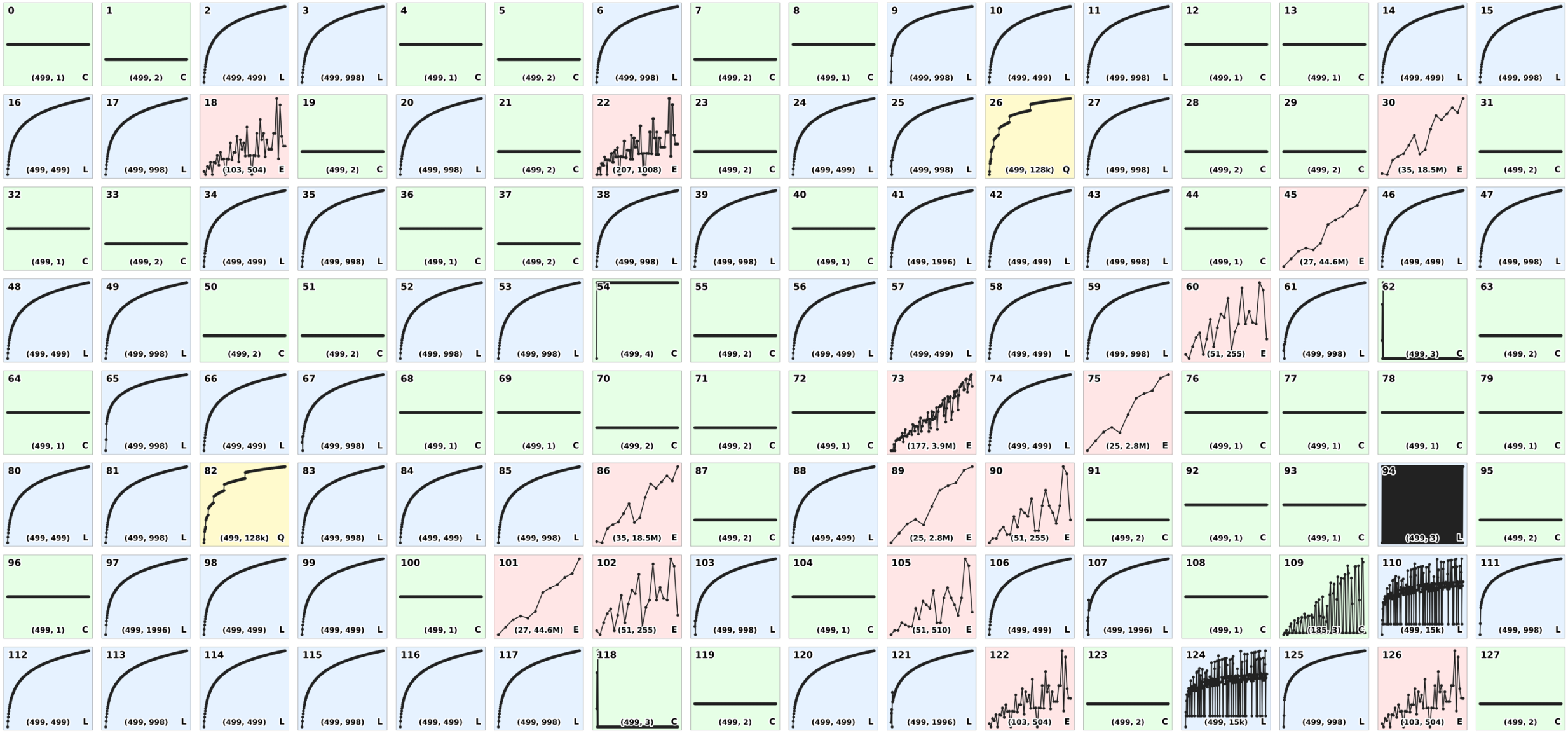
- I used sketchy heuristics to code these up. The hope is that as I run the simulations for longer periods of time, the more likely it is for each rule to fall under its ‘correct’ complexity class.

# INTERPRETING RESULTS

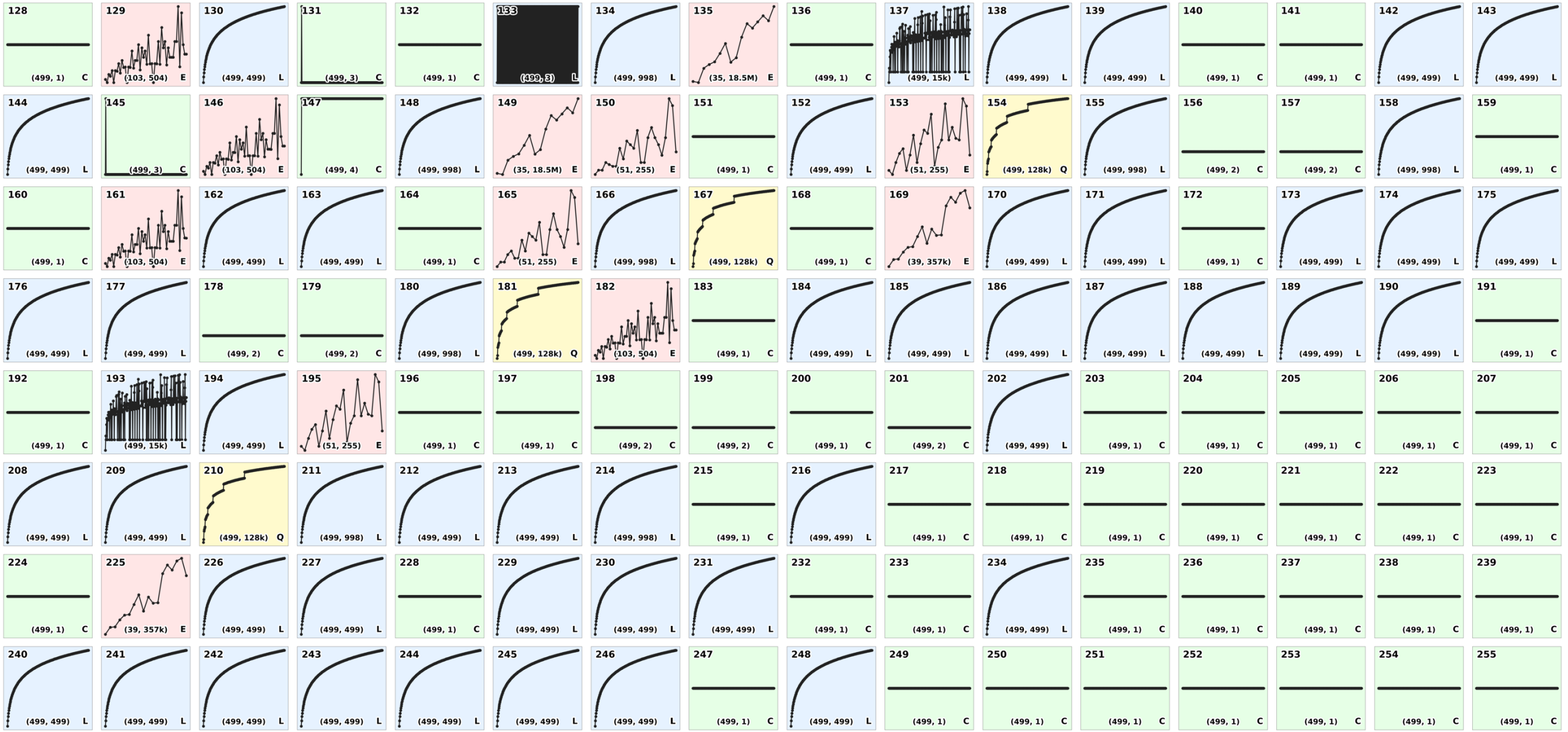
- Each N run using Brent's for up to 3 minutes
- Up to N = 499



# Cycles Growth Analysis



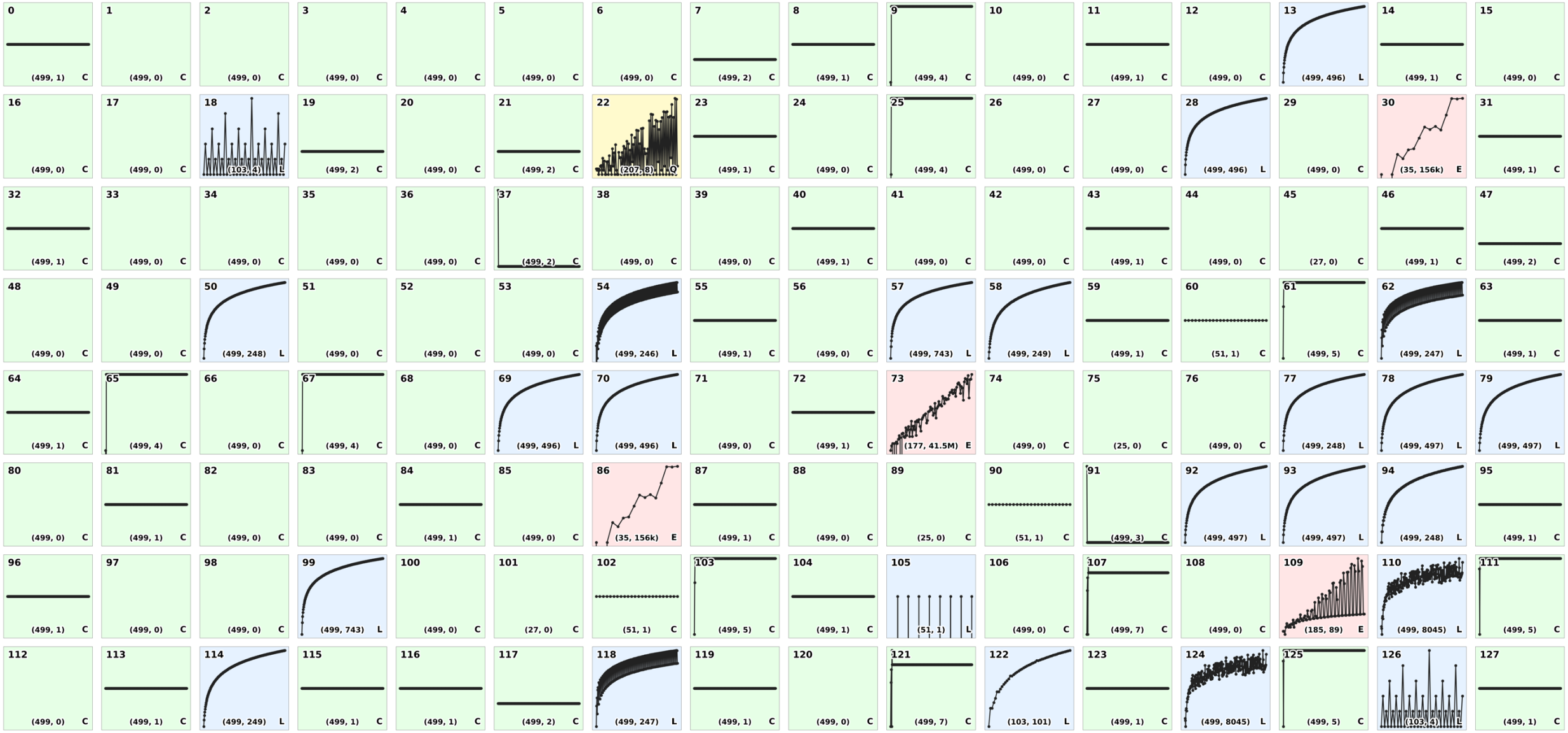
# Ponder even more deeply.

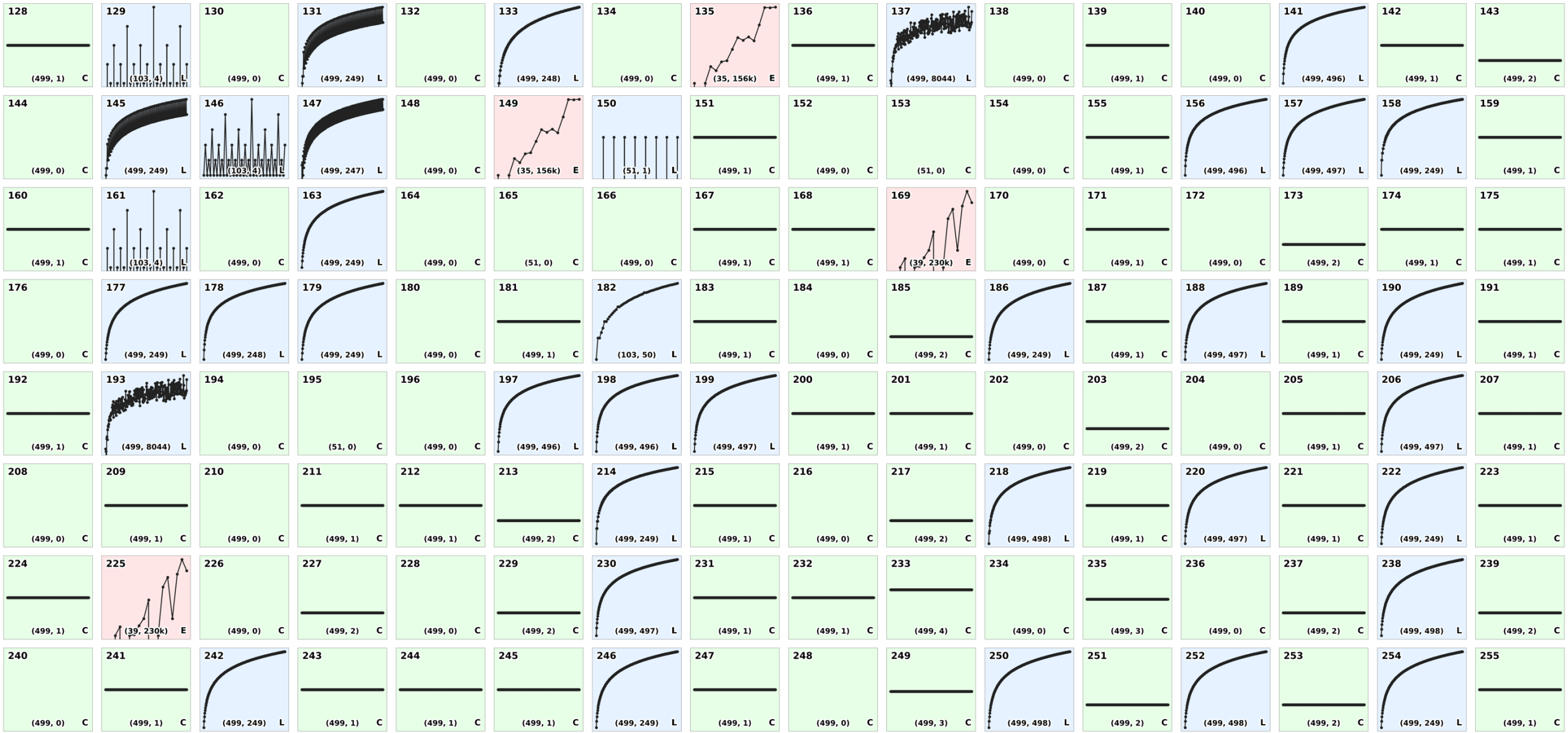


# TRANSIENT LENGTH

- One can similarly gather intuition from the transient. It is certainly less clear to me, but we can still infer order and chaos based on the constant/exponential growth.
- A linear transient implies that it takes some time to ‘saturate’ the grid- that is, to reach the wraparound zone. Afterwards, it never goes back. This is commonly seen in automata that branch out left and right initially.

# Transients Growth Analysis

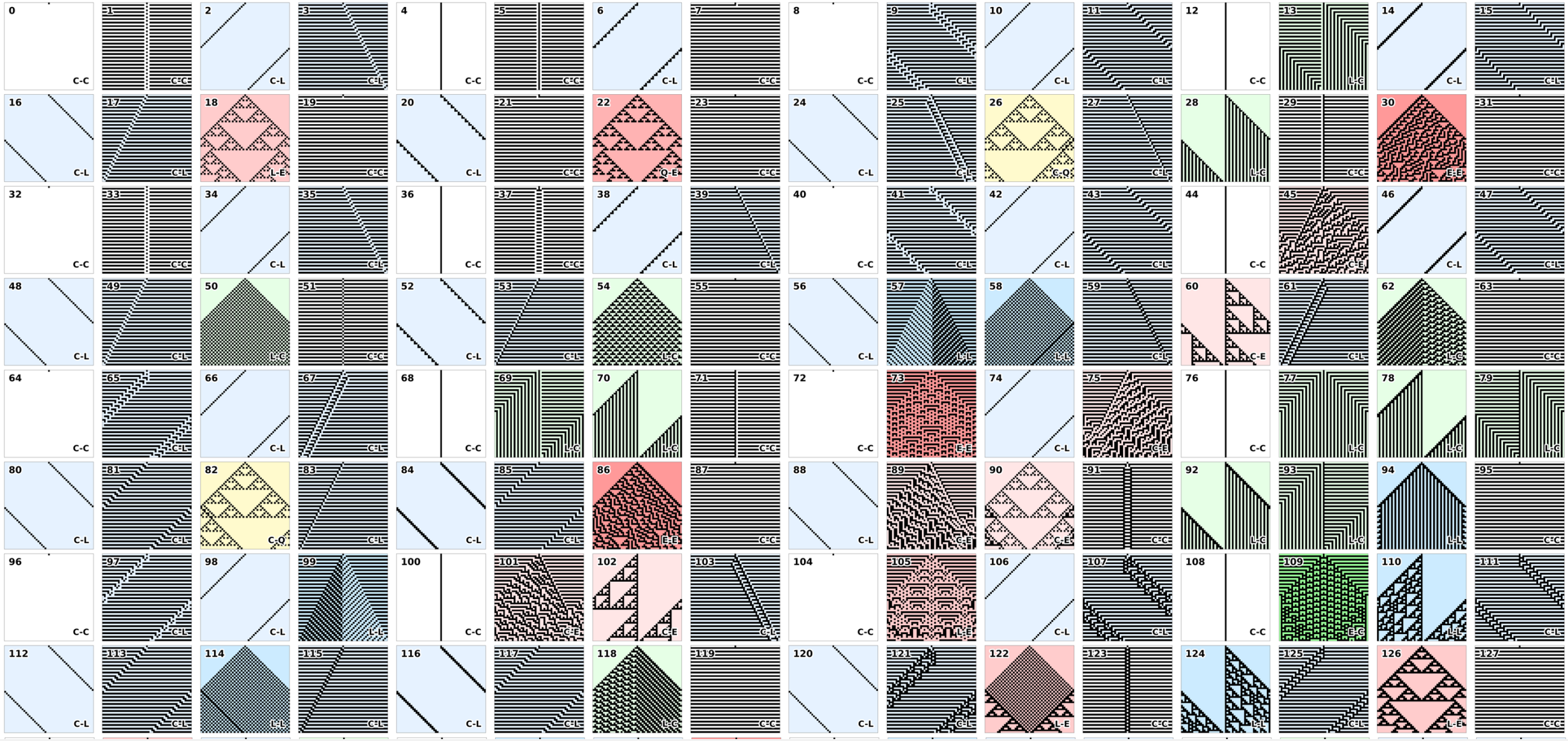


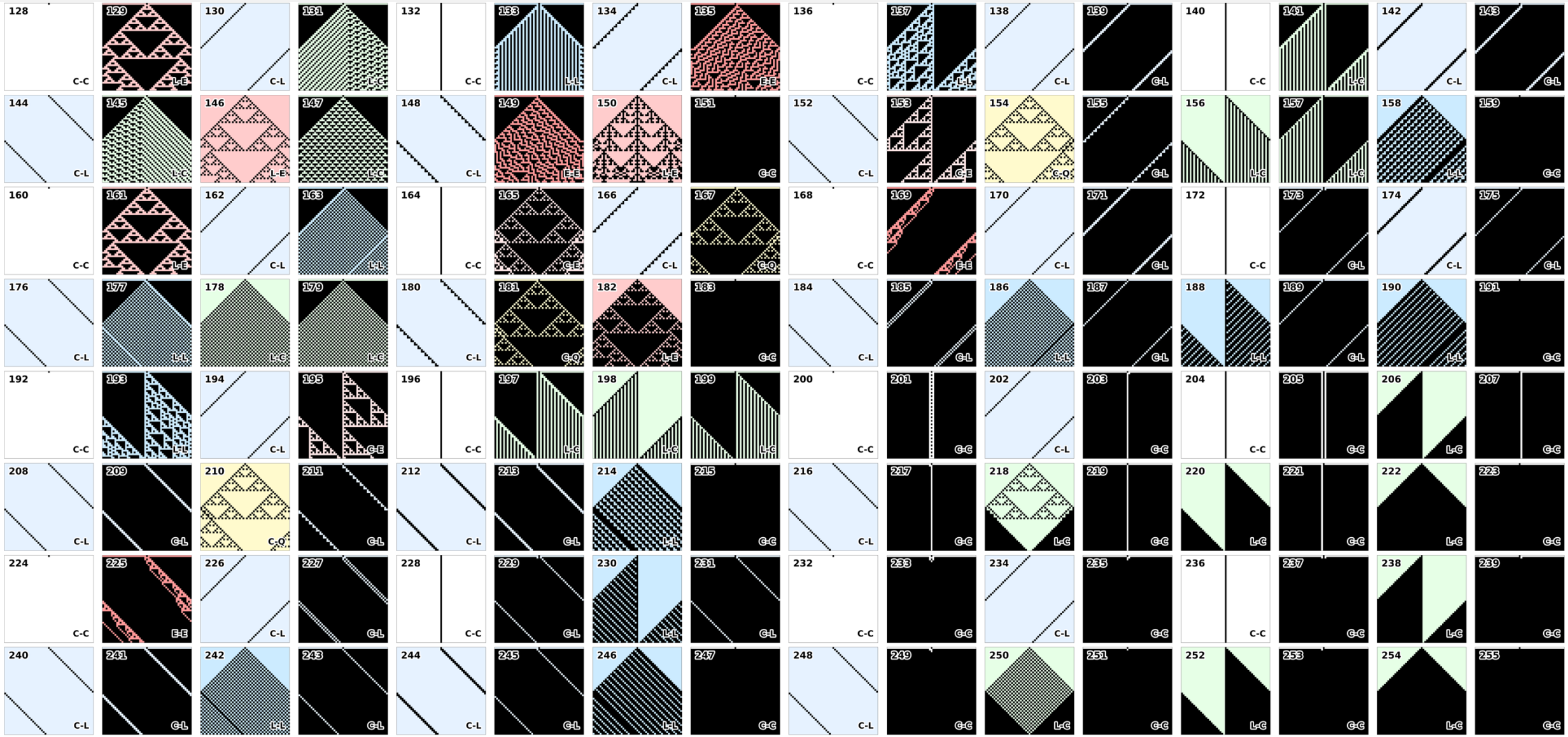

128 (499, 1) C
129 (103, 4) L
130 (499, 0) C
131 (499, 249) L
132 (499, 0) C
133 (499, 248) L
134 (499, 0) C
135 (35, 156k) E
136 (499, 1) C
137 (499, 8044) L
138 (499, 0) C
139 (499, 1) C
140 (499, 0) C
141 (499, 496) L
142 (499, 1) C
143 (499, 2) C
144 (499, 0) C
145 (499, 249) L
146 (103, 4) L
147 (499, 247) L
148 (499, 0) C
149 (35, 156k) E
150 (51, 1) L
151 (499, 1) C
152 (499, 0) C
153 (51, 0) C
154 (499, 0) C
155 (499, 1) C
156 (499, 496) L
157 (499, 497) L
158 (499, 249) L
159 (499, 1) C
160 (499, 1) C
161 (103, 4) L
162 (499, 0) C
163 (499, 249) L
164 (499, 0) C
165 (51, 0) C
166 (499, 0) C
167 (499, 1) C
168 (499, 1) C
169 (39, 230k) E
170 (499, 0) C
171 (499, 1) C
172 (499, 0) C
173 (499, 2) C
174 (499, 1) C
175 (499, 1) C
176 (499, 0) C
177 (499, 249) L
178 (499, 248) L
179 (499, 249) L
180 (499, 0) C
181 (499, 1) C
182 (103, 50) L
183 (499, 1) C
184 (499, 0) C
185 (499, 2) C
186 (499, 249) L
187 (499, 1) C
188 (499, 497) L
189 (499, 1) C
190 (499, 249) L
191 (499, 1) C
192 (499, 1) C
193 (499, 8044) L
194 (499, 0) C
195 (51, 0) C
196 (499, 0) C
197 (499, 496) L
198 (499, 496) L
199 (499, 497) L
200 (499, 1) C
201 (499, 1) C
202 (499, 0) C
203 (499, 2) C
204 (499, 0) C
205 (499, 1) C
206 (499, 497) L
207 (499, 1) C
208 (499, 0) C
209 (499, 1) C
210 (499, 0) C
211 (499, 1) C
212 (499, 1) C
213 (499, 2) C
214 (499, 249) L
215 (499, 1) C
216 (499, 0) C
217 (499, 2) C
218 (499, 498) L
219 (499, 1) C
220 (499, 497) L
221 (499, 1) C
222 (499, 249) L
223 (499, 1) C
224 (499, 1) C
225 (39, 230k) E
226 (499, 0) C
227 (499, 2) C
228 (499, 0) C
229 (499, 2) C
230 (499, 497) L
231 (499, 1) C
232 (499, 1) C
233 (499, 4) C
234 (499, 0) C
235 (499, 3) C
236 (499, 0) C
237 (499, 2) C
238 (499, 498) L
239 (499, 2) C
240 (499, 0) C
241 (499, 1) C
242 (499, 249) L
243 (499, 1) C
244 (499, 1) C
245 (499, 1) C
246 (499, 249) L
247 (499, 1) C
248 (499, 0) C
249 (499, 3) C
250 (499, 498) L
251 (499, 2) C
252 (499, 498) L
253 (499, 2) C
254 (499, 249) L
255 (499, 1) C

# DUAL CLASSIFICATION

- And finally, we can combine the classifications of the transient and cycles to produce a new classification

# Rule Complexity Visuals




128 C-C
129 L-E
130 C-L
131 L-C
132 C-C
133 L-L
134 C-L
135 E-E
136 C-C
137 L-L
138 C-L
139 C-L
140 C-C
141 L-C
142 C-L
143 C-L
144 C-L
145 L-C
146 L-E
147 L-C
148 C-L
149 E-E
150 L-E
151 C-C
152 C-L
153 C-E
154 C-O
155 C-L
156 L-C
157 L-C
158 L-L
159 C-C
160 C-C
161 L-E
162 C-L
163 L-L
164 C-C
165 C-E
166 C-L
167 C-O
168 C-C
169 E-E
170 C-L
171 C-L
172 C-C
173 C-L
174 C-L
175 C-L
176 C-L
177 L-L
178 L-C
179 L-C
180 C-L
181 C-O
182 L-E
183 C-C
184 C-L
185 C-L
186 L-L
187 C-L
188 L-L
189 C-L
190 L-L
191 C-C
192 C-C
193 L-E
194 C-L
195 L-E
196 C-C
197 L-C
198 L-C
199 L-C
200 C-C
201 C-C
202 C-L
203 C-C
204 C-C
205 C-C
206 L-C
207 C-C
208 C-L
209 C-L
210 C-O
211 C-L
212 C-L
213 C-L
214 L-L
215 C-C
216 C-L
217 C-C
218 L-C
219 C-C
220 L-C
221 C-C
222 L-C
223 C-C
224 C-C
225 E-E
226 C-L
227 C-L
228 C-C
229 C-L
230 L-L
231 C-L
232 C-C
233 C-C
234 C-L
235 C-C
236 C-C
237 C-C
238 L-C
239 C-C
240 C-L
241 C-L
242 L-L
243 C-L
244 C-L
245 C-L
246 L-L
247 C-C
248 C-L
249 C-C
250 L-C
251 C-C
252 L-C
253 C-C
254 L-C
255 C-C

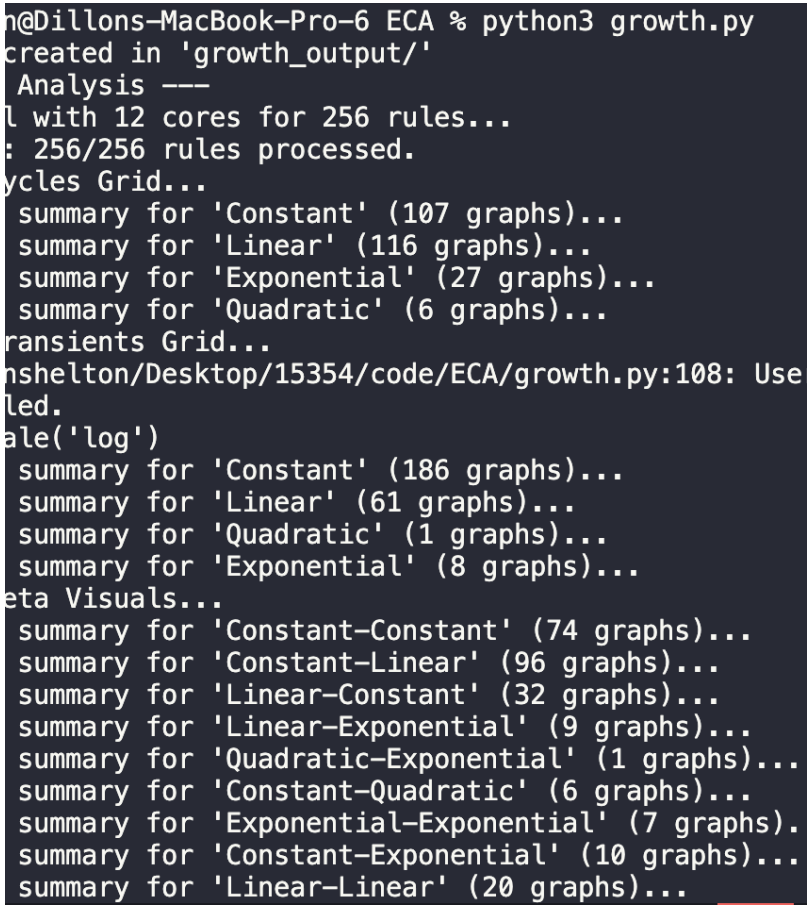
# PART 3

Analysis

# OKAY, THAT WAS A LOT

I'll give a quick summary of each of the meta-classifications.

I don't bother removing mirror/inverse rules

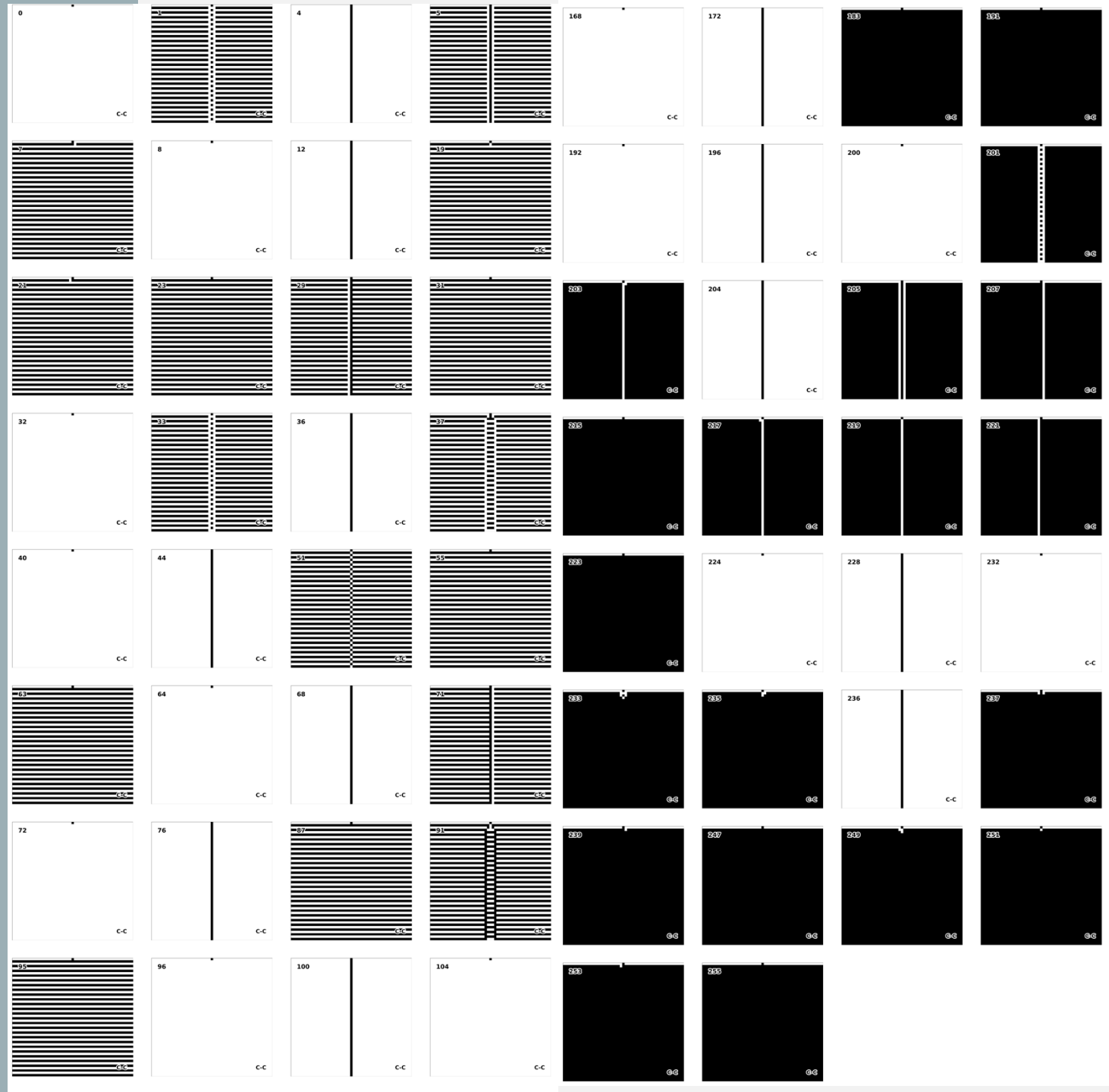


(transient class) - (period class)

# CONSTANT CONSTANT

74 Rules

Characterized by all-white, all-black, or alternating all-black all-white rows, with constant behavior down the center.

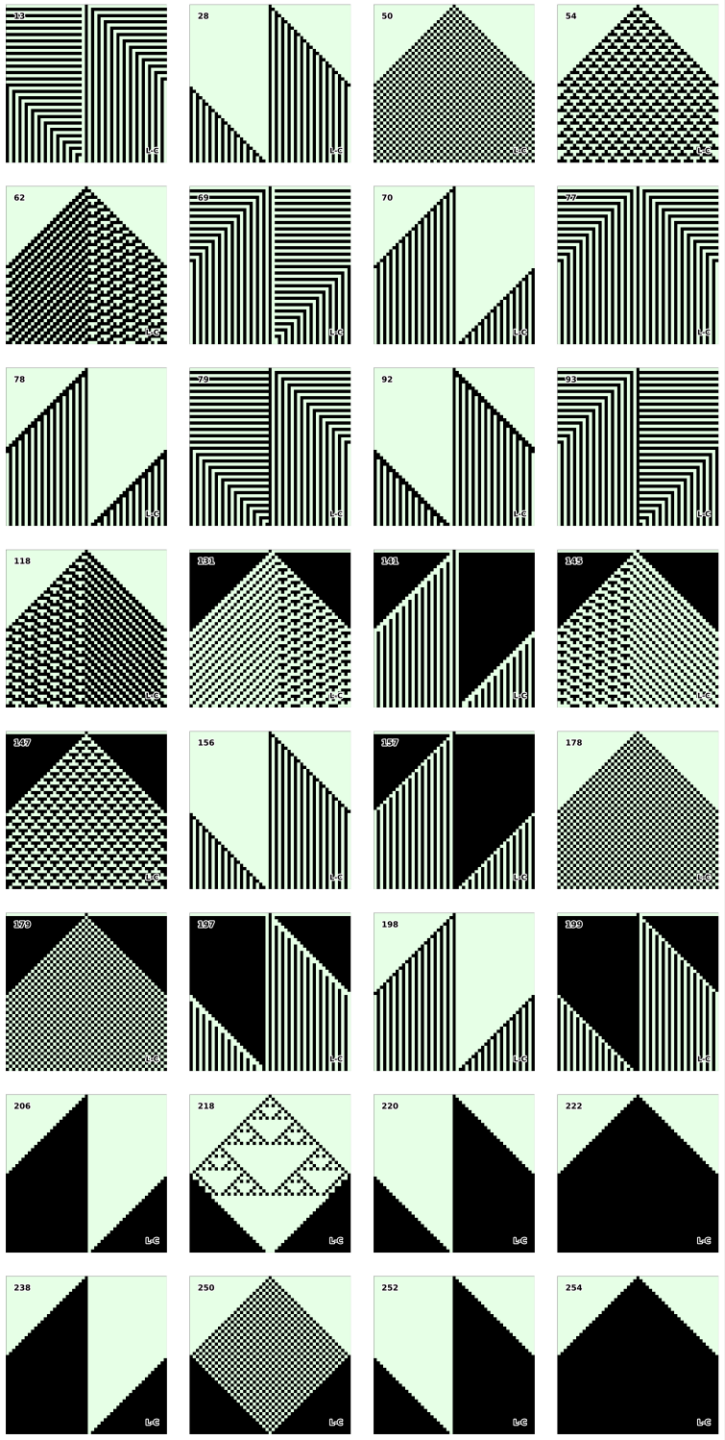


(transient class) - (period class)

# LINEAR CONSTANT

32 Rules

Characterized by grids which propagate out one cell at a time. Once both sides hit the edge, the behavior forms a cycle.

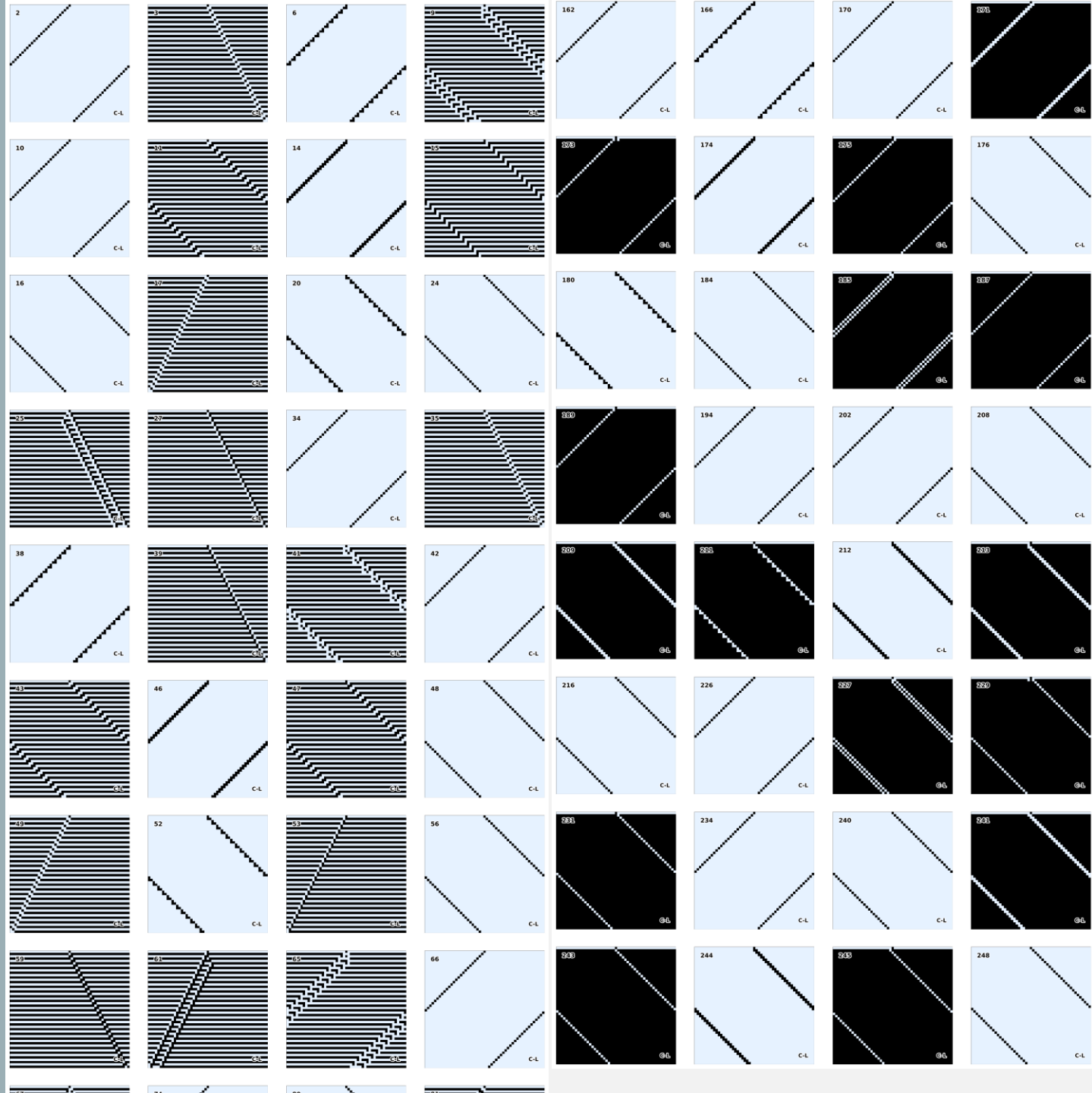


(transient class) - (period class)

# CONSTANT LINEAR

96 Rules

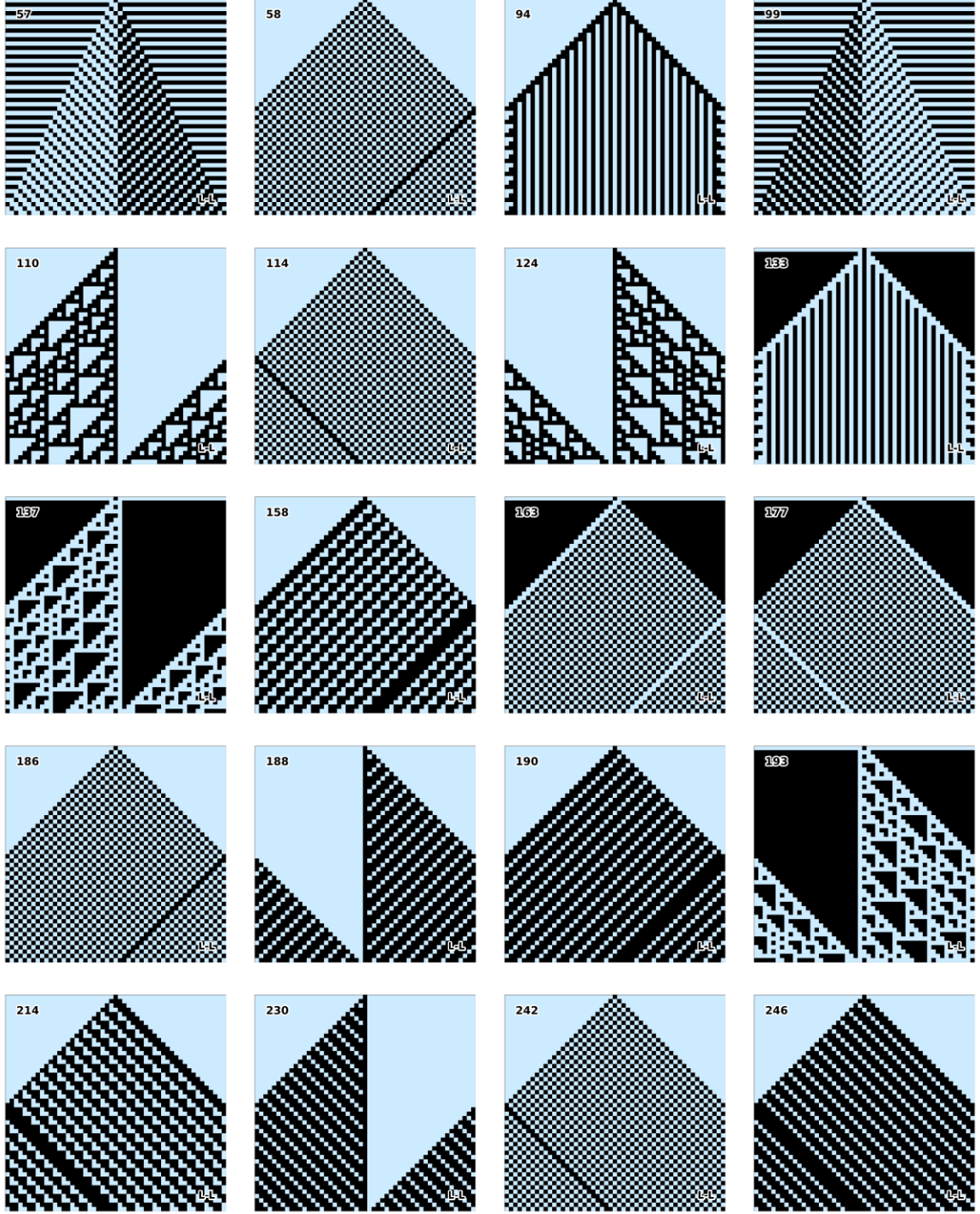
Characterized lines which shoot in one direction. Graphs repeat when the ‘line’ wraps around and returns to the center.



(transient class) - (period class)

# LINEAR LINEAR

20 Rules



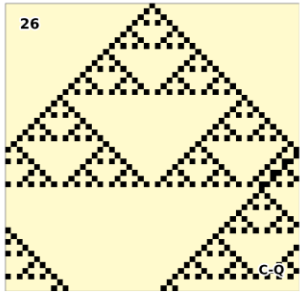
(transient class) - (period class)

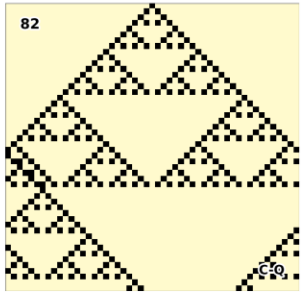
# CONSTANT QUADRATIC

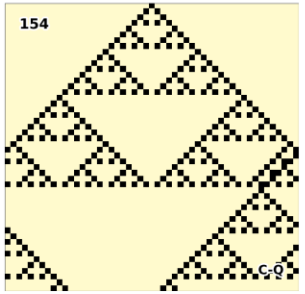
6 Rules

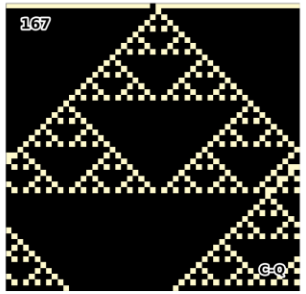
Quadratic periods are very interesting in that they are not actually quadratic. Take a look at the right- It doubles at powers of two, then increases linearly.

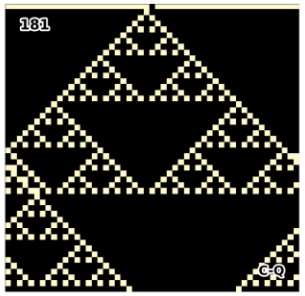
They are certainly worth looking at in the future.

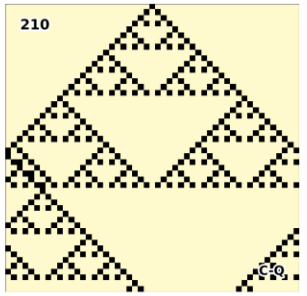


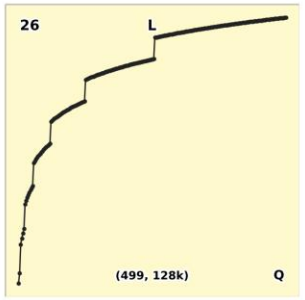



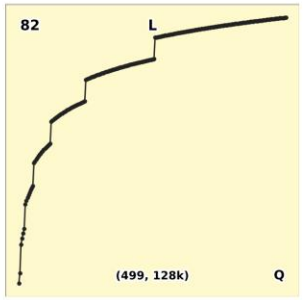



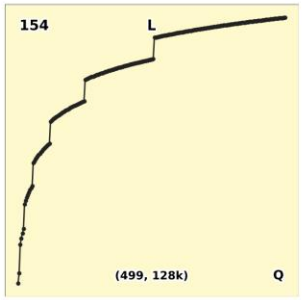



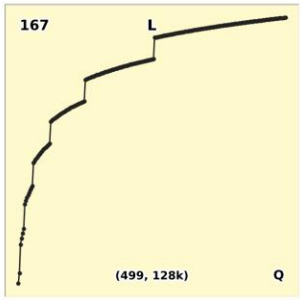



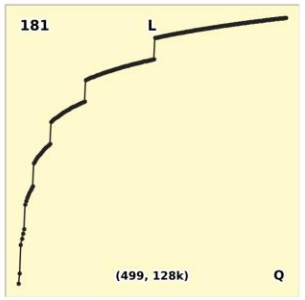



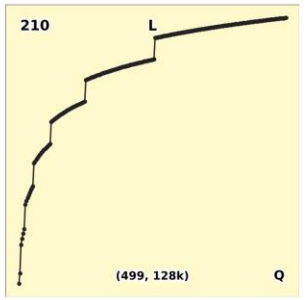



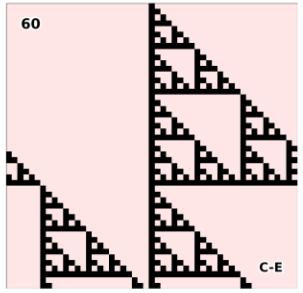



(transient class) - (period class)

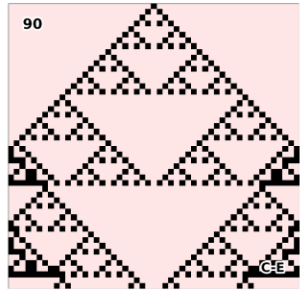
# CONSTANT EXPONENTIAL

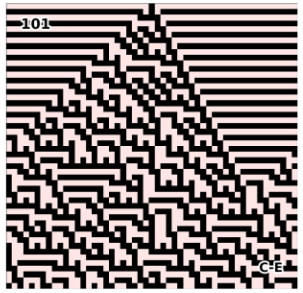
10 Rules

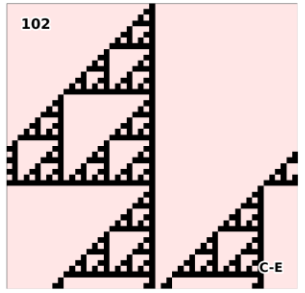
These hit the same transient every time, but take exponentially longer to get back to it.

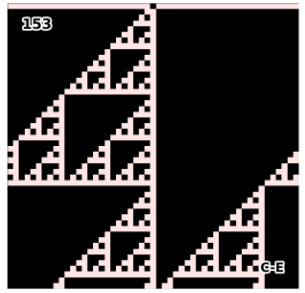



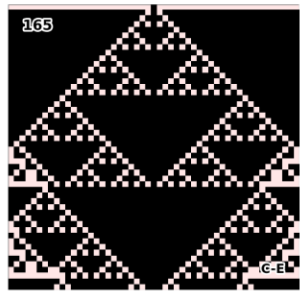


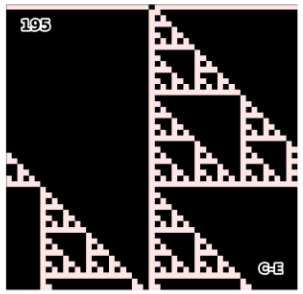



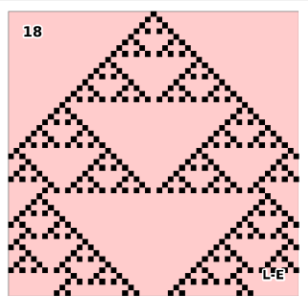



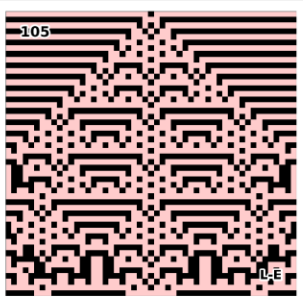
(transient class) - (period class)

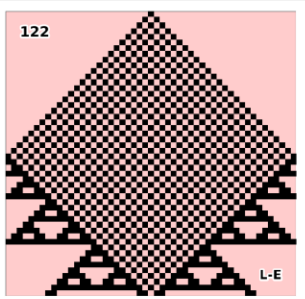
# LINEAR EXPONENTIAL

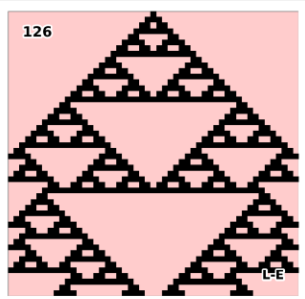
9 Rules

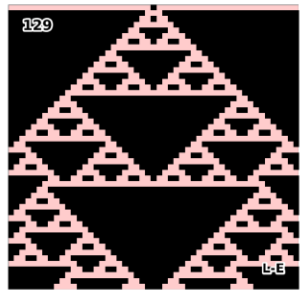
Again, the ECA ‘branches out’ until it hits the wraparound, but this time explores almost every other state before forming a cycle.

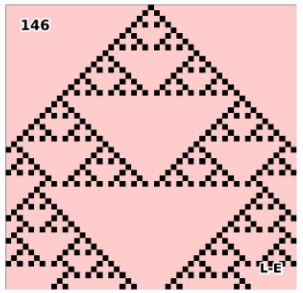


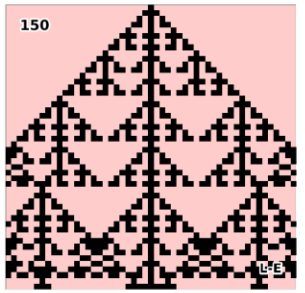



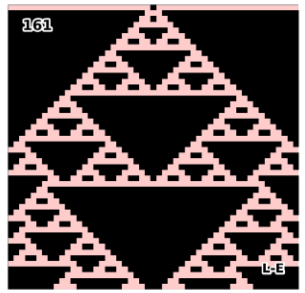



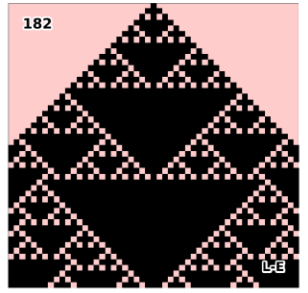



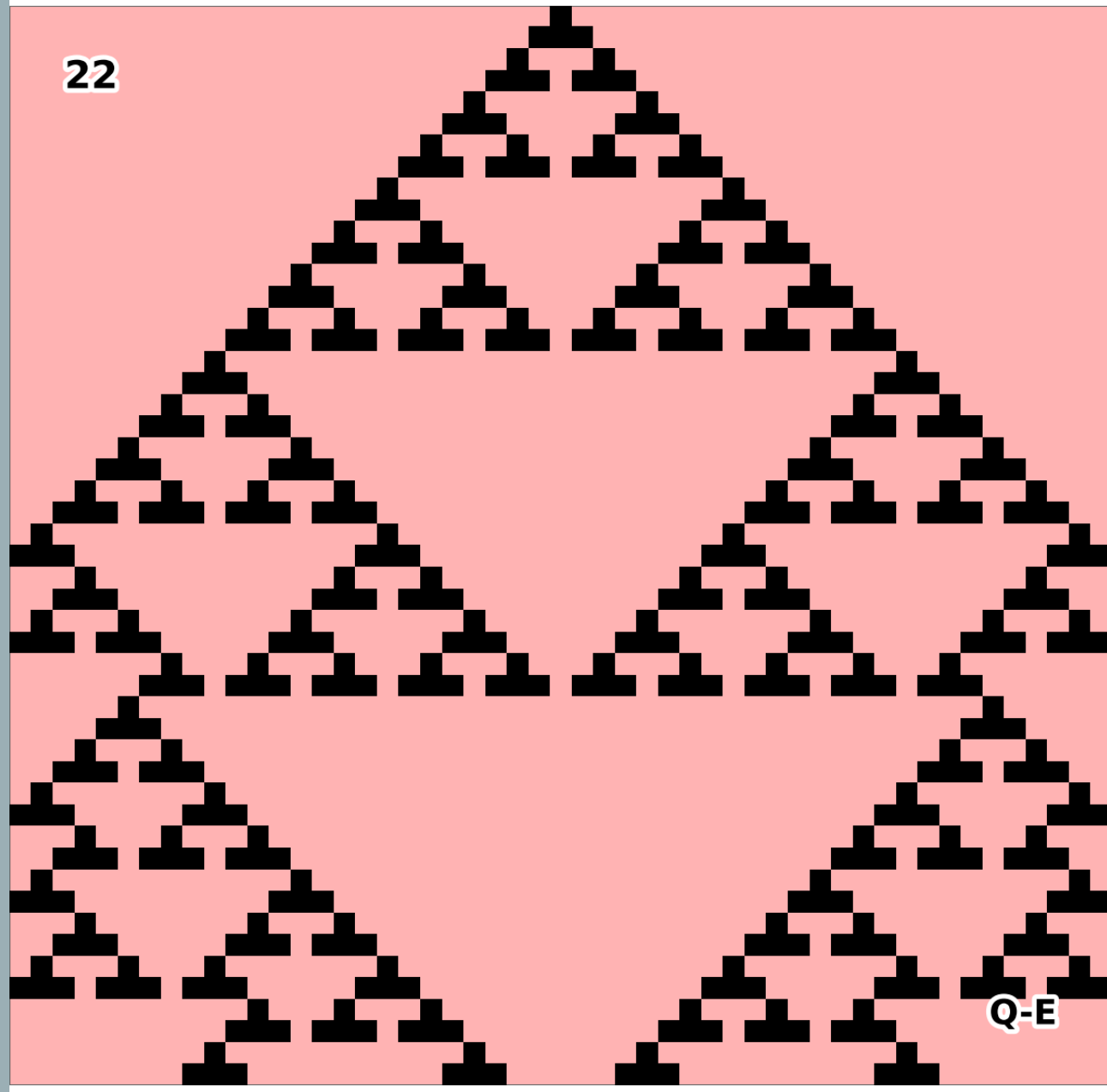



(transient class) - (period class)

# QUADRATIC EXPONENTIAL

1 Rules

I'm not sure how to interpret quadratic transients.

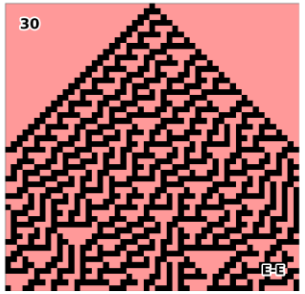


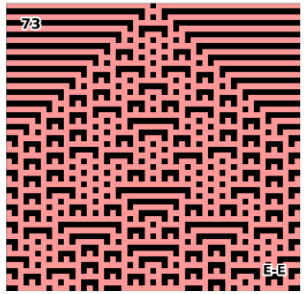
(transient class) - (period class)

# EXPONENTIAL EXPONENTIAL

7 Rules

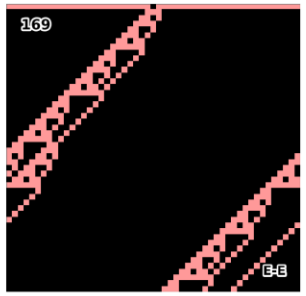
These are generally seem as the ‘chaotic’ rules.

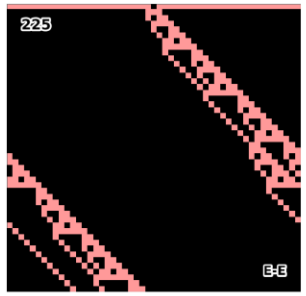








# LIMITATIONS

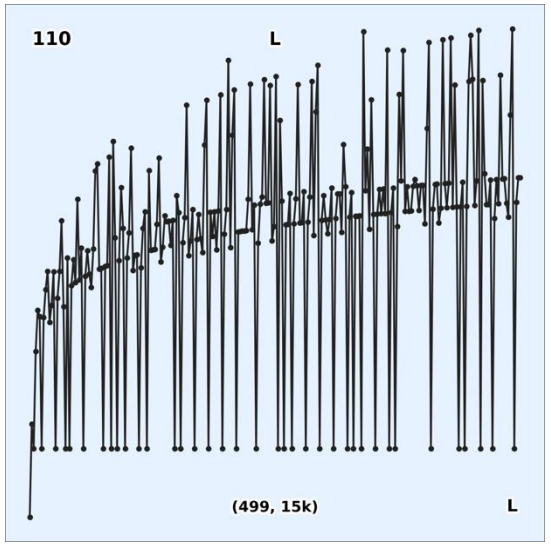
# CONCLUSION

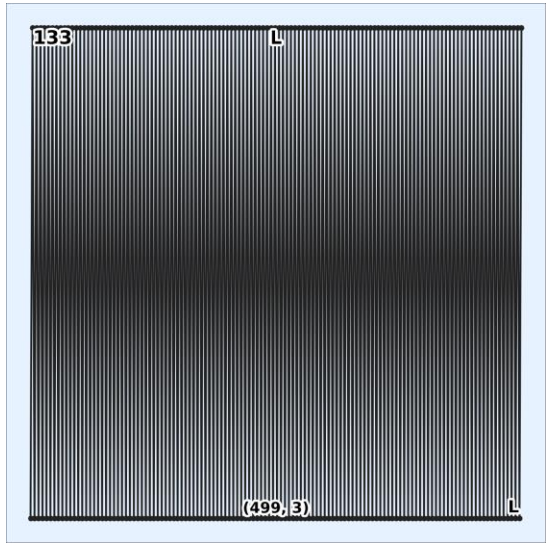
- Classifying ECA based on their cycle length and transient growth can provide valuable intuitions on how the ECAs behave.
- However, the complexity classes are not descriptive enough to describe certain automata.
- Nonetheless, investigating these patterns can provide new insights on ECA, and the graphs generated here are a first step in doing so.

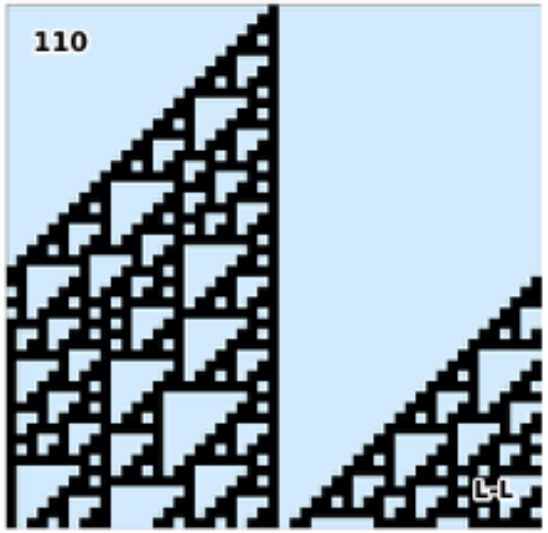
# OSCILLATING LENGTHS

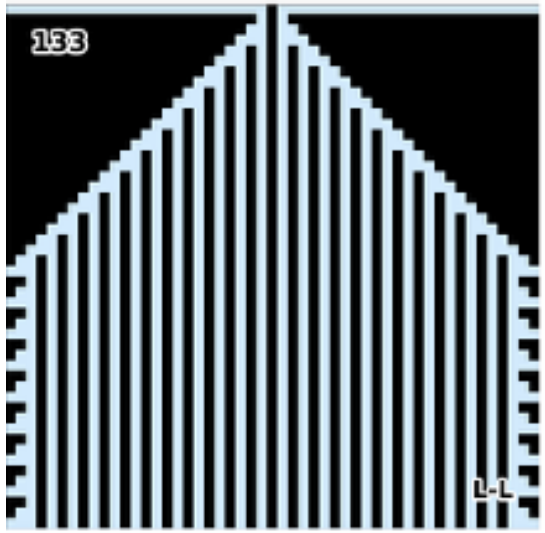
Rule 110, which is universal, has wild jumps.

Rule 133 is stuck between two values, Due to parity.



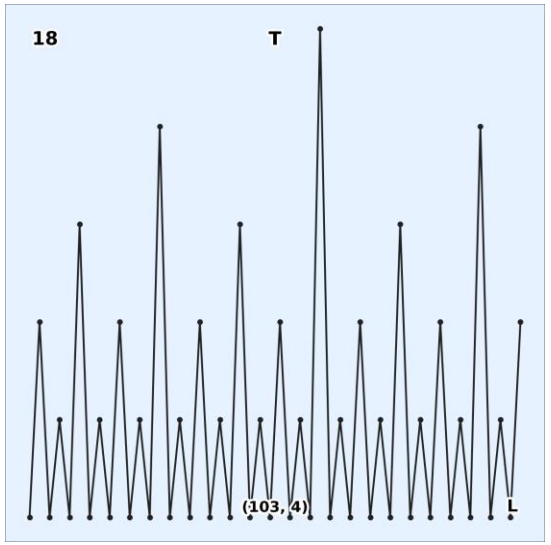



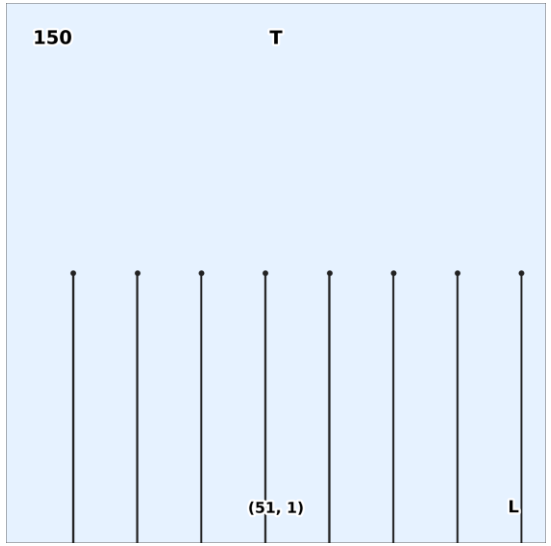



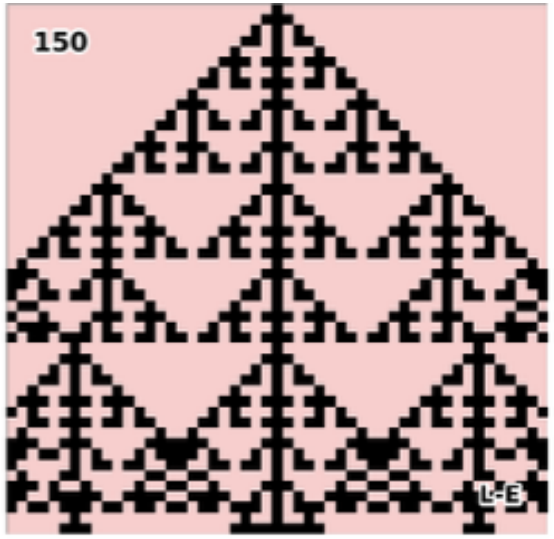
# OSCILLATING TRANSIENTS

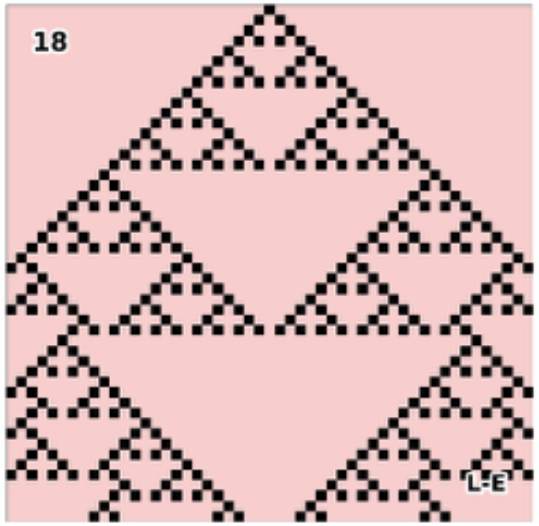
Rule 18, has interesting behavior that Probably depends on the binary representation of n.

Rule 133 has transients probably depending on the factors of n.



# OSCILLATING BOTH- RULE 109

These classifications are due to the atrocity
That is my classify function.

But either way they would not really fit
Any of the complexity classes I proposed.